



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES
INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

SendSeg: aplicación para la mejora de la seguridad de los
senderistas en escenarios de caza

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas
Software

AUTOR/A: Ballester Nicolás, Miguel

Tutor/a: Canós Cerdá, José Hilario

Cotutor/a: Llavador Campos, Manuel

CURSO ACADÉMICO: 2022/2023

Resumen: Es cada vez más frecuente que las actividades de senderismo se vean interrumpidas por las actividades relacionadas con la caza, lo cual lleva a que en ocasiones las actividades planeadas deban cancelarse al llegar a destino o, lo que es peor, una vez iniciadas, debido a las advertencias de seguridad instaladas por cazadores en amplias áreas de montaña. Desde un punto de vista neutral (esto es, ni a favor de unos ni de otros), el origen del conflicto radica en la falta de información pública acerca de las actividades cinegéticas que se desarrollan en nuestro territorio. Por ello, en este TFM se pretende desarrollar un recurso de información que permita, por un lado, a las sociedades de cazadores publicar información sobre las áreas sensibles en determinadas fechas y, por otro, a los senderistas consultar dichas áreas para así planificar actividades seguras. La información puede incluir tanto descripción textual de la actividad de caza a desarrollar, como el dibujo en un mapa del polígono restringido.

Palabras clave: Mashup; desarrollo web y app; ingeniería del software; integración de información.

Abstract: It is increasingly common for hiking activities to be interrupted by hunting-related activities. This implies sometimes to finally cancel activities when you arrive at the destiny, or even worse, when they have already started, due to the safety warnings posted by hunters in extensive mountain areas. From a neutral point of view, the origin of the conflict is caused by the lack of public information about hunting activities which take place in our territory. Therefore, this TFM aims to develop an information resource that could allow, on the one hand, hunting societies to publish information about sensible areas in specific dates and, on the other hand, hikers to be able to consult those areas in order to check and verify that their activities are safe. The information may include both a textual description of the hunting activity to be carried out as well as a polygon drawing in a map representing the restricted area.

Key words: Mashup; web and app development; software engineering; information integration.

Resum: Cada vegada es més freqüent que les activitats de senderisme es vegen interrompudes per les activitats relacionades amb la caça, la qual cosa porta a que en ocasions les activitats planejades hagen de cancel·lar-se a l'hora d'arribar al destí, o el que és pitjor, una vegada han sigut iniciades, degut a les advertències de seguretat instal·lades per caçadors en àrees de muntanya. Des d'un punt de vista neutral, l'origen del conflicte rau en la manca d'informació pública sobre les activitats cinegètiques que es desenvolupen al nostre territori. Per això, aquest TFM preten desenvolupar un recurs d'informació que permeti, per una banda, a les societats de caçadors publicar informació sobre les àrees sensibles en determinades dades, i d'altra banda, als senderistes consultar eixes àrees per així planificar activitats segures. La informació pot incloure tant una descripció textual de l'activitat de caça a desenvolupar, com un dibuix en un mapa del polígon restringit.

Paraules clau: Mashup; desenvolupament web i app; enginyeria del software; integració de l'informació.

ÍNDICE

Capítulo 1: Introducción.....	11
Motivación	11
Estado del arte	12
Objetivos	14
Metodología	14
Planificación	17
Capítulo 2: Fase de especificación de requisitos.....	19
Introducción	19
Propósito	19
Elicitación	19
Fuente de requisitos.....	19
<i>Stakeholders</i>	19
Alcance del producto	20
Funciones del producto.....	20
Requisitos funcionales.....	21
Requisitos no funcionales	23
Características de los usuarios	24
Restricciones	25
Suposiciones y dependencias.....	25
Requisitos futuros	26
Capítulo 3: Análisis, Diseño y Prototipado	27
Diagrama de casos de uso	27
Diagrama de clases.....	29
Diagrama relacional	30
Diagrama de flujo de navegación.....	31
Diagrama de flujo de navegación de usuario senderista	31
Diagrama de flujo de navegación de usuario Sociedad de Cazadores.....	32
Diagrama de flujo de navegación de usuario Administrador.....	33
Prototipado	33
Selección de API para Mapas y Polígonos	35
Capítulo 4: Implementación	37
Visual Studio Code.....	37
Node JS.....	37
Express	38

Express-session	39
Bulma	39
EJS.....	40
Mysql2.....	41
Bcrypt	41
Nodemailer.....	42
Multer.....	44
JSTS.....	44
API Google Maps	45
MySQL	49
Python	52
Capítulo 5: Testing	57
Pruebas unitarias.....	57
Pruebas integración	58
Pruebas compatibilidad.....	59
Pruebas de aceptación	59
Capítulo 6: Conclusiones	60
Líneas futuras de trabajo	60
Sociedades Cazadores	60
Senderistas	61
Administraciones públicas.....	62
Desarrolladores de otras aplicaciones	62
Bibliografía.....	63
Anexo 1: Descripciones Diagramas Casos de Uso	65
Anexo 2: Mockups Prototipado	75
Anexo 3: Script base de datos	83
Anexo 4: Manual de usuario	85
Anexo 5: Despliegue de Sendseg	103

Índice de ilustraciones

Ilustración 1. Resultados cadena de búsqueda	12
Ilustración 2. Aplicación consulta partidas caza mayor Diputación Foral de Álava	13
Ilustración 3. LandShare	13
Ilustración 4. Metodología Scrum	15
Ilustración 5. Tablero Kanban	15
Ilustración 6. Bases metodología Lean	16
Ilustración 7. Metodología en cascada	16
Ilustración 8. Metodología en Espiral	17
Ilustración 9. Diagrama de casos de uso general	27
Ilustración 10. Diagrama de clases	29
Ilustración 11. Diagrama relacional	30
Ilustración 12. Diagrama de flujo de navegación de Senderista	31
Ilustración 13. Diagrama de flujo de navegación de Sociedad de Cazadores autenticada	32
Ilustración 14. Diagrama de flujo de navegación de Administrador	33
Ilustración 15 Prototipo ventana Inicio	34
Ilustración 16. Prototipo ventana consulta zonas caza Senderistas	35
Ilustración 17. Logo Bulma.io	39
Ilustración 18. Código archivo errores.ejs	40
Ilustración 19. Código en servidor para la renderización adecuada de la plantilla pasando valores	40
Ilustración 20. Resultado de la encriptación y valor almacenado en la base de datos	42
Ilustración 21. Correo creado para el proyecto	43
Ilustración 22. Creación de clave de aplicación en Google	43
Ilustración 23. Resultado al enviar el correo automáticamente	44
Ilustración 24. APIs y servicios ofrecidas por Google Maps Platform	46
Ilustración 25. Programa gratuito Google Cloud	46
Ilustración 26. Creación proyecto Sendseg en Google Cloud Console	47
Ilustración 27. Proyecto Sendseg creado en Google Cloud Console	47
Ilustración 28. APIs y servicios	47
Ilustración 29. Creación de credenciales I	48
Ilustración 30. Creación de credenciales II	48
Ilustración 31. Creación de credenciales III	48
Ilustración 32. Credencial creada para Sendseg	49
Ilustración 33. Conexión root MySQL worckbench	49
Ilustración 34. Creación del schema	50
Ilustración 35. Creación y configuración de la tabla asociacion_cazadores	50
Ilustración 36. Creación y configuración tabla zona_de_caza	50
Ilustración 37. Creación y configuración tabla coordenadas	51
Ilustración 38. Creación y configuración tabla informe_actividad	51
Ilustración 39. Resultado petición INE	53
Ilustración 40. Código Python 1	54
Ilustración 41. Código Python 2	54
Ilustración 42. Archivo municipiosOrdenados.txt	55
Ilustración 43. Herramientas de desarrollo navegador	57
Ilustración 44. Pruebas en Postman	58
Ilustración 45. Prototipo ventana consulta zonas caza Senderistas	75

Ilustración 46. Prototipo ventana Sociedades sin credenciales	75
Ilustración 47. Prototipo ventana Iniciar sesión.....	76
Ilustración 48. Prototipo ventana Registro sociedades	76
Ilustración 49. Prototipo ventana Inicio sesión Sociedad sin actividades creadas.....	77
Ilustración 50. Prototipo ventana selección inicio y fin de actividad de caza	77
Ilustración 51. Prototipo ventana creación actividad.....	78
Ilustración 52. Prototipo ventana Sociedad con actividades caza creadas	78
Ilustración 53. Prototipo ventana Inicio Administrador	79
Ilustración 54. Prototipo ventana Administrador gestión sociedades	79
Ilustración 55. Prototipo ventana Administrador gestión actividades caza	80
Ilustración 56. Prototipo ventanas Administrador gestión coordinadas.....	80
Ilustración 57. Prototipo ventana Administrador edición de datos de Sociedad.....	81
Ilustración 58. Prototipo ventana responsive Inicio Sociedad	81
Ilustración 59. Prototipo ventana responsive consulta senderistas.....	82
Ilustración 60. Prototipo ventana responsive Administrador gestión coordinadas	82
Ilustración 61. Vista Inicio Sendseg	85
Ilustración 62. Vista Senderistas.....	86
Ilustración 63. Vista error consulta senderistas	86
Ilustración 64. Vista consulta senderistas correcta	87
Ilustración 65. Vista Sociedades Cazadores.....	87
Ilustración 66. Vista Registro	88
Ilustración 67. Vista Inicio sesión	88
Ilustración 68. Vista sociedad cazadores con sesión iniciada sin actividades	89
Ilustración 69. Vista creación actividad sociedad cazadores I.....	89
Ilustración 70. Vista creación actividad sociedad cazadores II.....	90
Ilustración 71. Vista sociedad cazadores con actividad creada	91
Ilustración 72. Consulta del polígono o área de una actividad de sociedad de cazadores	91
Ilustración 73. Subida informe de una actividad I.....	92
Ilustración 74. Subida de informe de una actividad II	92
Ilustración 75. Informe de actividad subido y posibilidad de descarga	92
Ilustración 76. Posibilidad de modificar el perfil de sociedad cazadores	93
Ilustración 77. Modificación del perfil.....	93
Ilustración 78. Creación de actividad con la aparición de otras definidas previamente.....	94
Ilustración 79. Vista del Administrador para comprobar las entidades y acciones que se pueden hacer.....	95
Ilustración 80. Despliegue de Acciones con cada recurso.....	95
Ilustración 81. Vista de Acciones sobre Sociedades Cazadores del Administrador	96
Ilustración 82. Vista Administrador para crear nueva sociedad cazadores.....	96
Ilustración 83. Vista modificación de una sociedad de cazadores por Administrador.....	97
Ilustración 84. Vista Acciones de Actividades Caza por el Administrador.....	98
Ilustración 85. Vista edición actividad de caza por Administrador I.....	98
Ilustración 86. Vista edición actividad de caza por Administrador II.....	99
Ilustración 87. Vista modificación actividad de caza por Administrador III.....	100
Ilustración 88. Vista Acciones del recurso coordinadas por Administrador.....	101
Ilustración 89. Vista Acciones del recurso Informes por Administrador	101
Ilustración 90. Descarga de Informe de actividad de caza	102
Ilustración 91. Página de Railway	103
Ilustración 92. Página de Github	104

Ilustración 93. Creación nuevo repositorio Github	104
Ilustración 94. Comando para inicializar Git en Visual Studio	105
Ilustración 95. Comando para añadir archivos al siguiente commit	105
Ilustración 96. Comando para establecer el commit y su nombre.....	105
Ilustración 97. Comando para definir el origen y destino del commit	105
Ilustración 98. Comando para hacer push del commit	106
Ilustración 99. Repositorio después de commit.....	106
Ilustración 100. Archivo .gitignore	106
Ilustración 101. Railway opciones New project.....	107
Ilustración 102. Provision MySQL.....	107
Ilustración 103. Railway MySQL query	108
Ilustración 104. Tablas base de datos creadas en Railway	108
Ilustración 105. Despliegue aplicación a partir del repositorio en Railway	109
Ilustración 106. Variables entorno base de datos Railway.....	109
Ilustración 107. Establecer variables en despliegue aplicación Railway	110
Ilustración 108. Uso de variables de entorno.....	110

Índice de gráficos

Gráfico 1. Diagrama de Gantt	18
------------------------------------	----

Índice de tablas

Tabla 1. Stakeholders.....	20
Tabla 2. Caso uso: Informarse sobre Sendseg	28
Tabla 3. Caso uso: Comprobar actividades de caza	28
Tabla 4. Comparación API Google Maps y OpenStreetMap	36
Tabla 5. Caso uso: Registrarse	65
Tabla 6. Caso uso: Iniciar Sesión.....	65
Tabla 7. Caso uso: Buscar lugares en el mapa	66
Tabla 8. Caso uso: Navegar por el mapa.....	66
Tabla 9. Caso uso: Definir horario actividad, características y área	66
Tabla 10. Caso uso: Crear actividad de caza	67
Tabla 11. Caso uso: Editar perfil sociedad de cazadores	67
Tabla 12. Caso uso: Ver área de caza.....	68
Tabla 13. Caso uso: Subir informe	68
Tabla 14. Caso uso: Descargar informe	68
Tabla 15. Caso uso: Consultar información acciones disponibles para cada entidad	69
Tabla 16. Caso uso: Listar Sociedades Cazadores.....	69
Tabla 17. Caso uso: Crear usuario sociedad cazadores	70
Tabla 18. Caso uso: Editar usuario sociedad cazadores	70
Tabla 19. Caso uso: Eliminar usuario sociedad cazadores.....	71
Tabla 20. Caso uso: Listar actividades caza	71
Tabla 21. Caso uso: Editar actividades caza.....	72
Tabla 22. Caso uso: Eliminar actividades caza	72
Tabla 23. Caso uso: Listar coordenadas de Zonas de Caza	73
Tabla 24. Caso uso: Listar informes	73
Tabla 25. Caso uso: Descargar Informe	74
Tabla 26. Caso uso: Eliminar informe	74

Capítulo 1: Introducción

En este primer capítulo de la memoria del trabajo final de máster o TFM, se expondrá el principal problema que se ha propuesto resolver así como la motivación detrás para el desarrollo de este. Se indicarán los objetivos que se pretenden alcanzar y se explicará cual es el estado del arte del problema a solventar, es decir, qué medios se están ofreciendo en estos momentos que intenten resolverlo. Después se explicará la metodología que se ha usado en el proyecto para desarrollar el software que ofrece una solución al problema. Por último, se mostrará la planificación que se ha seguido para el desarrollo del trabajo.

Motivación

La motivación para hacer este proyecto va muy ligada al propio problema para el que se pretende desarrollar y ofrecer una solución.

En primer lugar vamos a introducir el problema, este implica a dos colectivos: **senderistas** y **sociedades de cazadores**. Radica en que hoy en día, no se ofrecen medios o al menos lo suficientemente buenos como para que los senderistas, antes o durante la realización de una actividad, puedan estar prevenidos y saber a ciencia cierta donde y cuando se está desarrollando una actividad de caza por parte de las sociedades. A raíz de esta falta de transparencia y flujo en la transmisión de la información por los canales adecuados, se pueden vivir situaciones desagradables para cualquiera de las partes y que podrían evitarse o al menos prevenirse.

Esto no pretende señalar a una de las partes específicamente como la que genera el problema, sino que la naturaleza de este atañe e involucra a dos partes que comparten un mismo recurso, el espacio natural, y que tienen que colaborar para encontrar la solución con la cual un senderista sepa con el cien por cien de conocimiento de causa que va a ir a hacer su ruta a una zona con total seguridad y que no se adentra en un área donde se está cazando y que puede molestar a la sociedad y generar malestar entre ambos.

Bien es cierto que las sociedades en un ejercicio de prevención de problemas hacen intentos por prevenir incursiones en zonas de caza a los senderistas dejando avisos en la zona en la que están realizando *in situ* la cacería, pero estos no siempre funcionan debido a que las zonas de caza pueden ser muy amplias y esta solución no es suficiente por evitar que alguien se “cuele” sin ni siquiera haber podido percatarse de la presencia de los cazadores.

Por lo tanto, conociendo el problema para el que se aportará una solución, la motivación detrás de realizar este proyecto está en que se aplicarán conocimientos que se han ido adquiriendo tanto durante la carrera como ahora en el propio máster con la finalidad de impactar muy positivamente y de proporcionar mayor seguridad a los senderistas durante sus excursiones, y tranquilidad a las sociedades durante el desarrollo de sus actividades, de manera en que la solución obtenida, en caso de implantarse, impactaría como medio de mediación y colaboración para la transmisión de información entre ambos colectivos y así para alcanzar esa seguridad imprescindible durante las actividades de unos u otros.

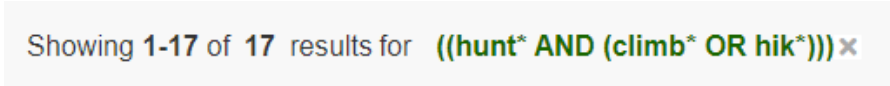
Estado del arte

Una vez descrito el problema existente y la motivación detrás de este proyecto, se ha hecho una búsqueda con la intención de mostrar algunas de las soluciones que se han proporcionado en el pasado para ver qué limitaciones tienen e intentar tanto en este proyecto como en futuras ampliaciones dar una mejor y más completa solución.

En primer lugar, lo que se hizo fue ir a repositorios de artículos y trabajos como IEEE Xplore para introducir cadenas de búsqueda con la finalidad de encontrar documentos o *papers* de proyectos que se hayan realizado en el pasado y tengan que ver con el problema.

Para poder hacer uso completo del repositorio y hacer la búsqueda, me identifiqué con las credenciales que proporciona la UPV. Una vez identificado, la cadena de búsqueda que se usó fue la siguiente:

Ilustración 1. Resultados cadena de búsqueda



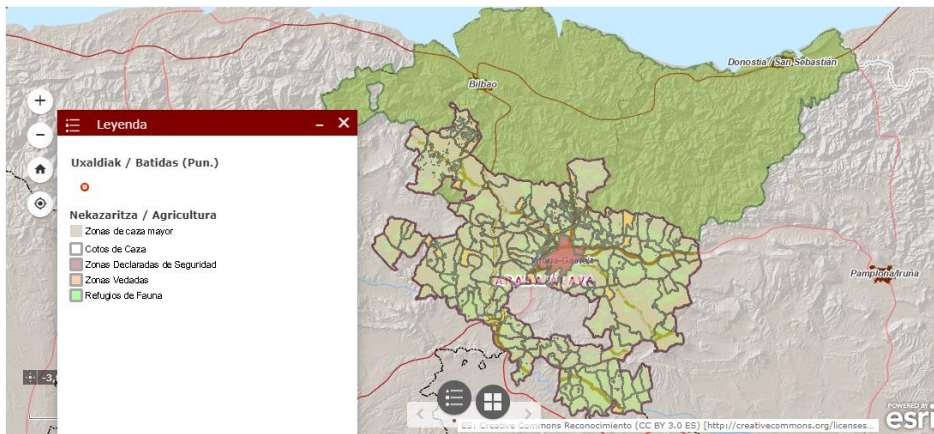
Showing 1-17 of 17 results for **((hunt* AND (climb* OR hik*))) x**

Fuente: Elaboración propia en IEEE Xplore

De este modo, se encontraban todos los trabajos en los que apareciera la raíz de “caza” y de “send” comprendidos entre los años 1973 a 2023. En este caso, se obtenían muy pocos trabajos, solo 17 tal y como se ve en la ilustración anterior, y además, ninguno de ellos tenía nada que ver con el problema a solucionar. Es decir, los resultados obtenidos eran falsos positivos. Por ello, decidí hacer una búsqueda en Google para al menos identificar y observar al menos a nivel de nacional, y si es posible internacional, qué herramientas se han desarrollado y qué sistemas o aplicaciones web están disponibles como posibles soluciones al problema.

Buscando en Google se pudieron encontrar muy pocas aplicaciones web que sí aportan una posible solución al conflicto, la mayoría de estas propuestas vienen encargadas por sociedades y administraciones públicas. De entre todas ellas, cabe destacar una que está en fase de pruebas que realizó la Diputación Foral de Álava, en la cual, únicamente se puede consultar a tiempo real para ver y localizar las partidas de caza mayor, lo cual es un comienzo, pero tampoco sería una solución completa debido a que si un senderista necesita planificar previamente su actividad, y desea consultar las actividades de caza futuras, no puede hacerlo.

Ilustración 2. Aplicación consulta partidas caza mayor Diputación Foral de Álava



Fuente: Diputación Foral de Álava

En el panorama nacional esta es la que más destaca, sin embargo, se ha podido localizar que en Francia, la Federación Francesa de Senderismo, lanzó una aplicación móvil experimental compuesta por dos aplicaciones, una en la que las Sociedades de Cazadores crean las zonas de caza a tiempo real que se denominó Protect Hunt, y otra denominada Land Share para senderistas, ciclistas u otros actores externos a la caza que pueden comprobar a tiempo real donde se está desarrollando la caza.

Esta solución además agrega la posibilidad de cuando la persona que usa Land Share entre en una zona marcada, se le notifique que está en una zona de caza y que debería alejarse. Por lo tanto, esta segunda opción que se ha desarrollado parece bastante mejor que la de Álava pero sigue teniendo el principal inconveniente de que solo se puede consultar a tiempo real. Además de esta limitación, según comentan hay una condición de que se debe de estar a un rango menor de 20 kilómetros con tu ubicación actual respecto a las zonas marcadas para visualizar los polígonos o dibujos que usan para representar las actividades.

Ilustración 3. LandShare



Fuente: MBF-France.fr

Por lo tanto, comentadas las dos propuestas anteriores que son las que más destacan en relación con el trabajo a desarrollar, en ambos casos, identificamos el problema principal de que no se da lugar a que el senderista o ciclista planifique previamente su actividad con la posibilidad de organizar o reorganizar su ruta de manera que con previo conocimiento de causa proceda a llevarla a término.

Estas carencias en las soluciones comentadas serán tenidas en cuenta de cara a desarrollar el presente proyecto.

Objetivos

Una vez introducido el problema, así como presentadas dos de las aplicaciones que se han desarrollado con la intención de solucionarlo, es momento de establecer de manera clara cuál es el objetivo principal de este proyecto y cuáles podrían ser otros objetivos secundarios.

En cuanto al objetivo principal del proyecto es desarrollar una aplicación web que pueda ser utilizada por las sociedades para definir sus actividades de caza y que los senderistas puedan consultar dichas actividades, representadas en un mapa como un polígono, para saber las zonas que son seguras y las que no durante un intervalo de tiempo que se correspondería con la duración de la actividad que vayan a realizar.

Los objetivos secundarios de este proyecto irían desde una perspectiva de crecimiento personal como el propio aprendizaje de tecnologías que no haya usado nunca, así como el uso y refuerzo de aquellas que he podido aprender íntegramente en el máster. Dentro del aprendizaje de nuevas tecnologías o herramientas será vital el interactuar y usar servicios de mapas externos para poder usar estos y definir las actividades por medio de polígonos, cosa que nunca he hecho. También de manera secundaria, la idea sería dejar preparada la base de lo que serían posibles mejoras futuras de la aplicación que puedan ser llevadas a cabo tanto por mi como por otras personas en futuros proyectos que continúen con el desarrollo y ampliación de esta aplicación web.

Metodología

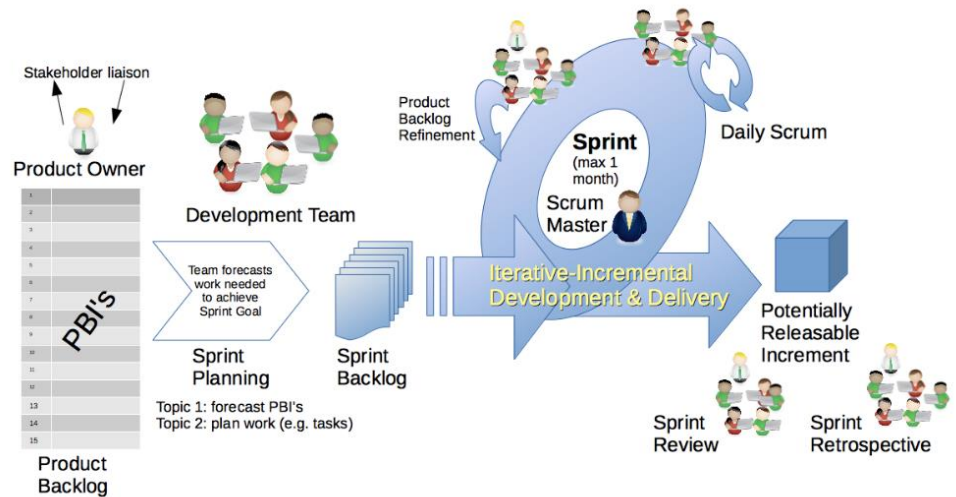
Una vez se conoce el problema a resolver, y que se espera desarrollar un producto software para ello, se debe de establecer la metodología. Esta hace referencia a la forma o modo en la que se va a desarrollar este tipo de producto y como se van a organizar los miembros del equipo que crean el producto.

Dentro de las metodologías de desarrollo de software existen una gran cantidad de opciones, cada una con sus correspondientes ventajas e inconvenientes. Algunas de las metodologías más importantes y conocidas son:

- **Metodologías ágiles**
 - *Scrum*: Conocida también como metodología incremental, se realizan iteraciones de corto periodo de tiempo y fijo denominadas *sprints* (entre 2 y 4 semanas normalmente), donde cada miembro del equipo se centra en unas tareas concretas y repartidas para ir implementando

el software, de modo en que al final de cada *sprint*, se proporciona un resultado completo.

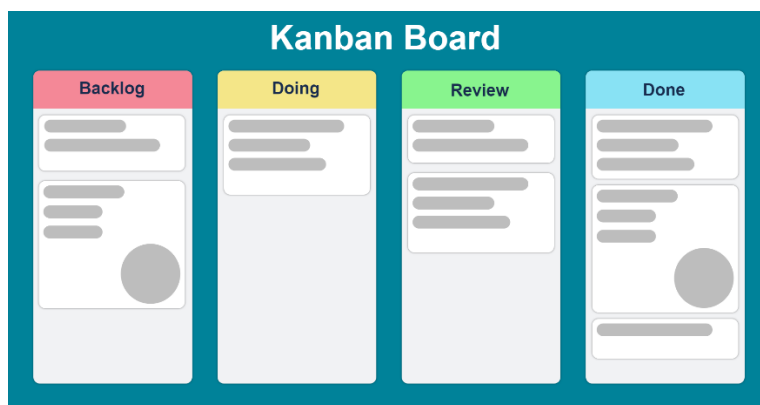
Ilustración 4. Metodología Scrum



Fuente: Wikimedia Commons

- **Kanban:** Consiste en segmentar o dividir las tareas a su mínima expresión para organizarlas en un tablero dividido en tareas pendientes, en desarrollo y terminadas. Esta forma de trabajar permite de una manera muy visual diferenciar tareas prioritarias respecto a las que no lo son, y observar el incremento en el valor del producto a medida que se finalizan las tareas.

Ilustración 5. Tablero Kanban

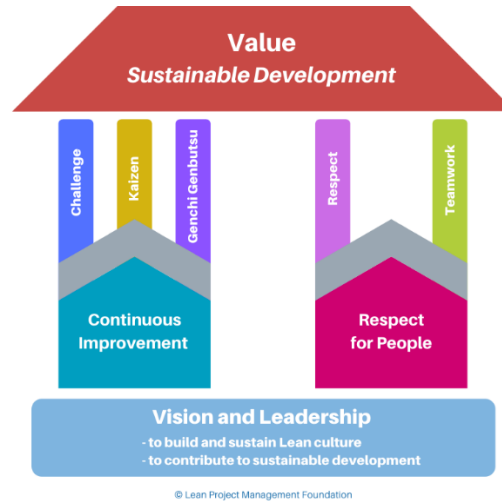


Fuente: Wikimedia Commons

- **Lean:** Metodología pensada para pequeños equipos de trabajo pero con mucha capacidad, que sean capaces de realizar las tareas de manera muy rápida. Busca minimizar tiempos de entrega y retrasos

durante el desarrollo pero siempre fomentando el aprendizaje, la capacidad de reacción y el potenciamiento del trabajo en equipo.

Ilustración 6. Bases metodología Lean

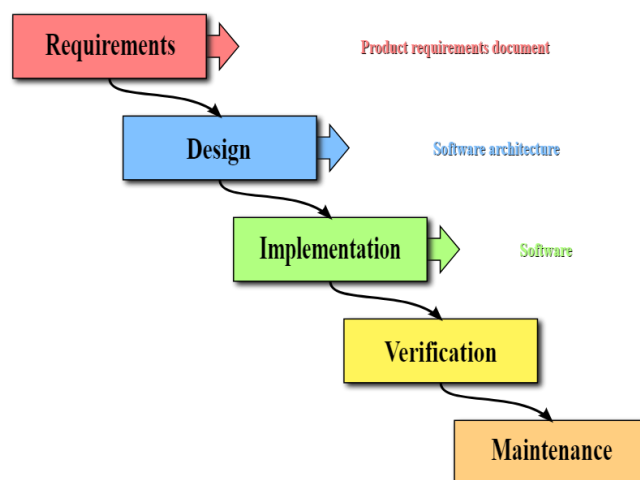


Fuente: Wikimedia Commons

- **Metodologías tradicionales**

- *Waterfall* (En cascada): Es una de las más tradicionales, en esta el proyecto se divide en distintas etapas, donde hasta que no se finaliza una de manera definitiva no se pasa a la siguiente, se caracteriza por ser por lo tanto secuencial. Normalmente, no se puede proporcionar una muestra funcional hasta la parte final del proceso ya que no se disponen de un resultado completo hasta el final.

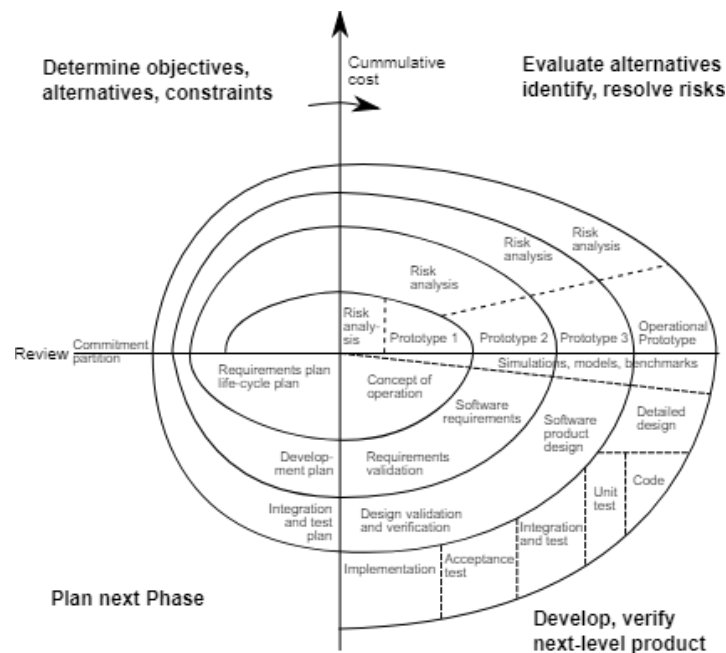
Ilustración 7. Metodología en cascada



Fuente: Wikimedia Commons

- *Spiral process model* (Modelo en espiral): Esta metodología está muy dirigida para usarse en proyectos de desarrollo de software cuya importancia relativa a la incertidumbre y riesgo del proyecto sea muy elevado. Por ello, tiene un enfoque iterativo para que entre iteraciones se puedan dar cambios muy rápidos, de manera flexible que permita pivotar al proyecto de la mejor forma posible.

Ilustración 8. Metodología en Espiral



Fuente: Wikimedia Commons

Estas son solo algunas de la gran cantidad de metodologías de desarrollo software que existen.

Una vez se han introducido las metodologías, pasando concretamente a la que se aplicó en este proyecto, se tomó la decisión de usar la metodología de desarrollo tradicional “**En cascada**” o *Waterfall*, debido a que se deseaba de una manera secuencial ir ejecutando cada una de las fases sin dar cambios repentinos y tener que deshacer cosas en fases anteriores. Además, dado que es probable que se usen múltiples herramientas de desarrollo que nunca se han usado, y por tanto, no se tiene experiencia previa en ellas, es mejor centrarse en el desarrollo de una manera secuencial y sin imponerse una metodología ágil que puede requerir una mayor base de conocimientos previos para poder alcanzar los objetivos de iteraciones con unos tiempos mucho más inferiores.

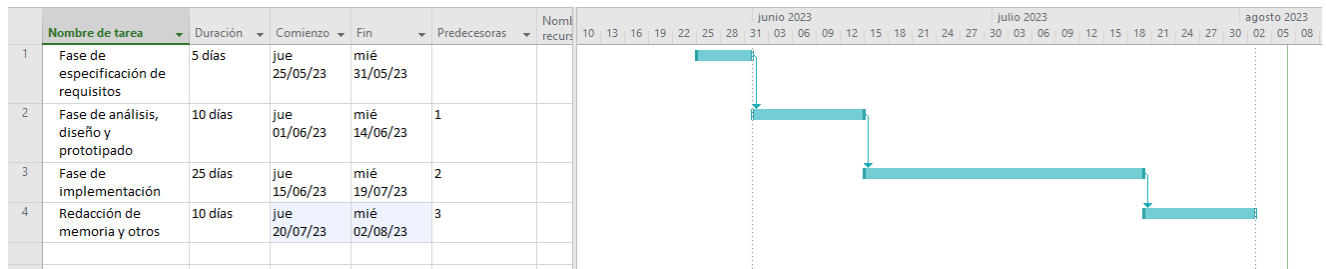
Planificación

Respecto a la planificación del proyecto y al desarrollo de sus distintas fases, pese a comenzar a ver algunos aspectos con anterioridad, de manera aproximada, se comenzó

verdaderamente a trabajar de forma constante entre finales de Mayo y principios de Junio, momento en el que se terminaron todas las asignaturas y trabajos del máster. Teniendo esto en cuenta, se hizo una distribución del tiempo restante hasta finales de Julio y principios de Agosto.

Para hacer esa distribución temporal se ha realizado el siguiente diagrama de Gantt usando Microsoft Project:

Gráfico 1. Diagrama de Gantt



Fuente: Elaboración propia

De este modo, se ha establecido que el mayor peso en cuanto a tiempo dedicado del proyecto lo tendría la fase de la implementación a causa del propio aprendizaje que se haga mientras se desarrolla el proyecto y de los posibles problemas o inconvenientes que puedan ir surgiendo y tengan que ser solventados.

El resto de las fases tienen unas duraciones bastante similares entre sí, y dependiendo de la necesidad que se tenga, se podrá ampliar el tiempo de alguna de estas siempre y cuando sea necesario en detrimento de otras que luego no lo requieran tanto.

Capítulo 2: Fase de especificación de requisitos

Introducción

En este segundo capítulo, se va a explicar el proceso de elicitación que se realizó para la obtención y recolección necesaria de información a tener antes de establecer los requisitos del sistema y que se procediera a realizar ningún análisis o comenzar a implementar el sistema.

Cabe destacar que la estructura utilizada para definir los requisitos se ha basado en el estándar IEEE/ANSI 830/1998, aunque la estructura ha sido adaptada a este proyecto en concreto y se ha resumido en los siguientes apartados.

Propósito

Tal y como se ha introducido, la finalidad de la especificación de requisitos es procesar la información obtenida durante el proceso de elicitación. Este proceso es vital porque antes de realizar los análisis o implementar el sistema, el desarrollador debe conocer de manera formal cuales son las necesidades que espera que sean satisfechas y sus expectativas relativas al producto software que se le proporcionará.

Elicitación

Entrando ya a la elicitación realizada en este proyecto, cabe indicar una serie de particularidades, este no ha sido encargado por ningún cliente específicamente, es decir, se ha detectado un problema y se pretende generar un producto que lo solvante. Por lo tanto, podríamos decir que de algún modo, el contratante se podría considerar al tutor de este TFM, que fue quién identificó el problema, y me lo transmitió, así como el producto a desarrollar recibiría el nombre de Sendseg. Cabe indicar que este contactó con una sociedad de cazadores con la finalidad de consultar sus procesos y ver si de verdad la resolución al problema les generaba interés y aportaría valor. Además, se consultó si empleaban alguna herramienta que les ayudará actualmente, pero tal y como nos podíamos imaginar, no era el caso.

Fuente de requisitos

La fuente de requisitos hace referencia a qué entidades, particulares o colectivos son los que transmiten la información a través del proceso de elicitación. En este caso, tal y como se ha descrito anteriormente, la principal fuente de requisitos fue el propio tutor del TFM.

Stakeholders

Conocida la fuente de requisitos, se deben definir los *stakeholders* o grupos de interés. Estos son también entidades, particulares o colectivos que se verán afectados positivamente ya que tienen unos intereses específicos por el producto que se va a desarrollar. Normalmente, los

grupos de interés se suelen diferenciar en grupos internos o externos según si los usuarios del producto pertenecen al colectivo que lo encarga, o si no, respectivamente.

Por lo tanto, en este caso, como no hay ninguna entidad como tal contratante, sino que todos los posibles grupos de interés serían externos tanto para mí como para el tutor del TFM, podemos hacer la siguiente tabla:

Tabla 1. Stakeholders

Nombre	Rol	Usuario directo	Interés
Sociedades de Cazadores	Genera actividades de caza	Sí	Facilidad de uso, eficiencia y eficacia para crear sus actividades y operar
Senderistas	Consulta actividades de caza	Sí	Facilidad de uso, eficiencia y eficacia para consultar actividades caza
Desarrolladores otras aplicaciones de rutas	Consume API	Sí	API mantenible, sin cambios y multitud de peticiones con distintos fines
Administraciones públicas	Administración	Sí	Facilidad de uso, eficacia y eficiencia en la administración del sistema

Fuente: Elaboración propia

Así como los dos primeros grupos son los que intervienen de manera directa en el problema a solucionar, cabe indicar la inclusión de un tercer grupo que serían todas aquellas empresas o desarrolladoras de aplicaciones que se encargan de proporcionar rutas a senderistas para que se den a conocer. Este último grupo, estaría interesado en que en futuros desarrollos del sistema puedan consumir los servicios o datos proporcionados por la API que ofrecería Sendseg para poder indicar, por ejemplo, cuando es una ruta de las que ofrecen segura y cuando no al tener en cuenta las actividades de caza creadas con sus respectivas áreas, haciendo uso de nuestro sistema.

Por último, podría haber un cuarto grupo de interés que serían las administraciones u otras entidades de carácter público las que podrían usar el sistema con labores de administración para controlar que las sociedades utilizan adecuadamente la herramienta, e incluso en un futuro para tener de manera más sencilla un control de qué actividades se han realizado y cuándo.

Alcance del producto

A la finalización de la primera versión del producto software a desarrollar se espera que las sociedades puedan crear sus actividades de caza y que los senderistas puedan consultarlas. Se procurará dentro del tiempo de desarrollo incluir el mayor número de funcionalidades posible que gire en torno a estas dos.

Funciones del producto

Las funciones del producto son todas aquellas necesidades que se esperan que los grupos de interés puedan satisfacer haciendo uso del producto a desarrollar. Estas funciones o necesidades para cubrir se pueden desglosar en dos grupos:

- Requisitos funcionales
- Requisitos no funcionales.

A continuación, se define cada tipo y se indicará que espera cada grupo de interés.

Requisitos funcionales

Los requisitos funcionales son aquellas acciones u operaciones que esperan los grupos de interés que deben poder realizar interactuando con el sistema. En este proyecto, dado que estamos ante la construcción inicial del sistema, se espera solo poder atender a satisfacer las necesidades de 3 de los 4 grupos de interés, dejando el desarrollo del consumo de la API para fases posteriores cuando de manera efectiva estén cubiertas las necesidades de los otros 3 grupos y se busque aportar valor de otra manera a otras aplicaciones.

Por lo tanto, los requisitos funcionales de este proyecto con respecto a cada uno de los 3 grupos de interés a satisfacer son:

Senderistas

Los requisitos funcionales asociados a los senderistas son:

- **Obtener información sobre el uso del sistema:** Se deberá de ofrecer información sobre cómo se usa Sendseg desde la perspectiva de un senderista para que pueda realizar consultas correctamente sobre las zonas de caza. Además esta información debe ser lo suficientemente descriptiva como para que después de hacer una consulta el resultado de esta se pueda interpretar sin ningún tipo de problema o dificultad.
- **Realizar la consulta sobre las actividades de caza:** Esta será la funcionalidad principal y la que caracterizará a un senderista, la consulta en la que dada la fecha de inicio de la actividad que desea realizar y la de finalización, se le proporcionará un mapa en el que estarán representadas las actividades de caza durante cualquier momento del desarrollo de la actividad para que pueda planificar de manera anticipada su excursión. Además, será imprescindible proporcionar un margen temporal en el que no se puedan ya establecer nuevas actividades de caza para que así tenga la certeza de que no se encuentre una actividad con poco margen de maniobra para su toma de decisiones.

Para acceder a las funcionalidades e información asociadas a este tipo de usuario no será necesario estar autenticado en el sistema, por lo que cualquier usuario, podrá acceder a toda esta información.

Sociedades de Cazadores

Los requisitos funcionales asociados a las sociedades de caza son:

- **Obtener información sobre el uso del sistema:** Al igual que los senderistas o cualquier usuario, se debe de informar a las sociedades sobre cómo se usan sus respectivas funcionalidades dentro de la aplicación. Estas descripciones también deberán ser clara para que no tenga ningún problema a la hora de desarrollar sus acciones de manera correcta.

- **Registro e identificación de sociedades:** El sistema debe permitir a estas el registro e inicio de sesión, esto tendrá como objetivo poder diferenciar usuarios autenticados como sociedades de los que no.
- **Creación de actividades de caza:** El sistema debe de proporcionar la posibilidad de crear actividades de caza a las sociedades que tengan credenciales y se identifiquen en el sistema. La creación de estas recogerá todos los datos que sean necesarios como la fecha de inicio y de fin de la actividad y no se permitirá el solapamiento de áreas con otras ya indicadas por medio de polígonos sean de la misma u otra sociedad.
- **Modificación de perfil de la sociedad:** Una sociedad identificada con sus credenciales en el sistema podrá modificar todos los datos que introdujo al registrarse para actualizarlos o por si ha tenido algún error al registrarse con excepción del email que introduzca.
- **Subida de informes de actividad:** Después de crear y llevar a cabo una actividad de caza, las sociedades, deben de realizar un archivo que sea un informe donde se recoja información como los participantes de la actividad, las armas usadas, los perros que se han llevado, o las presas que se han obtenido.
- **Descarga de informes de la actividad:** El informe que se permite subir asociado a una actividad realizada, podrá ser descargado posteriormente en caso de necesitar recuperarlo con algún fin.

En caso de que no se permita alguna acción a la sociedad para proteger la integridad del sistema o por otras motivaciones, habrán acciones que se le proporcionen al administrador para que con el contacto previo de la sociedad se realicen modificaciones o cambios en el sistema de aquellas acciones que ya se hayan realizado.

Administrador del sistema

Los requisitos funcionales asociados al usuario administrador del sistema que podría ser llevado a cabo por alguna administración pública son:

- **Acceder a información relativa a la base de datos y las tablas:** Se mostrará para cada clase a la que tiene acceso en la base de datos las operaciones que puede llevar a cabo.
- **Ver, crear, editar o eliminar sociedades:** El administrador podrá visualizar, dar de alta nuevas sociedades haciendo uso de un formulario equivalente al de registro, editar toda la información de las sociedades registradas incluyendo el correo y además podrá eliminar sociedades dadas de alta, en consecuencia, eliminando las actividades de caza que haya podido crear. Esto permitiría de algún modo evitar que algunos individuos o sociedades hagan un uso indebido del sistema.

- **Ver, editar o eliminar actividades de caza:** El administrador podrá ver las actividades de caza creadas por las sociedades, permitiéndosele editarlas o borrarlas.
- **Consultar las coordenadas de los polígonos de cada actividad:** El administrador podrá visualizar las coordenadas, formadas por la latitud y longitud, de cada uno de los polígonos asociados a una zona de caza.
- **Descarga o eliminación de informes asociados a las actividades:** El administrador podrá descargar los informes que suban las sociedades de cazadores después de realizar una actividad. Además podrá eliminar el informe en caso de que la sociedad lo solicite para subir una nueva versión o para simplemente subir el documento apropiado en caso de equivocación.

Requisitos no funcionales

Una vez han sido definidos los requisitos funcionales de cada uno de los tres grupos de interés, se deben determinar los requisitos no funcionales, que son definidos normalmente de manera íntegra por los desarrolladores del producto en función de las necesidades. Este tipo de requisitos son tan importantes como los funcionales debido a que alcanzar únicamente estos, no garantiza que se satisfagan todas las necesidades.

Los requisitos no funcionales no son fáciles de gestionar y vienen en función de las restricciones que se imponen al sistema. Además, estos requisitos normalmente para definirlos se deben de cuantificar de algún modo, lo que complica bastante su formalización. Por tanto los requisitos no funcionales son:

- Los servicios que proporcione el producto deberán de ofrecerse de manera ininterrumpida.
- La velocidad de respuesta del sistema debe de ser rápida y fluida para los distintos usuarios. Deberá de procurarse definir restricciones adecuadas para el procesamiento de las peticiones que se realicen, de manera en que la carga de elementos en los mapas sea hasta cierto punto limitado.
- Deberá de garantizarse la integridad de los datos que se almacenen en la base de datos que contiene toda la información relativa a usuarios, actividades de caza o informes.
- El sistema deberá de ser escalable y estable, para poder resolver todas las peticiones que los usuarios realicen al servidor.
- En cuanto a la velocidad de aprendizaje del uso de la aplicación debe de ser muy rápida, de manera en que solo leyendo la información que se muestre a cada tipo de usuario puedan llevar a cabo todas sus acciones.
- El sistema deberá de intentar ser lo más explícito posible, en el sentido de que cada acción realizada por el usuario deberá de obtener una respuesta ya sea correcta o incorrecta en caso de hacer alguna petición errónea.

- Se espera que un porcentaje de error muy bajo de todas las peticiones realizadas, menor del 5%, el resultado de esto iría muy asociado a la curva de aprendizaje en el uso del sistema.

Características de los usuarios

Tal y como ya se ha estado comentando anteriormente existirán distintos tipos de usuarios, cada uno de ellos se corresponde con uno de los grupos de interés. Ahora se va a dar una descripción formal uno por uno y se verá porqué se han proporcionado dichas funciones a cada uno de ellos.

En cuanto a los **senderistas**, la realidad es que este podría ser cualquier persona que desee realizar una actividad en una zona aislada que pueda ser de caza, es decir, de las funciones de este tipo de usuario, no solo se podrían aprovechar ellos, sino personas que realicen otras actividades como los ciclistas de montaña. El perfil de este tipo de usuario es aquel que quiere hacer alguna actividad y quiere velar por su propia seguridad de manera que sabiendo cuando inicia y cuando acaba esta, pueda consultar las zonas de caza para que de manera anticipada pueda modificar o llevarla a cabo con conocimiento de causa. Precisamente, para potenciar y favorecer el interés en la seguridad de este tipo de usuario se hará como se ha comentado anteriormente que no sean necesarios permisos en el sistema para poder simplificar el proceso de consulta.

Respecto a las **sociedades**, tendrán un gran interés en definir en la aplicación donde van a realizar una actividad de caza, no solo por informar a los senderistas, y evitar los posibles conflictos, sino que además, estarán reservando esa zona para que otras sociedades no puedan disponer de la misma en el mismo horario.

Por último, los **administradores** podrán llevar un control en el uso del sistema por parte de las sociedades y solucionar cualquier problema o fallo que pueda tener una sociedad a la hora de principalmente gestionar su propia cuenta o actividades.

Evidentemente, la mayor diferencia entre los 3 tipos de usuarios radicará en si son necesarias credenciales o no, y en caso de necesitarlas, si estas son de sociedad o de Administración. A medida que mayor grado de credenciales tenga el usuario podrá acceder a operaciones que sean más delicadas en el sistema y que impliquen más cambios en los registros de la base de datos que contenga toda la información que se va generando y almacenando.

Por tanto, la jerarquía de credenciales, de menor cantidad de permisos a mayor en el sistema será:

- Los senderistas se corresponderían con los usuarios anónimos que no requieren de ninguna autenticación.
- Las sociedades de cazadores necesitarán de autenticación para poder acceder a información y funcionalidades que puedan cambiar el estado del sistema. Pese a esto, solo podrán acceder y modificar actividades de caza o datos que le pertenezcan, imposibilitándole acceder a información de otros usuarios.

- Los administradores del sistema necesitarán credenciales y dispondrán de la mayor cantidad de permisos. Estos podrán acceder a información y funcionalidades generadas por las sociedades de cazadores con la finalidad de consultarla o modificar el estado del sistema.

Restricciones

El sistema deberá de tener en cuenta algunas de las siguientes restricciones:

- Se debe de garantizar que cada tipo de usuario solo tenga acceso a las peticiones que corresponden con las acciones que puede realizar y solicitar al servidor.
- La información visualizada a través de las distintas ventanas solo puede ser accedida por el tipo de usuario que corresponda.
- La base de datos que se utilizará para la implementación será relacional para que, por un lado, pueda ser reutilizada, y por el otro, sea más sencilla la comprensión de esta para futuras ampliaciones del proyecto. A nivel de la base de datos solo se implementarán aquellas tablas necesarias para el correcto funcionamiento de la aplicación, de modo que en futuras ampliaciones se pueda minimizar la dificultad de comprensión de esta.
- Por cada petición que haga una sociedad para crear una nueva zona de caza solo se permitirá la definición de un único polígono que represente el área.

Suposiciones y dependencias

La única condición para que cualquier tipo de usuario pueda acceder al sistema será el tener un dispositivo con conectividad a internet para poder navegar e ir realizando los distintos tipos de peticiones asociadas a las distintas acciones que puede hacer cada usuario según sus credenciales. En caso de no disponer de una conexión de calidad la única implicación a priori sería que podría haber una pérdida en la fluidez entre las peticiones y respuestas entre servidor y dispositivo.

Además del dispositivo y de la conexión a internet será necesario el uso de un navegador, en concreto se recomendaría el uso de Google Chrome, ya que en este proyecto el desarrollo será para una aplicación web y en el futuro se podría implementar una aplicación puramente móvil.

Respecto al propio dispositivo usado en la navegación, se empleará algún framework que permita la responsividad o *responsiveness* de la aplicación, es decir, la interfaz se adaptará a los distintos tamaños o resoluciones de pantalla que se puedan usar.

Por lo tanto, se puede entender que las limitaciones para poder acceder al sistema por parte de los distintos tipos de usuarios serían prácticamente nulas a día de hoy. El único factor a tener en cuenta será que donde se aloje la aplicación se tenga la potencia suficiente como para atender a todas las peticiones que se puedan realizar a lo largo del tiempo y en los picos de peticiones que pueda haber.

Requisitos futuros

Tal y como se ha comentado anteriormente, este proyecto supone la resolución a un problema concreto y la primera fase de desarrollo de una aplicación que con el paso del tiempo, generación de nuevos proyectos, e incrementos, podrían hacer crecer sustancialmente el sistema, por lo que a medida que pase el tiempo se irán identificando y definiendo nuevos requisitos para cada tipo de usuario y se irá dando solución a estos nuevos, tanto funcionales como no funcionales que vayan apareciendo.

Sí que cabe indicar que considero que se deberá de intentar priorizar, mejorar y profundizar la parte de consulta de los senderistas o, evidentemente, proporcionar satisfacción a las necesidades del cuarto tipo de grupo de interés que consumirá la API proporcionada por Sendseg para ser utilizada por otras aplicaciones que puedan tener en cuenta las actividades de caza que se hayan definido en este sistema.

Capítulo 3: Análisis, Diseño y Prototipado

En este tercer capítulo tras formalizar los requisitos, se van a realizar todos aquellos pasos previos a la implementación que son:

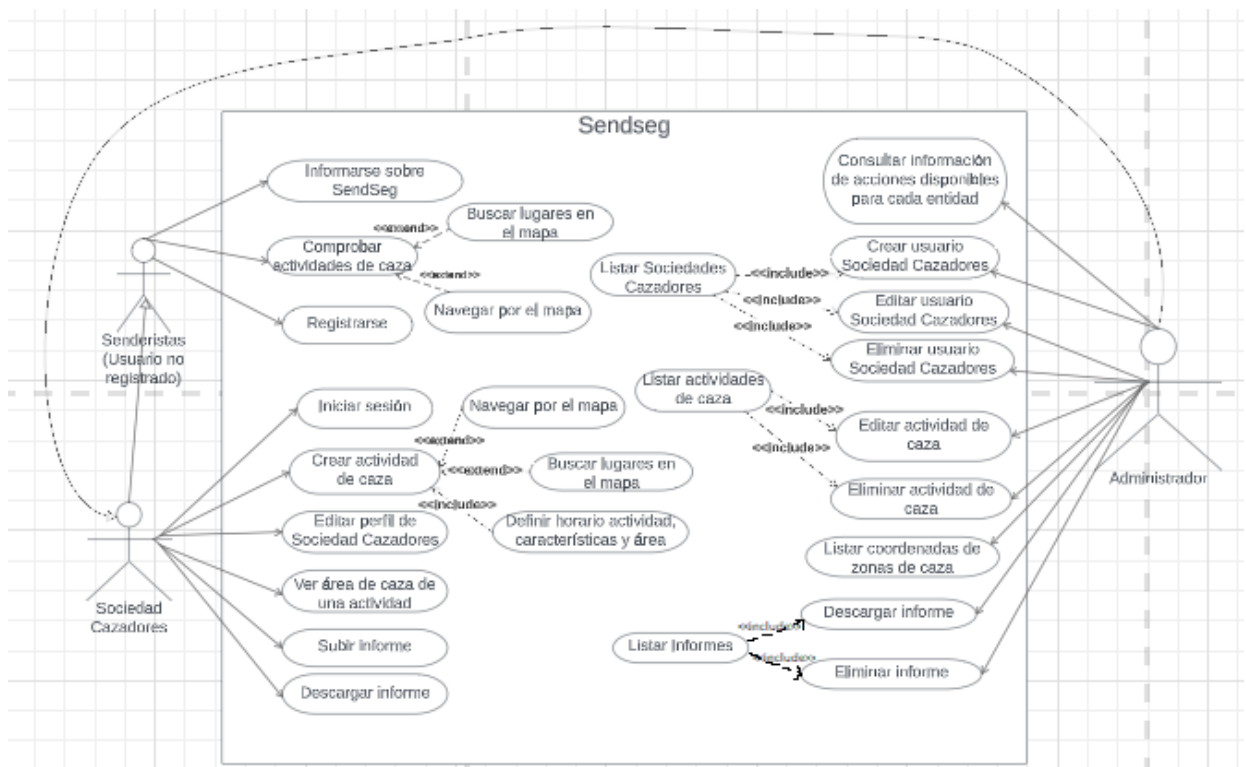
- Análisis
- Diseño
- Prototipado

Esta fase es muy importante en el devenir del proyecto y en el resultado final debido a que cada una de las decisiones que se tomen en función de los requisitos y siguientes análisis que se hagan, lastrarán y afectarán de manera muy directa sobre la aplicación o sistema final que se obtendrá.

Diagrama de casos de uso

El primer análisis que se ha realizado ha sido el diagrama de casos de uso de Sendseg, el cual es el siguiente:

Ilustración 9. Diagrama de casos de uso general



Fuente: Elaboración propia

En este diagrama podemos observar cómo los actores que intervienen se corresponden con cada uno de los tipos de usuarios que representaría a cada uno de los tres grupos de interés que se desean satisfacer: senderistas, sociedades de cazadores y administradores.

De este diagrama cabe destacar el uso de la especialización indicada entre los 3 usuarios, esta se ha usado para indicar que un senderista solo puede hacer sus casos de uso, mientras que una sociedad de cazadores puede hacer los suyos y además los de los senderistas. En cuanto al último, el administrador, teóricamente podría efectuar todos los casos de uso, aunque cabe indicar que en principio no podrá crear actividades de caza, esto no se ha reflejado en el diagrama por la gran cantidad de líneas que habría que poner, por eso, se hace explícito en esta explicación.

A continuación, se va a mostrar en una tabla por cada caso de uso, como se podría ejecutar cada uno de ellos, es decir, se van a comentar las precondiciones para poderlo ejecutar, las postcondiciones en el sistema una vez se haya realizado y las interacciones que habrá entre el sistema y el usuario. Debido a que cada caso genera una tabla se van a dejar dos como ejemplo en esta lectura, aunque se pueden encontrar el resto en el [Anexo 1](#).

Tabla 2. Caso uso: Informarse sobre Sendseg

Caso de uso	Informarse sobre Sendseg
Precondición	-
Postcondición	-
Proceso	
Usuario	Sistema
El usuario navega por las ventanas de Inicio o sociedades cazadores y puede leer información y ver otros recursos gráficos sobre qué puede hacer cada usuario en Sendseg.	
	El sistema mostrará las ventanas de Inicio o Sociedades de cazadores con sus recursos ya sean textos, imágenes o videos según se le solicite

Fuente: Elaboración propia

Tabla 3. Caso uso: Comprobar actividades de caza

Caso de uso	Comprobar actividades de caza
Precondición	-
Postcondición	-
Proceso	
Usuario	Sistema
El usuario pulsa en la barra de navegación "Senderistas"	
	El sistema muestra la ventana con el formulario en el que se deben introducir las fechas de inicio de la actividad y final
El usuario rellena los dos datos	
	El sistema comprueba y valida los datos introducidos y carga el mapa correspondiente

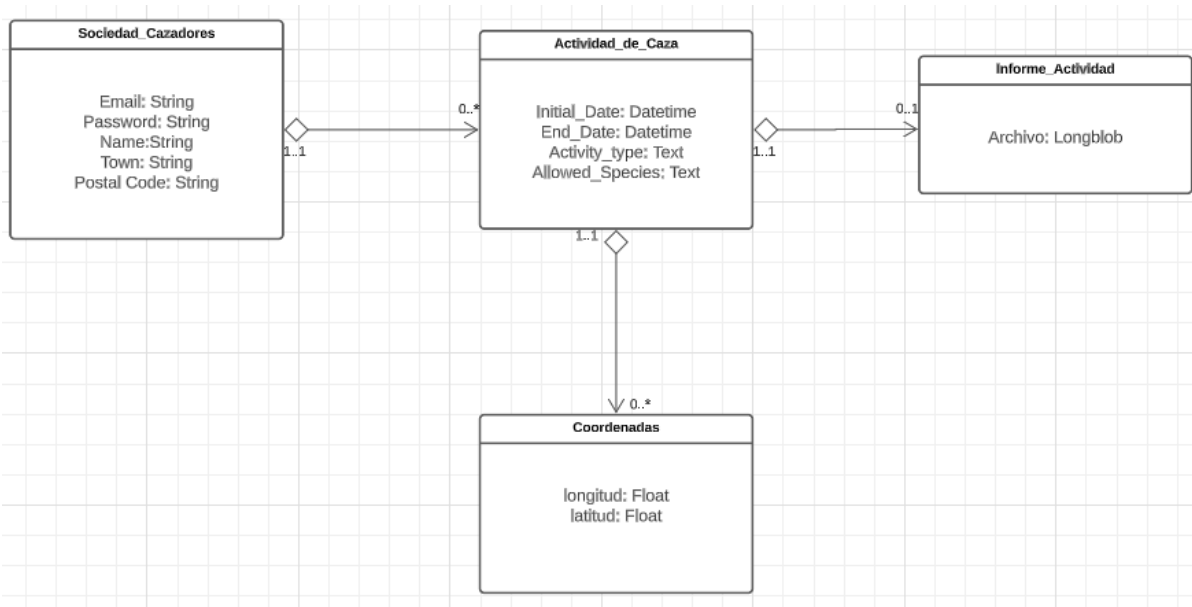
en el que se ven los polígonos que representan las zonas de caza que estarán activas en algún momento del Intervalo de tiempo introducido. En caso contrario muestra el error oportuno.

Fuente: Elaboración propia

Diagrama de clases

Una vez explicado el diagrama de casos de uso, se deben identificar los elementos, entidades o clases que serán necesarios utilizar en el sistema para que este se comporte del modo deseado. Este comportamiento, no solo viene determinado por las clases que lo definen, sino también por las relaciones y el modo en el que interactúan o se relacionan entre estas. Para identificar y formalizar esto, usamos el diagrama de clases, el cual en el caso de Sendseg es el siguiente, y que constará de 4 clases:

Ilustración 10. Diagrama de clases



Fuente: Elaboración propia

La primera clase será la de **Sociedades_Cazadores**, la cual permitirá representar a los usuarios que se correspondan con sociedades que hagan uso del sistema. Cada una de ellas, encapsulará una serie de características o atributos que la definan y diferencien del resto de su mismo tipo. Por último, cabe destacar que solo se relaciona con una clase que serán las **Actividad_de_Caza**, de forma en que una sociedad puede tener ninguna o muchas actividades.

La segunda clase es la **Actividad_de_Caza**, la cual con sus características permitirán definir y diferenciar las distintas actividades que se creen por las **Sociedades_Cazadores**. Esta podríamos considerarla como la columna vertebral o central del sistema ya que se relaciona con cada una del resto de las clases. Respecto a su relación con las **Sociedades_Cazadores** es muy

importante señalar que una actividad de caza deberá pertenecer o estar directamente relacionada con una única sociedad desde el mismo momento de su creación. Por otro lado, una Actividad_de_Caza estará formada o constituida con 0 o más coordenadas y puede o no tener asociado un informe.

La tercera clase es la de **Coordenadas**, que permitirá representar de cada actividad de caza a todos y cada uno de los vértices del polígono que se corresponde con el área afectada, siendo cada uno definido por una latitud y longitud concreta. Esta clase solo se relaciona con la Actividad_de_Caza y nos indica que toda coordenada o vértice debe de ir asociado o pertenecer a una actividad de caza.

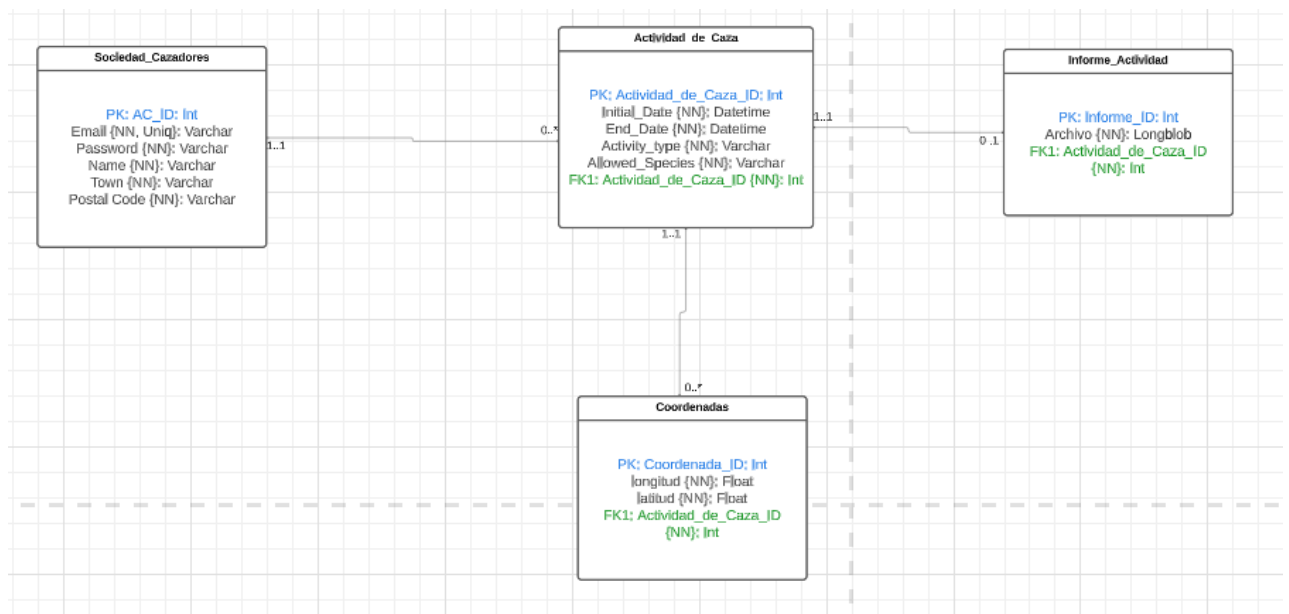
La última clase es la de **Informe_actividad**, que representará a todos los informes que se asociarán con Actividad_de_Caza. Básicamente, contendrán un archivo que será un documento que se realizará después del desarrollo de la actividad. En caso de crear un informe siempre, deberá ir asociado a una, es decir, existe la limitación de que no se podrán crear informes que estén aislados o no asociados a ninguna actividad.

Diagrama relacional

Una vez se ha realizado el diagrama de clases, se ha planteado el diagrama relacional asociado, ya que como se ha comentado con anterioridad la base de datos que almacene la información para garantizar la persistencia de los datos y la posibilidad de consultarlos a lo largo del tiempo, será de tipo relacional o estructurada en tablas.

Por tanto, el diagrama relacional es:

Ilustración 11. Diagrama relacional



Fuente: Elaboración propia

Del diagrama relacional cabe indicar que cada tabla tendrá un identificador que no puede ser nulo y será único marcado de color azul. Luego cada tabla tendrá una serie de atributos que al ser marcados con “{NN}” no podrán ser nulos, y lo marcado en color verde en las tablas de Coordenadas, Actividad_de_Caza e Informe_Actividad serán claves foráneas o ajenas que

permitirán cumplir con la propia definición de cada una de las relaciones. Un ejemplo sería que cuando se crea una Actividad_de_Caza, esta tiene que pertenecer o haber sido definida por una Sociedad de manera obligatoria, y este mismo comportamiento se replica entre las coordenadas y la Actividad_de_Caza, así como entre, el informe_ actividad y la Actividad_de_Caza.

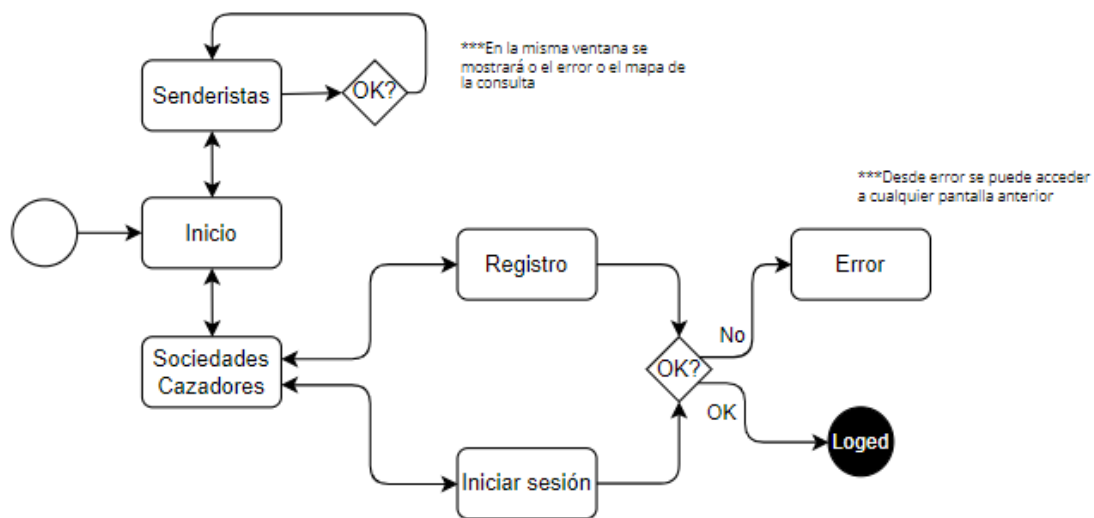
Diagrama de flujo de navegación

Ahora pasamos al diagrama de flujo de navegación, el cual permitirá estructurar y ordenar como cada tipo de usuario, con su correspondiente nivel de permisos, puede navegar y desplazarse entre las distintas interfaces, donde cada una, permite realizar una serie de acciones y posibilita la consulta o el acceso a cierta información.

Diagrama de flujo de navegación de usuario senderista

El diagrama de flujo de navegación asociado a un usuario de tipo senderista, sin credenciales, en el sistema es el siguiente:

Ilustración 12. Diagrama de flujo de navegación de Senderista



Fuente: Elaboración propia

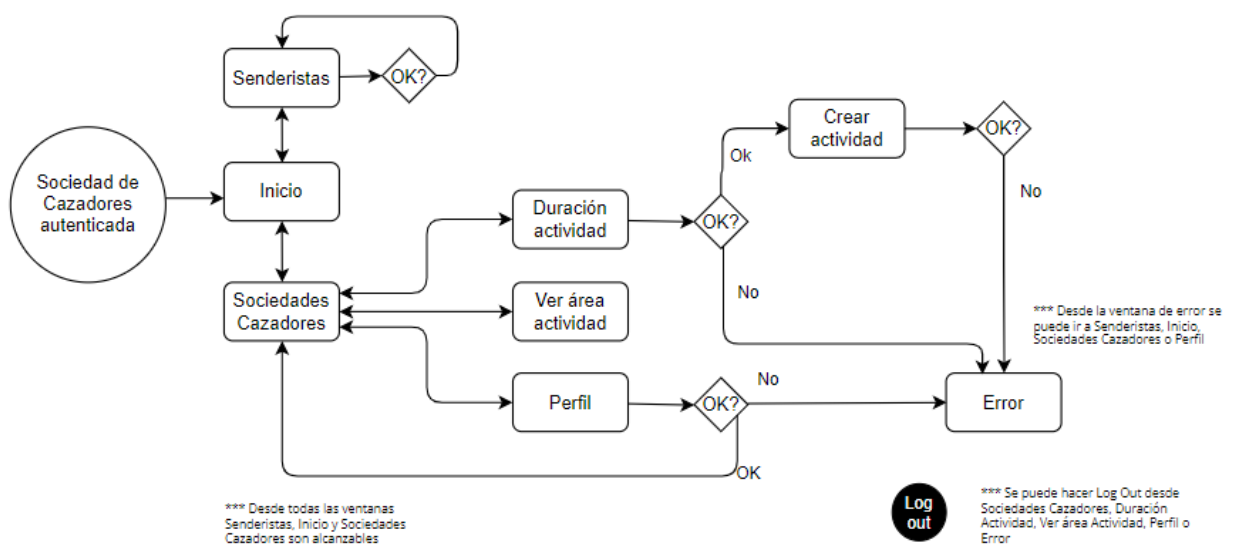
En este cabe indicar que la ventana de senderistas que contendrá la petición relativa a la consulta de las actividades de caza renderizará la respuesta, ya sea de error o el mapa correspondiente en la misma ventana, de forma en que no será necesario realizar ninguna transición de ventana. Por otro lado, en caso de error en el inicio de sesión o registro, se mostrará una ventana de error, la cual permitirá navegar a cualquiera de las otras ventanas que tiene disponible este tipo de usuario. Finalmente, cuando un usuario senderista se registre como sociedad o inicie sesión como cualquier otro tipo de usuario, finalizará todas sus posibles transiciones o flujos de navegación, representado con el círculo de color negro, ya que pasará a

tener otros permisos en el sistema y será ya identificado con credenciales o bien como Sociedad de Cazadores o bien como Administrador.

Diagrama de flujo de navegación de usuario Sociedad de Cazadores

El siguiente diagrama representa el mismo concepto anterior pero con respecto al usuario que fue senderista y ahora es identificado por el sistema como un usuario de tipo Sociedad:

Ilustración 13. Diagrama de flujo de navegación de Sociedad de Cazadores autenticada



Fuente: Elaboración propia

De este diagrama cabe indicar que independientemente de la ventana en la que esté el usuario sociedad de cazadores autenticado puede acceder a las tres ventanas o interfaces de Senderistas, Sociedad de Cazadores e Inicio.

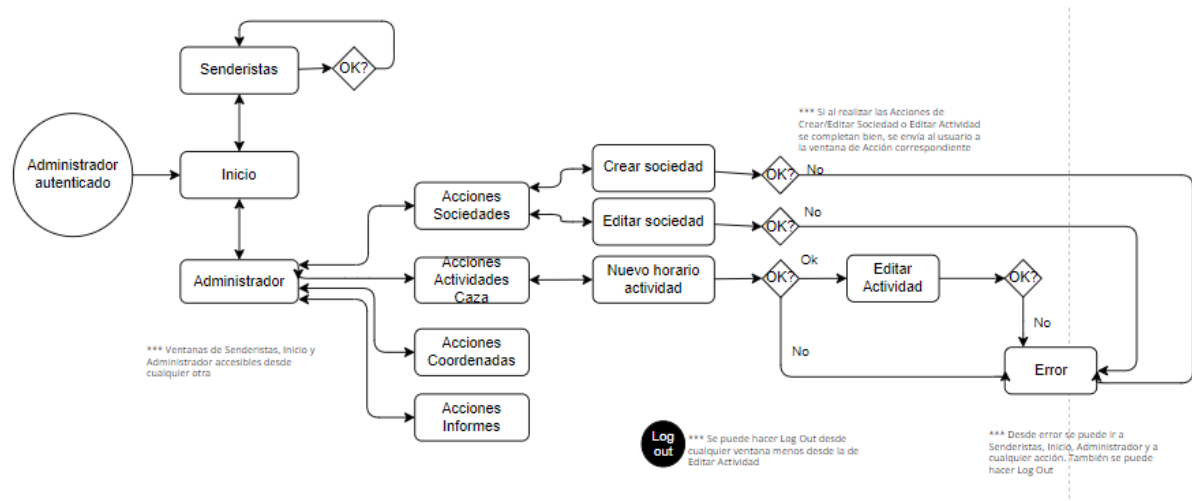
Aquí podemos observar una mayor cantidad de pantallas respecto al usuario senderista debido al incremento en la cantidad de acciones que puede realizar. De todas ellas, la operación más relevante de este usuario será la creación de las actividades de caza la cual podemos observar que estará dividida en dos pasos, el primero será definir el inicio y fin de la actividad para en función de lo que se introduzca ir a una ventana de error o, pasar al segundo paso que es proceder a la creación en sí creando el polígono y cumplimentar el resto de los datos. Adicionalmente, podrá acceder a una interfaz que le permita modificar sus datos de usuario.

Finalmente, el círculo de color negro se referirá a la acción de cerrar sesión como usuario identificado, lo cual le devolverá a un nivel de permisos equivalente al de un usuario senderista sin autenticar.

Diagrama de flujo de navegación de usuario Administrador

El diagrama de flujo de navegación del usuario administrador, que es aquel que cuenta con mayor nivel de permisos, acciones disponibles, e impacto en el sistema y sus datos es:

Ilustración 14. Diagrama de flujo de navegación de Administrador.



Fuente: Elaboración propia

El administrador tiene en común el alcance y las ventanas de Senderistas e Inicio con el tipo de usuario sociedad de cazadores, lo que sucede es que aparece una ventana que sustituye a “Sociedades Cazadores” donde aparecerá Administrador. En esta ventana, tendrá una descripción de para cada entidad o clase, que acciones puede realizar a modo de información y tendrá la opción de ir a visualizar las distintas instancias de cada entidad. También podrá dependiendo de la tabla que visita, realizar más acciones como con las Sociedades o menos como las Coordinadas.

En este diagrama, al haber una mayor cantidad de ventanas, se ha intentado definir en texto junto al cómo se accede al cierre de sesión o *log out*, así como a donde se puede ir desde la ventana de error. Por último, al igual que el usuario de tipo sociedad de cazadores, cuando cierre sesión perderá sus credenciales y permisos y se convertirá en un usuario de tipo senderista.

Prototipado

Una vez se ha definido el posible flujo de navegación para cada tipo de usuario es momento de hacer un prototipado de cada una de las interfaces que representarán cada una de las ventanas que se han visto anteriormente.

Para realizar esta tarea existen múltiples herramientas de diseño y prototipado de interfaces como Figma, aunque por su facilidad de uso, y debido a que se utilizó previamente en una asignatura del máster hemos decidido usar Balsamiq.

Antes de mostrar los prototipos realizados se debe de comentar el hecho de que estas interfaces serán una referencia a la hora de implementar las ventanas pero durante la propia implementación se puede ver sujeta a ligeras modificaciones según aparezcan problemas o se puedan incorporar posibles mejoras.

A causa de la gran cantidad de ilustraciones que se va a mostrar, para facilitar la lectura de la memoria, solo se van a incluir en este apartado 2 ejemplos aunque en el [Anexo 2](#), se pueden encontrar el resto de los prototipos de ventanas.

Ilustración 15 Prototipo ventana Inicio



Fuente: Elaboración propia

Ilustración 16. Prototipo ventana consulta zonas caza Senderistas

https://SENDESEG.com

SENDSEG Senderistas Sociedades Cazadores

Consulte las zonas de caza

Fecha de Inicio

dd/mm/aaaa----

Fecha de Fin

dd/mm/aaaa----

Consultar Cancelar

SENDSEG

Senderistas Sociedades Cazadores

Redes Sociales

Instagram Facebook

Sendseg by Miguel Ballester Nicolás
+34 XXX XXX XXX

Fuente: Elaboración propia

Por tanto, el objetivo a nivel estético, de elementos necesarios visibles en las interfaces y de cómo estos estarán dispuestos en ellas, ha quedado formalizado tras la elaboración del prototipo.

Selección de API para Mapas y Polígonos

En primer lugar se debe de entender que en la aplicación que se va a desarrollar en este proyecto, existe la necesidad de utilizar servicios de mapeados externos, no solo para obtener los mapas, sino además para realizar operaciones sobre ellos, principalmente para la definición de polígonos constituidos por una serie de vértices que son definidos por un valor de latitud y longitud.

La forma de acceder a estos servicios de mapeados externos será a través de consumir una API, cuyas siglas significan *Application Programming Interface*, que permite que distintas aplicaciones de software se comuniquen entre sí utilizando protocolos para permitir la solicitar y compartir información.

Por lo tanto, en este último apartado dentro de la fase de análisis, diseño y prototipado, se va a tomar la decisión de qué API o que proveedor de los servicios de mapeado será la que se consuma o utilice en Sendseg. Para ello, se ha realizado una tabla comparativa entre las dos opciones principales a elegir, OpenStreetMap o Google Maps:

Tabla 4. Comparación API Google Maps y OpenStreetMap

Google Maps	OpenStreetMap
Gratuita en una cuenta nueva, pero después de pago	Abierta, no requiere pagos
Actualizaciones del mapa constantes	Menor ritmo de actualizaciones del mapa
Posibilidad de usar otras librerías o bibliotecas para consumir otros servicios e integrarlos fácilmente	Menor cantidad de complementos o bibliotecas
Menor curva de aprendizaje para usuarios	Mayor curva de aprendizaje para usuarios
Ambas proporcionan la posibilidad de crear polígonos	

Fuente: Elaboración propia

En esta tabla podemos identificar cuáles son las principales diferencias entre una opción y la otra. A continuación voy a justificar porque me he decidido por el uso de Google Maps en detrimento de OpenStreetMap.

Pese a que Google Maps tenga el inconveniente de que es de pago con respecto a OpenStreetMap, esta primera proporciona la posibilidad de tener una cuenta durante un periodo de tiempo y siempre que se haga una cantidad inferior de peticiones a un límite mensual no se procederá a ningún cobro, permitiéndome realizar un desarrollo sin costes. Por otro lado, tratándose Sendseg de dibujar polígonos en zonas alejadas o remotas, cabe la posibilidad de que se actualicen cada más tiempo o ni se actualicen los mapas o datos en OpenStreetMap. Sin embargo, Google sería una garantía en este aspecto. Todo esto, sumado a que cualquier usuario reconoce Google Maps y sus opciones en comparación a OpenStreetMap, así como que Google ofrece otras librerías o servicios que se podrían usar e integrar más fácilmente con Sendseg no solo ahora sino en un futuro, solo ha añadido más valor y reforzado esta decisión de usar Google Maps API.

Capítulo 4: Implementación

Una vez realizados los análisis previos y tomadas las decisiones de diseño y prototipado, pasamos al cuarto capítulo, donde se comentarán distintas tecnologías o elementos que se han utilizado en la fase de implementación y se explicará que objetivo o para qué el uso de cada una de ellas.

Visual Studio Code

El entorno de desarrollo es aquella herramienta que permite la creación de proyectos, estructuración, edición y gestión de los archivos, así como, instalación de librerías, paquetes o dependencias para el desarrollo de un producto software.

Respecto a este, el seleccionado para el desarrollo del proyecto debido a factores como la gran versatilidad, compatibles con gran cantidad de lenguajes de programación, facilidad para estructurar proyectos, posibilidad de personalización debido a la instalación tanto extensiones como módulos para ayudar y hacer el desarrollo de software de la forma más ligera posible se decidió usar **Visual Studio Code**, o VS Code.

Además de todas las ventajas anteriores, existen otras ventajas como que tiene integrado el control de versiones que ofrece GitHub lo cual facilita su uso desde un momento inicial o que tiene una gran cantidad de usuarios que forman una comunidad con la que en caso de que surja cualquier tipo de problema es bastante posible que ya le haya sucedido a otro usuario y exista una solución documentada que poder seguir.

Node JS

Introducido el entorno de desarrollo del proyecto, para desarrollar Sendseg se decidió hacer uso de Node JS como entorno de ejecución Javascript por varios factores:

- Posibilidad de usar código de JavaScript que se ejecute en el lado del servidor.
- Arquitectura basada en el motor V8 de Google Chrome, buscador más usado por los usuarios y compatible con multitud de dispositivos heterogéneos.
- Modelo de entrada/salida no bloqueante y asíncrono, ideal para aplicaciones en tiempo real y con alta concurrencia.
- Amplia escalabilidad y rendimiento.
- Gran oferta de librerías, frameworks o módulos que facilitan y aceleran el desarrollo (NPM).

De todos los anteriores factores se debe destacar respecto al desarrollo de Sendseg que posibilitó el escribir en lenguaje JavaScript tanto el código que se ejecuta del lado del servidor como del cliente. Además, la cantidad de paquetes y módulos que existen es enorme, y para ello se usó NPM (*Node Package Manager*) para la instalación y uso de cada uno de ellos. De hecho, en los siguientes apartados se comentarán los usados en la implementación de Sendseg y el objetivo de su utilización.

Para dejar constancia, la versión de Node JS que se ha usado en la implementación de Sendseg es la **v.18.16.0**.

Express

El primer módulo NPM instalado y usado en el proyecto es Express, este tiene una importancia vital ya que permite la creación y configuración del servidor así como de las peticiones que debe de manejar y resolver.

Los distintos tipos de peticiones que se utilizaron en el proyecto son las siguientes:

- **Get:** Peticiones al servidor que permiten la obtención de recursos y de vistas que los representan a medida que el usuario va navegando. En este tipo de peticiones se puede pasar información a través de la url aunque tiene una mayor limitación respecto al siguiente tipo de peticiones y tienen un menor nivel de seguridad respecto al paso de información.
- **Post:** Peticiones que permiten el transporte de datos introducidos en la cabecera de mensajes que se envían desde los formularios que introduce el usuario hacia el servidor para procesarlos y crear un nuevo recurso, esto generará la correspondiente respuesta al usuario con la carga de la vista que corresponda en caso de que los datos sean correctos o no.
- **Put:** Peticiones que al igual que post se realizan a través del envío de datos introducidos por el usuario en formularios y que el propósito que tienen es la modificación de alguno de los recursos que ya ha sido creado previamente con una petición post.
- **Delete:** Peticiones que permiten la eliminación de uno de los recursos que fueron creados con peticiones de tipo post.

Para las peticiones de tipo *put* y *post*, el envío de los datos de los formularios se puede hacer por el hecho de que express tiene integrado el *middleware* de “body-parser” que permite que los datos pasados a través de las cabeceras puedan ser usados y procesados en las peticiones correspondientes.

Cuando cada una de las peticiones del servidor que procesa puede generar dos resultados posibles, o que la petición es correcta y se lista, crea, modifica o elimina recursos y se redirige a la vista que corresponda en el sistema o que haya algún error que lo que haga sea mostrar al usuario el falló o motivo por el cual no se completó esta satisfactoriamente para que lo pueda subsanar y volver a intentar.

Por último, cabe indicar que todas y cada una de las peticiones que conforman la aplicación web de Sendseg, se diseñan e implementan basándonos y siguiendo los principios de REST (*Representational State Transfer*).

Express-session

Una vez comentado el módulo de express, se debe de explicar que se usó un módulo relacionado directamente que extiende este y se llama express-session.

Este módulo permite administrar sesiones en aplicaciones web, es decir, posibilita que se mantenga información de la sesión del usuario, lo que permite que las acciones o peticiones que lanza un usuario no sean aisladas y mantenga el estado y la autenticación del usuario durante la visita a la aplicación.

Básicamente, en Sendseg, desde el momento en el que un usuario inicia sesión, se crea una sesión de usuario donde se almacena en una variable de sesión el *email* correspondiente, a partir de esto, las respuestas del servidor en función de las peticiones se encargan de renderizar las vistas de la forma oportuna. Evidentemente, cuando el usuario cierra sesión la sesión esta queda destruida y no queda información ni variables de la sesión, solo quedará en el sistema aquella información persistente en la propia base de datos.

En futuras ampliaciones de Sendseg, el uso de este módulo se podrá acrecentar según se necesiten tener más valores almacenados durante la sesión que tenga abierta un usuario.

Bulma

Desde un primer momento, la intención era que Sendseg fuese *responsive* y tuviera una estética lo más bonita posible, para lograr esto, la idea era tratar de usar alguna herramienta o módulo que facilitara componentes y estilos para así optimizar el resultado frente al tiempo necesario como para obtener un buen resultado.

Previamente, se tenía experiencia en Bootstrap como *framework* para desarrollar el *front-end*, pero con la intención era conocer algún otro módulo similar que nunca se hubiera usado se buscó alguna otra alternativa. Después de realizar una búsqueda, se descubrió Bulma:

Ilustración 17. Logo Bulma.io



Fuente: Bulma.io

Este fue el seleccionado por una estética agradable, así como por el hecho de que proporcionaba elementos o componentes que serían los que en más se usarían en Sendseg y tenía una documentación con bastantes ejemplos que facilitaba el desarrollo. Además, una gra

ventaja que se comenta en su web es que Bulma es compatible con todos los navegadores más usados en el mercado por los usuarios hoy en día.

En la bibliografía se deja el enlace a su sitio web donde se puede ver toda la documentación que muestra ejemplos de los elementos que se pueden usar así como la explicación de su funcionamiento.

EJS

Sendseg como cualquier otro sistema web deberá disponer de unas vistas con las que interactuará el usuario. Evidentemente, estas vistas se deben de crear usando código HTML. Dentro de NodeJS, y en concreto de Express, existen distintos módulos que permiten ser usados como motor de vistas, entre el que destaca el que se ha usado en este proyecto, EJS (Embedded JavaScript).

La particularidad de este módulo es que permite generar vistas de manera dinámica al procesar las peticiones en el servidor, y en función de los datos que se pasen a estas, sean renderizadas de la manera correcta. Esto se logra permitiendo introducir en el código HTML de las vistas tanto lógica de programación como los datos concretos pasados por el servidor a través de variables que se deban usar al renderizarse las vistas. Para poner un ejemplo sencillo al lector de manera práctica, podemos ver en la siguiente imagen cómo dentro del cuerpo, hay un elemento que tiene como valor “<%=error%>”, esta es la notación que usa ejs para que desde el servidor, se pase el valor del error que se desea mostrar en una variable llamada error.

Ilustración 18. Código archivo errores.ejs

```
<body style="background-color: rgba(42, 118, 53, 0.199);">
  <section class="section is-small">
    <h1 class="title has-text-centered">ERROR</h1>
    <p class="help is-danger has-text-centered" style="font-size: 1rem;"><%=error%></p>
  </section>
```

Fuente: Elaboración propia

Entonces desde una petición del servidor se puede renderizar el valor de “error” con el valor que corresponda.

Ilustración 19. Código en servidor para la renderización adecuada de la plantilla pasando valores

```
const error = "Debes iniciar sesión para acceder a esta página."
const valoresSelector = await leerArchivoTexto()
const logged = false
res.render('errores.ejs', { error, logged, valoresSelector})
```

Fuente: Elaboración propia

Por último, cabe indicar que la forma de cargar los valores de variables con valores simples es de la forma anterior pero para cargar valores más complejos como pueden ser listas se cambia levemente la notación, usando: “<%- variable %>”

Estas son las principales características de las plantillas ejs y han resultado ser muy útiles para la creación de Sendseg debido precisamente a la posibilidad de reutilización que han ofrecido y a la generación de contenido de manera dinámica.

Mysql2

Con la finalidad de realizar la conexión de Sendseg de una manera eficiente y asíncrona con la base de datos MySQL donde se almacena información persistente de Sendseg, se usó un módulo que hace de controlador llamado mysql2.

Bien es cierto que existe un módulo mysql que a priori permitiría hacer la conexión, pero después de leer información acerca de este módulo, al parecer tiene una menor cantidad de usuarios y recibe una menor cantidad de actualizaciones sobre todo relacionadas con temas de fallos de seguridad, cosa que en el módulo usado, mysql2, al parecer no es tan habitual, por eso se seleccionó mysql2 en lugar de mysql.

Es muy importante destacar que para crear la conexión a la base de datos se introducen los siguientes datos:

- User
- Password
- Host
- Port
- Database
- Timezone

Una vez se crea la configuración de la conexión, esta se usa cuando corresponde para realizar las distintas operaciones en la base de datos a medida que el usuario hace peticiones al servidor.

Cabe indicar que en el uso de la conexión con la base de datos, como ciertas operaciones se deben de realizar en un orden concreto, para asegurar esto se ha hecho uso de lo que se conoce como asincronía secuencial. Este enfoque en programación permite ejecutar tareas o instrucciones de manera asíncrona pero en un orden concreto.

Por lo tanto, usando este módulo se ha podido establecer la conexión entre la aplicación de Sendseg y la base de datos MySQL que se usa para asegurar la persistencia de los datos necesarios para que el sistema funcione correctamente.

Bcrypt

Con el objetivo de poder encriptar las contraseñas que los usuarios introducen para registrar su cuenta como Sociedad de Cazadores y que este valor encriptado se pueda almacenar

en la base de datos, se hizo uso del módulo denominado Bcrypt. Este módulo permite la encriptación al hacer uso de una combinación de técnicas, destaca que hace uso del cifrado Blowfish.

En el uso de este módulo para la encriptación de la contraseña antes de almacenarla en la base de datos hay dos pasos que se deben realizar, el primero es hacer **salt** y en caso de que no haya error el **hash**.

Respecto al salt o salting de una cadena de texto es la función que agrega una cadena de texto aleatoria a la contraseña antes de hacer el hash. En el salt, se pasa un parámetro de rondas que en este caso se dejó el valor por defecto de 10, ya que este es el estándar, aunque podría llegar hasta a 30. Esto añade un coste adicional que permite por un lado ganar unicidad en las contraseñas a almacenar, y por otro, establece una dificultad adicional a ataques de fuerza bruta como puedan ser de diccionario o tablas arcoíris. Precisamente, esa dificultad a medida que el parámetro es más elevado, más alta se establece, aunque este incremento de coste temporal de procesamiento crece de manera cuadrática, por lo que se debe de tener cuidado con el valor a establecer ya que el usuario podría tener que esperar mucho tiempo hasta que se termine el proceso de encriptación cosa que haría que el sistema sea lento y tosco para el usuario final. Para ilustrarlo, con un valor de 10 bcrypt realizará 2 elevado a 10 iteraciones del algoritmo de hash, donde cada iteración tiene como entrada el resultado de la ronda anterior.

Entendido el salt, el parámetro de las rondas, y su importancia durante el proceso, se debe explicar el hash. Básicamente, consiste en una función matemática que dada una cadena de caracteres se hace una serie de transformaciones que proporcionan otra cadena de texto con longitud fija. Un pequeño cambio en la cadena de entrada genera un gran cambio en el hash obtenido de manera en que no se puede extraer o deducir el cambio que ha ocurrido en la entrada.

A modo de ejemplo, a continuación se muestra como queda una contraseña encriptada almacenada en la base de datos después usar este módulo y de realizar este proceso:

Ilustración 20. Resultado de la encriptación y valor almacenado en la base de datos

22	encryptedUser@gmail.com	\$2b\$10\$jR2aYbI//6wo9eufZkw50OchEzo/wnuM...	encryptedUser	Abegondo	12356
----	-------------------------	---	---------------	----------	-------

Fuente: Elaboración propia

Por otro lado, también se usa este módulo cuando un usuario desea iniciar sesión e introduce la contraseña en lenguaje natural que es transformado para comprobar si se corresponde con el hash o valor encriptado almacenado en la base de datos.

Nodemailer

El módulo de Nodemailer fue el que se usó para dar solución a la necesidad de enviar correos a los usuarios que sean sociedades cuando se realicen ciertas acciones en el sistema, ya sea por él mismo o por el administrador.

Pasando ya a la solución que se implementó usando este módulo, lo primero a comentar es que se creó un correo de Gmail que sería el que Sendseg usará como origen de cada correo automático que se envíe.

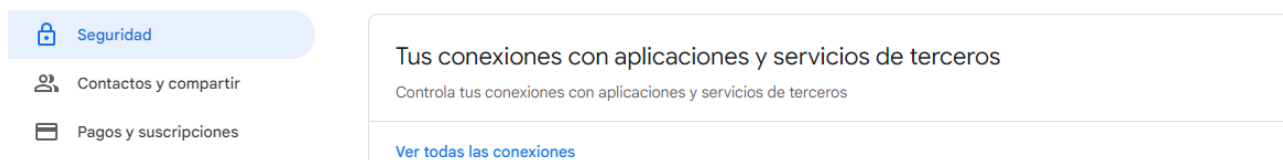
Ilustración 21. Correo creado para el proyecto

Correo electrónico **sendseg.notificaciones@gmail.com**

Fuente: Elaboración propia

Una vez creado este, se activó la autenticación en dos pasos y se proporcionó como forma de autenticación un teléfono. Posteriormente, se creó una clave de aplicación desde este lugar en accediendo a la cuenta.

Ilustración 22. Creación de clave de aplicación en Google



Fuente: Elaboración propia

Este tipo de clave permite que una aplicación tercera, en este caso Sendseg, pueda enviar los correos de manera automática sin la necesidad de iniciar sesión y autenticarse cada vez que se necesite enviar uno.

Comentado esto, nodemailer permite crear lo que se denomina un transportador que contiene una serie de parámetros entre los que está esa clave de aplicación generada. A destacar de la configuración del transportador, que como el correo que hace uso de Nodemailer es Gmail se usa como host el servicio smtp de Gmail. Además, se pone como autenticación el correo creado y como contraseña la clave de aplicación generada, esto es muy importante porque si ponemos la contraseña con la que se ha creado el correo no funcionará ya que pedirá la autenticación en dos pasos y acabaremos obteniendo un error.

Después, cuando se realiza alguna de las siguientes peticiones se usa el transportador para enviar el mensaje que corresponda.

- Registro Sociedad Cazadores por Sociedad de Cazadores
- Registro Sociedad Cazadores por Administrador
- Actualización del perfil de la sociedad de cazadores por Administrador
- Edición actividad caza por Administrador
- Eliminación actividad caza por Administrador
- Borrar informe de actividad caza por Administrador

Cabe indicar que al introducir esta operación de envío del correo después de alguna petición al sistema, añade cierta latencia para la finalización de la petición, aunque la magnitud de esta se ajusta a la de cualquier otro sistema, siendo asumible para cualquier usuario.

Para finalizar con este apartado, se muestra ahora una captura de pantalla de un mensaje que se envía automáticamente después de que el Administrador actualice el perfil de una Sociedad de Cazadores:

Ilustración 23. Resultado al enviar el correo automáticamente



Fuente: Elaboración propia

En versiones posteriores se podría ampliar y mejorar la estética de los correos ya que existe la posibilidad de usar una propiedad "html" que permite incluir código HTML que se enviará en el mensaje y se renderizará correctamente.

Multer

Con el objetivo de permitir la subida y descarga de archivos se encontró un módulo que hacía función de middleware y facilitaba la carga de cualquier tipo de archivo ya sea documento, imagen o video a través de los formularios HTML, este es Multer.

Este módulo requiere definir la configuración de almacenamiento, es decir, indicar donde se suben los archivos. De esta configuración se debe de destacar que cuando se suba en un formulario HTML un fichero, este se guardará con su nombre original que el usuario establezca en una carpeta dentro del proyecto siempre y cuando sea un archivo de tipo pdf y solo se adjunte un único archivo.

Relacionado con este módulo cabe indicar el uso de otro que viene por defecto en NodeJS que se llama "fs" o *File System*, que permite leer los datos de los ficheros subidos usando multer para procesarlo o guardarlos en la base de datos.

Por lo tanto, queda justificado el uso de este módulo, el problema que permite resolver, y cuál es su función en la implementación de Sendseg.

JSTS

Uno de los problemas que se debían afrontar y resolver para que Sendseg funcionara correctamente era como llevar a cabo la comprobación indispensable que se debe realizar para que cuando un usuario crea o edite el polígono que representa el área donde se desarrollará la

actividad se compruebe si este hace intersección con otro ya definido para hacer que el nuevo no sea válido.

Esta comprobación, tal y como se acaba de decir es vital porque usando un símil para ejemplificar esta situación sería como que un sistema de reserva de pistas de pádel permita que dos usuarios distintos puedan reservar la misma pista en un horario que coincide total o parcialmente, básicamente, estaríamos intentando solucionar un problema entre Senderistas y Sociedades de Cazadores para generar otro solo entre las segundas.

Entendida la necesidad, se buscó algún módulo o herramienta que permitiese hacer la operación de la intersección entre dos polígonos dados los vértices que los forman para de este modo en caso de que sí que haya intersección el polígono introducido sea NO válido, y por tanto, la operación que se esté llevando a cabo tampoco lo sea.

Después de buscar, encontré un módulo que permitía crear polígonos y proporcionaba múltiples operaciones entre las que estaba la intersección, este fue JSTS

Cabe indicar que esta operación se ha usado por partida doble, una para que en el cliente, es decir en el navegador web del usuario se compruebe en el mapa la intersección con respecto a los polígonos o áreas dibujadas en él, y el otro ha sido para en el momento de lanzar la petición al servidor para crear la nueva actividad se haga una segunda comprobación en el lado del servidor para usando la base de datos asegurar que la comprobación se hace con los valores más actuales, esto se debe a un problema de concurrencia de que dos posibles usuarios accedan al mismo tiempo al mismo recurso, es decir, al mismo mapa a nivel temporal y en un determinado momento a un mismo lugar.

Para finalizar este apartado decir que pese a que solo se ha tenido interés por la intersección entre polígonos este proporciona más operaciones y otros elementos geométricos con los que poder trabajar, con lo que en futuras ampliaciones y mejoras en Sendseg, este es un buen módulo con el que poder continuar trabajando.

API Google Maps

Ahora ya pasamos a los elementos clave para la implementación de Sendseg, que ya no forman parte de los paquetes NPM que se instalaron de NodeJS.

En el capítulo anterior se decidió hacer uso de la API de Google Maps para usar los mapas y funciones relacionadas que este servicio proporciona. Su principal uso será el de establecer de las actividades de caza e indicar el área donde se desarrollarán estas, así como para que los senderistas puedan hacer sus consultas y ver estas zonas afectadas. Ahora, se va a relatar y mostrar el procedimiento que se siguió para poder hacer uso de la API:

En primer lugar, se vieron todas las APIs y servicios que Google ofrece relacionado con sus mapas, esto se hizo desde su página de desarrolladores que está en la bibliografía:

Ilustración 24. APIs y servicios ofrecidas por Google Maps Platform

Comenzar	Maps	Routes	Places
Cómo comenzar a utilizar Google Maps Platform	API de Maps JavaScript	API de Routes	API de Places
Selector de API	SDK de Maps para Android	API de Roads	SDK de Places para Android
Facturación y precios	SDK de Maps para iOS	API de Directions	SDK de Places para iOS
Informes y Monitoring	API de Maps Static	API de Distance Matrix	Biblioteca de Places para la API de Maps JavaScript
ID de mapa	API de Street View Static		API de Geocoding
Preguntas frecuentes	API de Maps Embed	Soluciones	API de Geolocation
Asistencia y recursos	URLs de Maps	Soluciones de la industria	API de Address Validation
Atención al cliente	API de Maps Elevation	Servicios de movilidad	API de Time Zone
Administración de incidentes			

Fuente: Google Cloud

De entre todas las opciones, se decidió que la solución que se debía de usar era la API de Maps JavaScript. Para poder hacer uso de esta o de cualquier otro servicio en general, se necesita generar una key o clave con la que poder usar la API. Para ello, se creó una cuenta en Google Cloud Platform donde pese a que existe un plan gratuito que era más que suficiente para el desarrollo de Sendseg, se debe de introducir una tarjeta de crédito para que en caso de exceder el número de peticiones mensuales máximas se proceda al cargo correspondiente.

Ilustración 25. Programa gratuito Google Cloud

Programa gratuito de Google Cloud


Consigue 300 USD de crédito gratis y más de 20 productos gratuitos.

Fuente: Google Cloud


Una vez creada la cuenta, se creó el proyecto Sendseg dentro de Google Cloud Console:

Ilustración 26. Creación proyecto Sendseg en Google Cloud Console


Proyecto nuevo

 Tienes 23 projects restantes en tu cuota. Solicita un incremento o borra algunos proyectos. [Más información](#)

[MANAGE QUOTAS](#)

Nombre del proyecto *
Sendseg 

ID de proyecto: sendseg-395310. No se podrá cambiar más tarde. [EDITAR](#)

Ubicación *
 Sin organización [EXPLORAR](#)





Organización o carpeta superior

[CREAR](#) [CANCELAR](#)

Fuente: Elaboración propia desde Google Cloud

Después de crearlo, accedemos al proyecto:

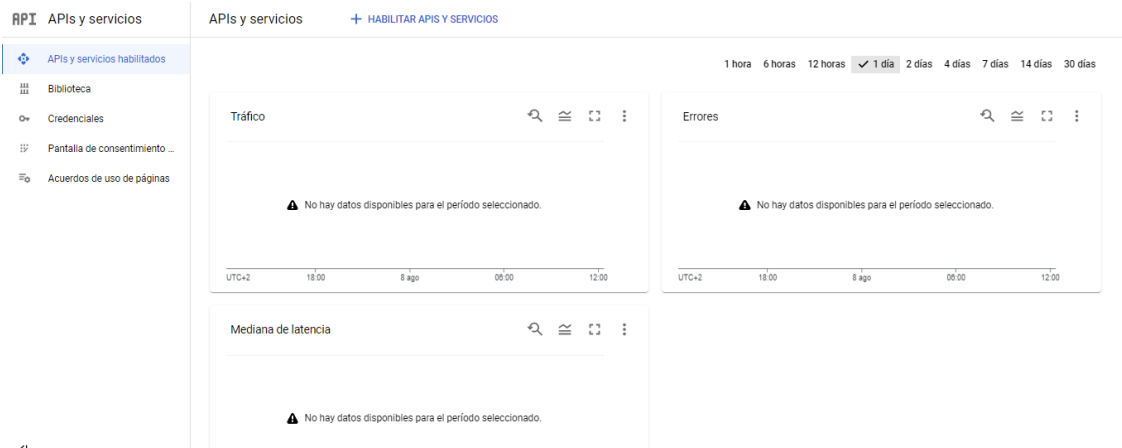
Ilustración 27. Proyecto Sendseg creado en Google Cloud Console

	Nombre	ID
  	SendSeg 	sendseg

Fuente: Google Cloud

Y, o bien, desde el panel vertical izquierdo en la pantalla, o bien, debajo en acceso rápido vamos a APIs y Servicios, donde se nos mostrará la siguiente ventana:

Ilustración 28. APIs y servicios



Fuente: Google Cloud

Desde el panel izquierdo se debe de acceder a Credenciales para crear una que será la que nos proporcionará la API key para usarla en Sendseg. Estas credenciales son muy importantes crearlas con los datos bien ya que se puede restringir que APIs se usarán con estas o la dirección del sitio web que la usará. En este caso los datos que se introdujeron a las credenciales creadas fueron:

Ilustración 29. Creación de credenciales I

Restricciones de clave

Restringir el lugar y la API en los que se puede usar esta clave ayuda a evitar el uso no autorizado. [Más información](#)

Establece una restricción de aplicaciones

Las restricciones de aplicaciones limitan el uso de las claves de API a sitios web, direcciones IP y aplicaciones para Android o iOS específicas. Puedes configurar una restricción de aplicaciones por clave.

- Ninguno
- Sitios web
- Direcciones IP
- Apps para Android
- Apps para iOS

Fuente: Elaboración propia desde Google Cloud

Ilustración 30. Creación de credenciales II

Filtro	Ingresar el nombre o el valor de la propiedad	
<input type="checkbox"/>	Estado	Sitio web Editar
<input type="checkbox"/>		localhost:3000/*

Fuente: Elaboración propia desde Google Cloud

Ilustración 31. Creación de credenciales III

Restricciones de API

Las restricciones de API especifican las API habilitadas que esta clave puede llamar.

- No restringir clave
Esta clave puede llamar a cualquier API
- Restringir clave

3 API

API seleccionadas:

Maps JavaScript API
Places API
Places API (New)

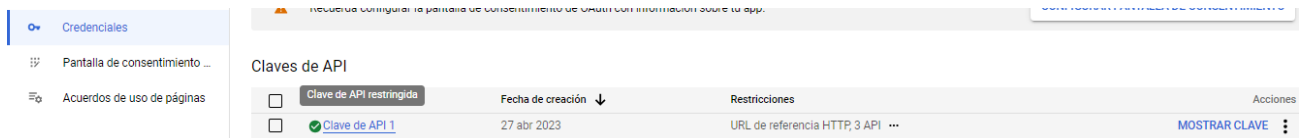
Nota: Es posible que la configuración tarde hasta 5 minutos en aplicarse.

GUARDAR CANCELAR

Fuente: Elaboración propia desde Google Cloud

Al darle a guardar, podemos observar como la credencial se crea correctamente:

Ilustración 32. Credencial creada para Sendseg



Fuente: Elaboración propia usando Google Cloud

Una vez creada, si el desarrollador pulsa en “Mostrar Clave”, se le muestra la API key que deberá usar en el código de la aplicación o sistema que esté desarrollando. Aquí no se muestra por motivos de privacidad y para que no pueda ser usada por nadie que lea esta memoria, dado que un uso de esta podría generar cargos asociados a mi tarjeta personal.

A modo de curiosidad, existen otros servicios que ofrece Google que se pueden usar de manera gratuita pero no están habilitadas por defecto, para habilitarlas se deberá de ir al apartado de Biblioteca, buscar la API y habilitarla, aunque posteriormente se deberá de modificar la credencial establecida añadiendo la nueva API habilitada que será usada en la aplicación. De hecho, esto se hizo con el servicio de Google Places que permite poner una barra de búsqueda para buscar lugares.

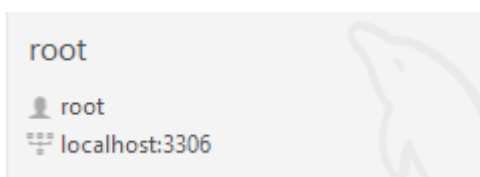
Mostrado esto, queda explicado el procedimiento que se siguió para crear la API key, a partir de ahí en la bibliografía se dejará el enlace a la documentación de la API que se usó para ver ejemplos de cómo usarla para mostrar mapas, poner la herramienta de dibujo, crear polígonos, eventos etc.

MySQL

Respecto a la creación de la base de datos relacional se usó la herramienta de MySQL Workbench.

En primer lugar, se accedió a la conexión del usuario root que puede realizar cualquier acción ya que tiene todos los permisos:

Ilustración 33. Conexión root MySQL worckbench



Fuente: Elaboración propia

Usando esta conexión, se crea el schema denominado 'sendseg':

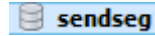
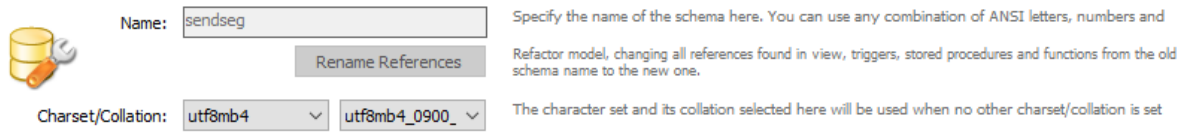


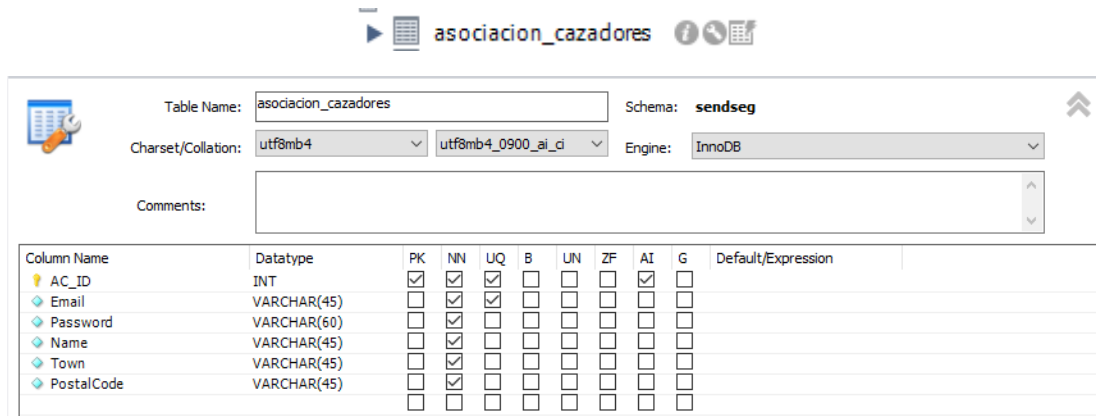
Ilustración 34. Creación del schema



Fuente: Elaboración propia

Una vez creado este, se van creando todas y cada una de las tablas tal y como se definieron en el diagrama relacional, cabe indicar que la tabla asociación_cazadores se corresponde con las Sociedades de cazadores ya que es terminológicamente es más apropiado denominarlas sociedades y no asociaciones. A continuación, se muestran capturas desde la interfaz de MySQL Workbench:

Ilustración 35. Creación y configuración de la tabla asociacion_cazadores



Fuente: Elaboración propia

Ilustración 36. Creación y configuración tabla zona_de_caza

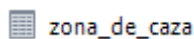


Table Name: Schema: **sendseg**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Zona_Caza_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Initial_Date	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
End_Date	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Allowed_Species	VARCHAR(70)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Type_Activity	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
FK_AC_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Foreign Key Name	Referenced Table	Column	Referenced Column
AC_ID	`sendseg`.`asociacion_cazadores`	<input type="checkbox"/> Zona_Caza_ID <input type="checkbox"/> Initial_Date <input type="checkbox"/> End_Date <input type="checkbox"/> Allowed_Species <input type="checkbox"/> Type_Activity <input checked="" type="checkbox"/> FK_AC_ID	AC_ID

Foreign Key Options

On Update:

On Delete:

Skip in SQL generation

Fuente: Elaboración propia

Ilustración 37. Creación y configuración tabla coordenadas

Table Name: Schema: **sendseg**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Coordenadas_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
latitud	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
longitud	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
FK_Zona_Caza_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Foreign Key Name	Referenced Table	Column	Referenced Column
FK_Zona_Caza	`sendseg`.`zona_de_caza`	<input type="checkbox"/> Coordenadas_ID <input type="checkbox"/> latitud <input type="checkbox"/> longitud <input checked="" type="checkbox"/> FK_Zona_Caza_ID	Zona_Caza_ID

Foreign Key Options

On Update:

On Delete:

Skip in SQL generation

Fuente: Elaboración propia

Ilustración 38. Creación y configuración tabla informe_actividad

informe_actividad

Table Name: Schema: **sendseg**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Informe_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
archivo	LONGBLOB	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
FK_Zona_de_Caza_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Foreign Key Name	Referenced Table	Column	Referenced Column	Foreign Key Options
FK_Zona_de_Caza_ID	`sendseg`.`zona_de_caza`	<input type="checkbox"/> Informe_ID		On Update: <input type="text" value="CASCADE"/>
		<input type="checkbox"/> archivo		On Delete: <input type="text" value="CASCADE"/>
		<input checked="" type="checkbox"/> FK_Zona_de_Ca...	Zona_Caza_ID	<input type="checkbox"/> Skip in SQL generation

Fuente: Elaboración propia

Una vez mostradas todas las capturas, en caso de que se desee replicar la base de datos, se deja en el [Anexo 3](#) el script que permitiría la creación.

Python

En este apartado cabe indicar una cosa para la que se hizo uso de Python. El uso de este fue necesario porque para en el formulario de registro de sociedades uno de los campos a incluir era el Municipio en el que está adscrito o pertenece la sociedad. Este campo, podría haberse rellenado de manera manual con un campo input de tipo texto en el cual el usuario manualmente introdujera el lugar, aunque esto no me parecía lo mejor, ya que las personas podrían equivocarse al introducirlo o podrían darse dos valores que se refiriesen al mismo municipio pero los usuarios lo introdujesen en idiomas distintos dándose lugar a valores estrictamente diferentes, por ejemplo: Alcoi/Alcoy.

Por ello, para evitar esto, lo que decidí fue utilizar un campo que fuese un selector y que se tuvieran como opciones todos y cada uno de los nombres de los municipios sin repetir. El problema radicaba justamente en obtener una lista con todos los municipios de España, y lo que hice fue buscar si existía algún repositorio u archivo que los tuviera, aunque este no fue el caso o al menos no lo encontré.

Tras una búsqueda, encontré que el Instituto Nacional de Estadística o INE, ofrece la posibilidad de usar de manera gratuita su API para realizar peticiones con la finalidad de obtener estadísticas u otros valores como el de las variables, entre los que están las localidades o municipios. Para ver la petición que necesitaba use su propio generador de URLs obteniendo por cada petición 500 valores:

https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=17

Esta es parcialmente la respuesta que proporciona esta petición:

Ilustración 39. Resultado petición INE

```
[{"Id":8506, "Fk_Variable":19, "Nombre":"Ausejo", "Codigo":"26020", "FK_JerarquiaPadres":[27,392393]}
,{"Id":8507, "Fk_Variable":19, "Nombre":"Bañares", "Codigo":"26024", "FK_JerarquiaPadres":[27,392392]}
,{"Id":8508, "Fk_Variable":19, "Nombre":"Oliana", "Codigo":"25149", "FK_JerarquiaPadres":[26,392293]}
,{"Id":8509, "Fk_Variable":19, "Nombre":"Os de Balaguer", "Codigo":"25156", "FK_JerarquiaPadres":[26,392286]}
,{"Id":8510, "Fk_Variable":19, "Nombre":"Portella, La", "Codigo":"25174", "FK_JerarquiaPadres":[26,392288]}
,{"Id":8511, "Fk_Variable":19, "Nombre":"Castellar", "Codigo":"23025", "FK_JerarquiaPadres":[24,392114]}
,{"Id":8512, "Fk_Variable":19, "Nombre":"Escalañuela", "Codigo":"23031", "FK_JerarquiaPadres":[24,392113]}
,{"Id":8513, "Fk_Variable":19, "Nombre":"Fonz", "Codigo":"22110", "FK_JerarquiaPadres":[23,392129]}
,{"Id":8514, "Fk_Variable":19, "Nombre":"Fueva, La", "Codigo":"22113", "FK_JerarquiaPadres":[23,392130]}
,{"Id":8515, "Fk_Variable":19, "Nombre":"Huerto", "Codigo":"22124", "FK_JerarquiaPadres":[23,392131]}
,{"Id":8516, "Fk_Variable":19, "Nombre":"Ilche", "Codigo":"22128", "FK_JerarquiaPadres":[23,392129]}
,{"Id":8517, "Fk_Variable":19, "Nombre":"Labuerda", "Codigo":"22133", "FK_JerarquiaPadres":[23,392130]}
,{"Id":8518, "Fk_Variable":19, "Nombre":"Lascuarre", "Codigo":"22142", "FK_JerarquiaPadres":[23,392136]}
,{"Id":8519, "Fk_Variable":19, "Nombre":"Nava, La", "Codigo":"21051", "FK_JerarquiaPadres":[22,392103]}
,{"Id":8520, "Fk_Variable":19, "Nombre":"Rosal de la Frontera", "Codigo":"21062", "FK_JerarquiaPadres":[22,392103]}
,{"Id":8521, "Fk_Variable":19, "Nombre":"Zufre", "Codigo":"21079", "FK_JerarquiaPadres":[22,392103]}
,{"Id":8522, "Fk_Variable":19, "Nombre":"Aisa", "Codigo":"22006", "FK_JerarquiaPadres":[23,392132]}
,{"Id":8523, "Fk_Variable":19, "Nombre":"Alcalá de Gurreea", "Codigo":"22014", "FK_JerarquiaPadres":[23,392131]}
,{"Id":8524, "Fk_Variable":19, "Nombre":"Algora", "Codigo":"19017", "FK_JerarquiaPadres":[20,392256]}
,{"Id":8525, "Fk_Variable":19, "Nombre":"Almonacid de Zorita", "Codigo":"19022", "FK_JerarquiaPadres":[20,392258]}
,{"Id":8526, "Fk_Variable":19, "Nombre":"Cogollos de Guadix", "Codigo":"18049", "FK_JerarquiaPadres":[19,392097]}
,{"Id":8527, "Fk_Variable":19, "Nombre":"Dílar", "Codigo":"18068", "FK_JerarquiaPadres":[19,392095]}
,{"Id":8528, "Fk_Variable":19, "Nombre":"Galera", "Codigo":"18082", "FK_JerarquiaPadres":[19,392102]}
,{"Id":8529, "Fk_Variable":19, "Nombre":"Gualchos", "Codigo":"18093", "FK_JerarquiaPadres":[19,392096]}
,{"Id":8530, "Fk_Variable":19, "Nombre":"Íllora", "Codigo":"18102", "FK_JerarquiaPadres":[19,392099]}
,{"Id":8531, "Fk_Variable":19, "Nombre":"Juviles", "Codigo":"18112", "FK_JerarquiaPadres":[19,392100]}
,{"Id":8532, "Fk_Variable":19, "Nombre":"Huelves", "Codigo":"16108", "FK_JerarquiaPadres":[17,392250]}
,{"Id":8533, "Fk_Variable":19, "Nombre":"Iniesta", "Codigo":"16113", "FK_JerarquiaPadres":[17,392251]}
,{"Id":8534, "Fk_Variable":19, "Nombre":"Leganiel", "Codigo":"16119", "FK_JerarquiaPadres":[17,392252]}
,{"Id":8535, "Fk_Variable":19, "Nombre":"Minglanilla", "Codigo":"16125", "FK_JerarquiaPadres":[17,392251]}
,{"Id":8536, "Fk_Variable":19, "Nombre":"San Sadurniño", "Codigo":"15076", "FK_JerarquiaPadres":[16,392350]}
,{"Id":8537, "Fk_Variable":19, "Nombre":"Tordoa", "Codigo":"15084", "FK_JerarquiaPadres":[16,392352]}
,{"Id":8538, "Fk_Variable":19, "Nombre":"Vedra", "Codigo":"15089", "FK_JerarquiaPadres":[16,392351]}
,{"Id":8539, "Fk_Variable":19, "Nombre":"Zas", "Codigo":"15093", "FK_JerarquiaPadres":[16,392351]}
,{"Id":8540, "Fk_Variable":19, "Nombre":"Acebrón, El", "Codigo":"16002", "FK_JerarquiaPadres":[17,392250]}
,{"Id":8541, "Fk_Variable":19, "Nombre":"Trebujena", "Codigo":"11037", "FK_JerarquiaPadres":[12,392084]}
,{"Id":8542, "Fk_Variable":19, "Nombre":"Portaje", "Codigo":"10150", "FK_JerarquiaPadres":[11,392342]}
,{"Id":8543, "Fk_Variable":19, "Nombre":"Riolobos", "Codigo":"10155", "FK_JerarquiaPadres":[11,392342]}
,{"Id":8544, "Fk_Variable":19, "Nombre":"Ruanes", "Codigo":"10161", "FK_JerarquiaPadres":[11,392346]}
,{"Id":8545, "Fk_Variable":19, "Nombre":"Gordo, El", "Codigo":"10085", "FK_JerarquiaPadres":[11,392348]}
,{"Id":8546, "Fk_Variable":19, "Nombre":"Pallejà", "Codigo":"08157", "FK_JerarquiaPadres":[9,392268]}
,{"Id":8547, "Fk_Variable":19, "Nombre":"Granja de Torrehermosa", "Codigo":"06059", "FK_JerarquiaPadres":[7,392330]}
```

Fuente: Elaboración propia, extraído del INE

Por tanto, lo que hice fue usando Google Colab hacer 17 peticiones para obtener todos los valores y las respuestas, convertirlas a JSON para obtener los valores, y procesarlos de forma en que se almacenaran un archivo de texto con todos los municipios ordenados alfabéticamente de línea en línea separados por un “\n” y sin que estuviese un solo valor repetido, ya que la petición solo devolvía los nombres pero la comunidad u otro valor que permitiese diferenciar el municipio de Torrent de Valencia del Torrent de Girona.

El código asociado a toda esta operación descrita es el siguiente:

Ilustración 40. Código Python 1

```
1 import requests
2 import json
3
4 municipios = []
5 municipiosOrdenados = []
6 def obtener_nombre_desde_json_Ordenado(url, archivo_salida):
7     global municipiosOrdenados
8     # Realizar la solicitud GET
9     respuesta = requests.get(url)
10
11     # Comprobar si la solicitud fue exitosa
12     if respuesta.status_code == 200:
13         # Obtener el JSON de respuesta
14         json_respuesta = respuesta.json()
15         print(json_respuesta)
16         # Obtener el valor de la propiedad "Nombre"
17         for municipio in json_respuesta:
18             #Si hay dos mismo municipios con mismo nombre solo habrá uno que se identificará por el Código Postal
19             if municipio['Nombre'] not in municipios:
20                 municipios.append(municipio['Nombre'])
21         else:
22             print("No se pudo realizar la solicitud GET.")
23
24     municipiosOrdenados = sorted(municipios)
25
```

Fuente: Elaboración propia

Aquí vemos la función “obtener_nombre_desde_json_ordenado(url,archivo_salida)”, que dada una URL para la petición genera y un nombre de archivo, guarda los nombres, comprueba que no sean repetidos y los va ordenando alfabéticamente.

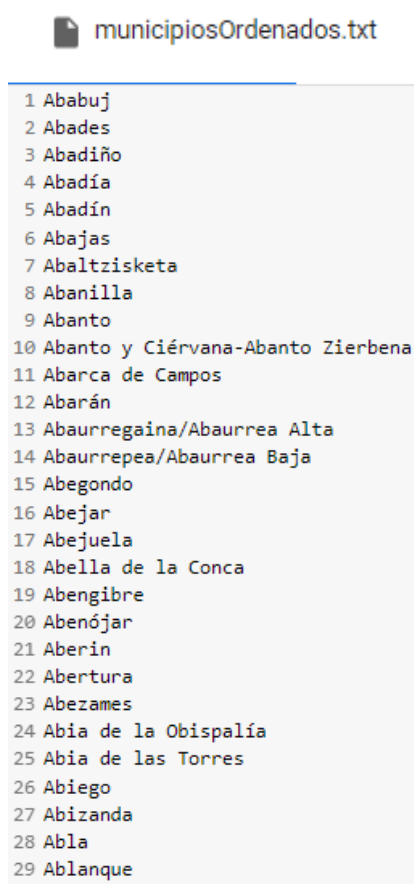
Ilustración 41. Código Python 2

```
1 # URL de la API que devuelve el JSON
2 url_api = "https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page="
3
4 # Nombre del archivo de salida
5 archivo_salida = "municipiosOrdenados.txt"
6
7 # Llamar a la función para obtener el nombre y guardarlo en el archivo
8
9 for i in range(1, 18):
10     obtener_nombre_desde_json_Ordenado(url_api+str(i), archivo_salida)
11     print(url_api+str(i))
12
13 for municipio in municipiosOrdenados:
14     with open(archivo_salida, "a") as archivo:
15         archivo.write(municipio+"\n")
16     #print(f"El nombre '{nombre}' ha sido guardado en '{archivo_salida}'.")
17 print(municipiosOrdenados[:50])
```

```
[{'Id': 456, 'Fk_Variable': 19, 'Nombre': 'Orbaizeta', 'Codigo': '31195', 'FK_JerarquiaPadres': [32, 392378]}, {'Id': 457, 'Fk_Variable': 19, 'Nombre': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=1'}, {'Id': 956, 'Fk_Variable': 19, 'Nombre': 'Camarles', 'Codigo': '43903', 'FK_JerarquiaPadres': [43, 392297]}, {'Id': 957, 'Fk_Variable': 19, 'Nombre': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=2'}, {'Id': 1456, 'Fk_Variable': 19, 'Nombre': 'Berrocal de Huebra', 'Codigo': '37050', 'FK_JerarquiaPadres': [37, 392208]}, {'Id': 1457, 'Fk_Variable': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=3'}, {'Id': 1956, 'Fk_Variable': 19, 'Nombre': 'Viver í Serrateix', 'Codigo': '08308', 'FK_JerarquiaPadres': [9, 392274]}, {'Id': 1957, 'Fk_Variable': 19, 'Nombre': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=4'}, {'Id': 2456, 'Fk_Variable': 19, 'Nombre': 'Senés de Alcubierre', 'Codigo': '22218', 'FK_JerarquiaPadres': [23, 392131]}, {'Id': 2457, 'Fk_Variable': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=5'}, {'Id': 2956, 'Fk_Variable': 19, 'Nombre': 'Villarejo', 'Codigo': '26172', 'FK_JerarquiaPadres': [27, 392392]}, {'Id': 2957, 'Fk_Variable': 19, 'Nombre': 'https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=6'}]
```

Fuente: Elaboración propia

En este código se llama a la URL desde 1 hasta 17 ya que de 500 en 500 elementos obtenemos la totalidad de municipios y se genera el archivo municipiosOrdenados.txt:



Fuente: Elaboración propia

Descargamos el archivo y es el que se lee para rellenar cada valor posible del selector municipio para dar de alta a una sociedad o modificar su valor.

Aunque la idea del selector es la que se tuvo inicialmente tal y como se ha relatado en este apartado, cuando se probó el resultado del selector de manera práctica, detecté tres problemas principales:

El primer problema, radicaba en que a la hora de desplegar los valores del selector al haber tantísimas opciones, más de 8.000, que cargar se producía un retraso considerable desde que el usuario pulsaba el selector hasta el despliegue.

El segundo problema era que aunque las opciones estaban ordenadas alfabéticamente, no era agradable para la experiencia del usuario tener que buscar entre tantas opciones la que deseaba seleccionar.

El tercer problema consistía en que un selector no permite indicar un valor fuera de las opciones que se ofrecen al usuario, y aunque en principio se recogen todos los municipios, me di cuenta de que por ejemplo si un usuario desea establecer como valor “Valencia” no se encuentra ese valor, lo cual no es descartable que pueda suceder con otros.

Por todo esto, finalmente se decidió en lugar de usar un selector y tener estos problemas asociados, usar un campo de texto input en el cual se ofrece la posibilidad de autocompletado a

medida que el usuario introduce su valor deseado con los municipios obtenidos que se introducen como opciones en un datalist asociado a ese input. Este cambio da solución a los tres problemas explicados anteriormente.

Capítulo 5: Testing

Una vez comentados los elementos de la implementación y de cada uno de ellos los aspectos más relevantes, vamos con lo que se correspondería con la siguiente fase de la metodología en cascada que es la fase de pruebas.

Esta fase tiene como objetivo identificar posibles fallos, defectos o errores en el software que se está o se ha desarrollado. Dentro de las pruebas, existen muchos tipos posibles:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de compatibilidad
- Pruebas de usabilidad
- Pruebas de seguridad
- Pruebas de aceptación
-

Evidentemente, todas las pruebas son muy importantes, pero debido a factores como la limitación del tiempo entre otros se dieron prioridad a unas respecto de otras. A continuación, se van a hablar de las pruebas que se pudieron llevar a cabo y cómo se realizaron.

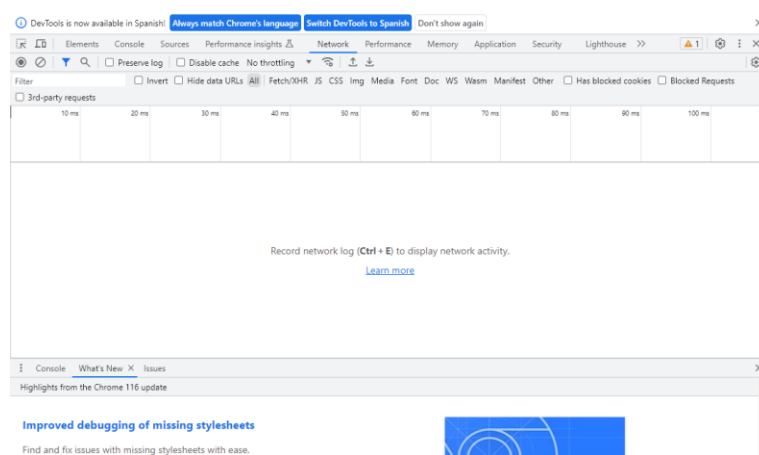
Pruebas unitarias

Las pruebas unitarias se centran en comprobar si las distintas funciones, métodos o clases concretas tienen algún error o problema.

En Sendseg, al ser una aplicación web donde el usuario va a ir realizando peticiones al servidor estas pruebas se centraron en comprobar el comportamiento de cada una de las peticiones, y se usaron dos formas para hacer las comprobaciones:

- **Herramientas Cliente/Navegador:** Interacción por interfaz con el sistema, así como con las propias herramientas de desarrollo del navegador. Por ejemplo, se usaba la red o *Network* para comprobar que se hacían las peticiones correctas y se pasaban bien los parámetros o la consola para comprobar ciertos valores.

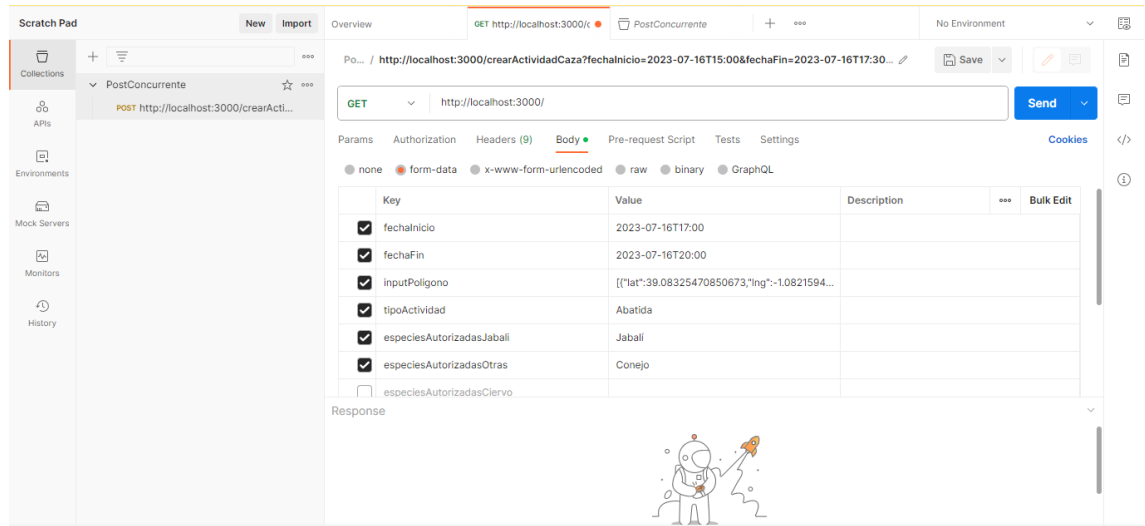
Ilustración 43. Herramientas de desarrollo navegador



Fuente: Navegador Chrome

- **Postman:** Programa que permite enviar todo tipo de peticiones al servidor para comprobar la respuesta después de procesarlas.

Ilustración 44. Pruebas en Postman



Fuente: Postman

A medida que se desarrollaban las peticiones del servidor se fueron realizando pruebas intentando alcanzar una cobertura máxima, tanto de instrucción, como de rama y de camino. Esto permite comprobar que en todos los casos las peticiones responden como se debería de esperar que se comporten.

Pruebas integración

Respecto a las pruebas de integración hacen referencia a comprobar cómo las distintas unidades de código de las pruebas unitarias se comportan cuando interactúan y se comunican unas con las otras. En este caso, en Sendseg, se referirían a cómo las distintas peticiones al servidor funcionan o interactúan cuando previa o posteriormente se hacen otras y que se muestra después de procesar una petición.

En este caso, y por una cuestión de comodidad, se usó íntegramente las herramientas de desarrollo en el cliente o navegador y en este aspecto se comprobó principalmente que la respuesta que se procesaba en el lado del servidor se comunicará correctamente con la vista y se mostrará lo que se debía de mostrar del modo adecuado. También al revés, que aquello que se hacía en el lado del cliente se enviaba bien a la petición que se hiciese al servidor.

Un ejemplo de este tipo de comprobaciones sería cuando una sociedad creaba una actividad y definía el área por medio del polígono, donde se comprobaba que el polígono definido se pasase bien desde el cliente hasta la petición que luego crea este. O al contrario, como se pasaban desde el servidor los polígonos a dibujar en el mapa.

Pruebas compatibilidad

Las pruebas de compatibilidad como su propio nombre indica se encargan de asegurar que el software desarrollado sea compatible o se pueda ejecutar en entornos heterogéneos. Estos entornos diferentes pueden ir desde dispositivos con tamaños de pantallas distintos, hasta diferentes sistemas operativos o simplemente clientes o navegadores distintos.

En el caso de Sendseg, se hicieron pruebas para comprobar que funcionaba bien desde los distintos clientes o navegadores más usados del mercado, cosa que en principio no iba a haber problema porque Bulma está diseñado precisamente para funcionar en ellos.

Además, cuando se hizo el despliegue en la nube que se comentará en el siguiente apartado, se entró a la aplicación web desde distintos dispositivos como teléfonos, portátil y ordenador de sobremesa y no se detectó ningún problema. Así que a falta de que en un futuro se hagan comprobaciones con todavía muchos más dispositivos y entornos parecen que en principio no hay problema, y en caso de haberlo, debe de ser bajo alguna circunstancia o entorno particular

Pruebas de aceptación

Las pruebas de aceptación son aquellas en las que los usuarios finales del sistema prueban este para determinar si el software cumple con sus expectativas y con las necesidades que deben de ser satisfechas.

En el caso de Sendseg, ha sido un desarrollo que no ha tenido un seguimiento directo por parte de los usuarios finales, ni senderistas ni sociedades de cazadores. Además, una vez se ha tenido la primera versión por una cuestión de fechas no se ha podido desplegar el sistema en nuestra propia máquina para poder hacer las pruebas para que una sociedad de cazadores lo probara y proporcionara *feedback* con sus opiniones. Por ello, este tipo de prueba no se ha podido hacer a tiempo pero sin ningún tipo de duda, en cuanto se haga el despliegue de la aplicación en la máquina que dispondremos, se hará lo primero para asegurarnos que todos lo realizado les parece bien y así focalizarnos en las siguientes funcionalidades o requisitos que ellos consideren que se debería priorizar.

Pese a no tener la posibilidad del despliegue en la máquina que deseábamos, y aunque tuvo muchas limitaciones, se realizó un despliegue usando un servicio para desplegar aplicaciones en la nube proporcionado por Railway. En caso de tener interés en ello, y cómo se hizo, se ha dejado el [Anexo 5](#) para su lectura.

Capítulo 6: Conclusiones

En este capítulo se se van a comentar las conclusiones respecto al desarrollo de Sendseg en este proyecto, después se comentarán aspectos a trabajar en un futuro.

En primer lugar, hay que indicar que distintos conocimientos que he adquirido a lo largo del Máster o en los que he profundizado en este respecto a la carrera, me han permitido desarrollar Sendseg. Los dos ejemplos más claros de aplicar conocimientos adquiridos en el máster son todo lo relacionado con las APIs y su consumo, en este caso con la de Google Maps JavaScript, así como también para la definición de las peticiones al servidor que acceden o manipulan los recursos usando los distintos tipos según sea oportuno.

Además de destacar los conocimientos aplicados en este proyecto, cabe indicar que todos los módulos y frameworks que se han usado para implementar Sendseg, no se habían usado nunca de modo que aunque haya requerido un esfuerzo adicional, me han servido como aprendizaje que me permitirá de una forma mucho más inmediata hacer su uso en un futuro.

Pasando a comentar la solución generada, este proyecto se ha enfocado en ofrecer una solución al problema expuesto en el primer capítulo, por ello se considera que la versión actual de Sendseg tiene la cantidad necesaria de funcionalidades para considerar el problema principal como resuelto y que proporciona tanto valor añadido como para que las Sociedades y Senderistas comiencen a usarlo, a falta de hacer las pruebas de aceptación y verificarlo. Pese a esto, tanto futuros alumnos del Máster como yo mismo, podríamos seguir mejorando el sistema para ofrecer una mayor cantidad de funcionalidades y resolver cualquier otro problema relacionado o nuevas necesidades que surjan a cualquiera de las partes implicadas. De hecho, durante el desarrollo de estas se fueron ocurriendo múltiples ideas que por una cuestión de tiempos y crecimiento del proyecto no se han alcanzado de manera complementaria, y es por eso que, en el siguiente apartado se plantearán algunas de estas posibles mejoras futuras y extensiones a nivel principalmente de funcionalidad.

Líneas futuras de trabajo

En este último apartado, se van a proponer posibles mejoras futuras que se podrían hacer respecto al producto software que se ha desarrollado. Cabe indicar que un trabajo futuro y que será de lo primero que se plantee sea el despliegue en nuestra propia máquina de la solución, pero si nos centramos en mejoras a nivel de funcionalidad y otras características, a continuación se hace un desglose por cada tipo de usuario:

Sociedades Cazadores

En primer lugar, las líneas futuras de trabajo centradas en las sociedades de cazadores podrían ser:

- Posibilidad de que los asociados puedan registrarse indicando a qué sociedad pertenecen e incluir toda la documentación necesaria (documento nacional de identidad, permisos, licencias de armas u otros) para poder participar en las actividades. Esto haría que cuando una sociedad cree una nueva actividad de caza, según su

tipología: gancho o abatida, se podrían los propios asociados adherir e inscribirse hasta alcanzar el cupo de plazas que correspondan.

- Ampliar la parte relativa a los informes de las actividades desarrolladas, una opción bastante interesante podría ser usar un motor de análisis de documentos que permita extraer información de los informes analizarla y obtener conclusiones que puedan ser relevantes o de interés tanto para sociedades como asociados. Además, las sociedades se podrían nutrir mutuamente de información con la cual se podría llevar de manera más automatizada el conteo de animales de cada especie que se ha cazado en una determinada zona, haciendo que se pueda decidir con más información cuando y qué cazar.
- Proporcionar una funcionalidad de que cuando una sociedad va a crear una nueva actividad se le muestren sus actividades anteriores para que si hace click en una de ellas se dibuje para la nueva el polígono o área que se definió en esa actividad pasada. Esto permitiría la reutilización de las áreas de caza ya definidas previamente y facilitaría el proceso de creación a las sociedades que deban realizar polígonos que están formados por muchos vértices cuando ya lo han hecho previamente.
- Hacer uso de otras librerías de la API de Google. Por ejemplo, si se usase Geocode, a partir del municipio y del código postal que introducen las sociedades al registrarse este segundo permite la diferenciación de municipios con el mismo nombre, se podrían hacer peticiones para que dado el lugar se devuelva la coordenada, compuesta por latitud y longitud, con la que cuando una sociedad vaya a crear una actividad el mapa se centre en dicha ubicación, esto sería una pequeña mejora aunque implique un incremento en las llamadas a la API.
- En el momento de la creación de una actividad, cuando se vaya a dibujar el polígono de esta, que al hacer click sobre otros polígonos dibujados por otras sociedades se despliegue en un marcador información que pueda ser relevante o de interés para otras sociedades sobre qué actividad van a realizar.
- Posibilidad de que durante el desarrollo de una actividad se conozca la ubicación a tiempo real de donde se encuentra cada uno de los miembros que la realiza e incluso puedan hacer avisos de avistamientos de especies en un punto y momento concreto.
- Se podrían mandar notificaciones a las sociedades y a los cazadores que forman parte de una actividad de caza, incluyendo información como el momento y lugar en el que un senderista que comparte su ubicación ha intersectado y se ha introducido dentro de la zona donde se está desarrollando la actividad.

Senderistas

Respecto a las líneas futuras de trabajo de las que los senderistas se podrían beneficiar, podemos identificar los siguientes ítems que se podrían incluir en futuras iteraciones sobre el producto:

- Funcionalidad que permita al senderista introducir una ruta que desea desarrollar así como el momento de inicio de dicha actividad, con el objetivo de que el sistema le diga si será segura o no durante la duración que estime Google Maps de la ruta.
- Posibilidad de que el senderista introduzca un polígono de una zona que le gustaría recorrer junto a la fecha de inicio y de fin, es decir la duración que le interesa, y que el sistema en función de estos inputs le devuelva una ruta que maximice la seguridad, entendiéndose esta como la combinación de distancia tanto física, como temporal, respecto a las actividades de caza que se vayan a desarrollar en ese intervalo de tiempo en esa zona.
- Una función que cuando el senderista esté realizando una ruta suba su ubicación a tiempo real y se haga la comprobación de si intersecta o toca alguno de las zonas de caza o polígonos para que en el momento del contacto se le envíe un aviso de modo que haya una reacción por su parte y evite un riesgo innecesario y el conflicto que podría generarse con la sociedad que esté operando allí.

Administraciones públicas

Respecto a las administraciones públicas, se podrían proporcionar funcionalidades relacionadas con estadísticas para tener un mayor control sobre cuándo se caza, qué se caza o ha cazado así como obtener datos sobre los lugares más consultados por los senderistas y las fechas y duraciones de las actividades que desarrollan para tener datos que les permitan tener un conocimiento total de la situación en este ámbito y les facilite la toma de decisiones al tener acceso a los datos que les permiten generar la información que necesiten y con ello conocimiento.

Desarrolladores de otras aplicaciones

Por último, este es el stakeholder al que no se le proporcionó atención en esta primera versión de Sendseg. La idea y el trabajo futuro que se tendría que desarrollar es generar una API que pueda ser consumida por otros desarrolladores de rutas como Wikiloc para que desde sus propias aplicaciones web se dé una mayor cobertura y posibilidad de que los senderistas se enteren de si una de sus rutas es o no segura. Esto permitiría por un lado potencialmente poder monetizar Sendseg, y por el otro, darla más a conocer.

Bibliografía

IEEE Xplore: Advanced Search. (s. f.). <https://ieeexplore.ieee.org/search/advanced>

Aplicación Diputación Álava (s. f.). <https://web.araba.eus/es/caza/batidas-caza-mayor>

Mbf. (s. f.). *Application Land Share: un retour d'expérimentation enthousiaste.* mbf-france. <http://mbf-france.fr/actionnationaleinter/application-land-share-retour-dexperimentation-enthousiaste/>

Download / Node.js. (s. f.). Node.js. <https://nodejs.org/en/download>

INE - Instituto Nacional de Estadística. (s. f.). *Productos y servicios / Datos abiertos / API JSON / Generador de URLs JSON.* INE. <https://www.ine.es/dyngs/DataLab/manual.html?cid=66>

INE - Instituto Nacional de Estadística. (s. f.). *Petición para obtener municipios desde page=1 hasta 17.*
https://servicios.ine.es/wstempus/js/ES/VALORES_VARIABLE/19?page=1

NPM: Express. (s. f.). npm. <https://www.npmjs.com/package/express>

NPM: express-session. (s. f.). npm. <https://www.npmjs.com/package/express-session>

NPM: bulma. (s. f.). npm. <https://www.npmjs.com/package/bulma>

Bulma: free, open source, and modern CSS framework based on Flexbox. (s. f.).
<https://bulma.io/>

NPM: Nodemailer. (s. f.). npm. <https://www.npmjs.com/package/nodemailer>

NPM: BCrypt. (s. f.). npm. <https://www.npmjs.com/package/bcrypt>

NPM: Mysql2. (s. f.). npm. <https://www.npmjs.com/package/mysql2>

NPM: JSTS. (s. f.). npm. <https://www.npmjs.com/package/jsts>

NPM: Multer. (s. f.). npm. <https://www.npmjs.com/package/multer>

Google Cloud Platform. (s. f.). <https://console.cloud.google.com/>

GitHub: Let's build from here. (s. f.). GitHub. <https://github.com/>

Railway. (s. f.). Railway. <https://railway.app/>

Anexo 1: Descripciones Diagramas Casos de Uso

Tabla 5. Caso uso: Registrarse

Caso de uso	Registrarse
Precondición	Sociedad de cazadores no registrada previamente
Postcondición	Sociedad de Cazadores ha sido registrada e inicia sesión con las credenciales introducidas
Proceso	
Usuario	Sistema
El usuario pulsa botón de “Registrar” disponible en la ventana de Sociedades Cazadores	
	El sistema muestra el formulario de registro con sus correspondientes campos
El usuario introduce todos los campos obligatorios y pulsa en “Registrar”	
	El sistema comprueba y valida los datos introducidos, los almacena en la base de datos y envía un correo de aviso al email especificado, dando acceso al nuevo usuario como Sociedad de Cazadores, en caso de error se muestra el error correspondiente.

Fuente: Elaboración propia

Tabla 6. Caso uso: Iniciar Sesión

Caso de uso	Iniciar sesión
Precondición	Sociedad de Cazadores registrada previamente
Postcondición	Sociedad de Cazadores accede al sistema (más funcionalidades disponibles) con sus credenciales
Proceso	
Usuario	Sistema
El usuario pulsa el botón de “Iniciar sesión”	
	El sistema muestra el formulario de inicio de sesión (email y contraseña)
El usuario introduce todos los campos requeridos y pulsa en “Iniciar sesión”	
	El sistema comprueba y verifica los datos introducidos y en caso de ser correctos permite el acceso a la sesión como una Sociedad de cazadores, en caso contrario la deniega mostrando el error que corresponda.

Fuente: Elaboración propia

Tabla 7. Caso uso: Buscar lugares en el mapa

Caso de uso	Buscar lugares en el mapa
Precondición	Haber realizado una petición que muestre un mapa
Postcondición	-
Proceso	
Usuario	Sistema
El usuario introduce el nombre del lugar que quiere buscar en la barra de búsqueda y presiona Enter o hace click en la ubicación que le interesa	
	El sistema lleva al usuario al punto correspondiente en el mapa y muestra señalizada la ubicación con un marcador

Fuente: Elaboración propia

Tabla 8. Caso uso: Navegar por el mapa

Caso de uso	Navegar por el mapa
Precondición	Haber realizado una petición que muestre un mapa
Postcondición	-
Proceso	
Usuario	Sistema
El usuario se desplaza arrastrando el cursor usando el mapa	
	El mapa va mostrando los lugares que no eran visibles en el mapa

Fuente: Elaboración propia

Tabla 9. Caso uso: Definir horario actividad, características y área

Caso de uso	Definir horario actividad, características y área
Precondición	Haber iniciado sesión como Sociedad de cazadores
Postcondición	Obtención de las áreas de caza para dicho horario
Proceso	
Usuario	Sistema
La sociedad de cazadores introduce la fecha de inicio y de fin de la actividad de caza	
	El sistema valida y comprueba las fechas y le presenta el formulario para la creación de la actividad con el mapa que contiene todos los

	polígonos que están activos durante algún momento.
--	--

Fuente: Elaboración propia

Tabla 10. Caso uso: Crear actividad de caza

Caso de uso		Crear actividad de caza
Precondición		Sociedad de Cazadores ha accedido con sus credenciales o iniciado sesión y ha definido el horario de la actividad
Postcondición		Creación de una nueva actividad representada por un nuevo polígono que indicará la zona reservada
Proceso		
Usuario	Sistema	
El usuario después de introducir el horario tiene un formulario donde describe la actividad y dibuja el polígono con el área donde se va a operar		
	El sistema comprueba que todos los datos se han introducido correctamente y crea la actividad o muestra el error correspondiente.	

Fuente: Elaboración propia

Tabla 11. Caso uso: Editar perfil sociedad de cazadores

Caso de uso		Editar perfil de sociedad de cazadores
Precondición		Sociedad de Cazadores ha accedido con sus credenciales o iniciado sesión
Postcondición		Queda modificada en la base de datos los registros de la Sociedad de cazadores
Proceso		
Usuario	Sistema	
El usuario accede a su perfil		
	El sistema le muestra el formulario con los campos que puede modificar la sociedad de cazadores	
El usuario rellena el formulario con los cambios que necesite realizar		
	El sistema comprueba y valida los datos de manera en que se realiza la actualización en la base de datos o se muestra el error que corresponda	

Fuente: Elaboración propia

Tabla 12. Caso uso: Ver área de caza

Caso de uso	Ver área de Caza
Precondición	Actividad de caza creada por Sociedad de cazadores con credenciales
Postcondición	-
Proceso	
Usuario	Sistema
La sociedad accede a su cuenta y en la tabla de sus actividades pulsa sobre el botón de "Ver" de la columna polígono	
	El sistema obtiene el polígono asociado a la actividad, y le muestra en una nueva ventana un mapa centrado con el polígono que representa el área de la actividad de caza que estableció al crear la actividad y que desea consultar.

Fuente: Elaboración propia

Tabla 13. Caso uso: Subir informe

Caso de uso	Subir informe
Precondición	Actividad de caza creada por Sociedad de cazadores con credenciales que no tiene asociado ningún informe todavía
Postcondición	Informe queda subido y asociado a una actividad de caza
Proceso	
Usuario	Sistema
La sociedad accede a su cuenta y en la tabla de sus actividades pulsa sobre el botón de "Subir" el archivo	
	El sistema obtiene el archivo, hace las comprobaciones correspondientes y genera los registros en la base de datos de manera que crea el informe y que la actividad correspondiente de caza tiene vinculado ese archivo

Fuente: Elaboración propia

Tabla 14. Caso uso: Descargar informe

Caso de uso	Descargar informe
Precondición	Actividad de caza creada por Sociedad de cazadores con credenciales que tiene ya subido un informe
Postcondición	El informe es obtenido por el usuario

Proceso	
Usuario	Sistema
La sociedad accede a su cuenta y en la tabla de sus actividades pulsa descargar el informe de una actividad	
	El sistema obtiene el archivo, de la base de datos y proporciona el típico panel de descarga para que el usuario lo nombre como quiera y defina la ruta donde se lo descargará
El usuario establece el nombre y ruta y pulsa en Descargar	

Fuente: Elaboración propia

Tabla 15. Caso uso: Consultar información acciones disponibles para cada entidad

Caso de uso	Consultar información acciones disponibles para cada entidad
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y llega a su pantalla inicial o pulsa sobre "Administrador"	
	El sistema le muestra una ventana donde aparecerán todas las entidades o tablas de la base de datos para que pueda ver para que sirve cada una y que acciones puede realizar

Fuente: Elaboración propia

Tabla 16. Caso uso: Listar Sociedades Cazadores

Caso de uso	Listar Sociedades Cazadores
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y pulsa sobre Acciones -> Sociedades Cazadores	
	El sistema le muestra una ventana donde aparecerán todas sociedad creadas hasta ese momento en forma de tabla con los datos correspondientes.

Fuente: Elaboración propia

Tabla 17. Caso uso: Crear usuario sociedad cazadores

Caso de uso		Crear usuario sociedad de cazadores
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión	
Postcondición	Quedará creada una nueva sociedad de cazadores	
Proceso		
Usuario	Sistema	
El usuario pulsa en "Crear sociedad"		
	El sistema le muestra el formulario con todos los datos a introducir que son los mismos que los del registro	
El usuario administrador introduce todos los datos del formulario		
	El sistema valida y comprueba los datos y o bien muestra el error oportuno o crea la nueva sociedad, enviando un mensaje de aviso al email introducido, y devuelve al usuario administrador a la ventana de las sociedades donde debe de aparecer la recién creada	

Fuente: Elaboración propia

Tabla 18. Caso uso: Editar usuario sociedad cazadores

Caso de uso		Editar usuario sociedad cazadores
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión	
Postcondición	Quedará modificada una sociedad de cazadores que ya estaba en la base de datos	
Proceso		
Usuario	Sistema	
El usuario pulsa sobre el botón "Editar" en una fila que representa a una Sociedad ya creada		
	El sistema le muestra el formulario de editar que sería el mismo que el de registro cargando los valores que corresponden para la edición.	
El usuario administrador modifica/introduce todos los datos del formulario		
	El sistema valida y comprueba los datos y o bien muestra el error oportuno o modifica la	

	sociedad, enviando un email de aviso y devolviendo al usuario administrador a la ventana de las sociedades donde debe de aparecer la sociedad ya modificada con los nuevos valores
--	--

Fuente: Elaboración propia

Tabla 19. Caso uso: Eliminar usuario sociedad cazadores

Caso de uso		Eliminar usuario sociedad de cazadores
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión	
Postcondición	Quedará eliminada la sociedad de cazadores que estaba guardada en la base de datos	
Proceso		
Usuario	Sistema	
El usuario pulsa sobre "Eliminar" en una fila que representa a una Sociedad de Cazadores ya creada		
	El sistema elimina la sociedad de la base de datos para la que el administrador ha pulsado el botón y le muestra la tabla actualizada sin que deba ya aparecer ese usuario. Se le manda un email al usuario eliminado avisándole de la acción realizada.	

Fuente: Elaboración propia

Tabla 20. Caso uso: Listar actividades caza

Caso de uso		Listar actividades caza
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión	
Postcondición	-	
Proceso		
Usuario	Sistema	
El usuario ha iniciado sesión y pulsa sobre Acciones -> Actividades Caza		
	El sistema le muestra una ventana donde aparecerán todas sociedad creadas en forma de tabla con los datos correspondientes.	

Fuente: Elaboración propia

Tabla 21. Caso uso: Editar actividades caza

Caso de uso	Editar actividades caza
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y pulsa sobre Acciones -> Actividades Caza	
	El sistema le muestra una ventana donde aparecerán todas sociedad creadas en forma de tabla con los datos correspondientes.
El usuario pulsa el botón de "Editar" sobre la Actividad que desea modificar	
	El sistema le muestra un formulario con la fecha de inicio/fin que estaba establecida para que la modifique el administrador
El administrador introduce la fecha de la actividad a editar	
	El sistema le muestra el formulario con el área a editar así como el resto de los polígonos que aparecen en la nueva fecha introducida y el resto del formulario
El administrador edita el polígono y rellena todos los datos y pulsa sobre el botón de "Editar"	
	El sistema hace las comprobaciones correspondientes y edita la Actividad de caza en la base de datos y manda un aviso a la Sociedad a la que pertenece esa Actividad. Después redirige al Administrador a la ventana de Acciones Actividades Caza

Fuente: Elaboración propia

Tabla 22. Caso uso: Eliminar actividades caza

Caso de uso	Eliminar actividades caza
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y pulsa sobre Acciones -> Actividades Caza	

	El sistema le muestra una ventana donde aparecerán todas sociedad creadas en forma de tabla con los datos correspondientes.
El usuario pulsa el botón de "Eliminar" sobre la Actividad que desea eliminar	
	El sistema elimina en la base de datos esa actividad y envía el aviso a la Sociedad que la creó. Se redirige al administrador a Acciones Actividades Caza donde puede comprobar que ha sido eliminada correctamente

Fuente: Elaboración propia

Tabla 23. Caso uso: Listar coordenadas de Zonas de Caza

Caso de uso	Listar coordenadas de Zonas de Caza
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y pulsa sobre Acciones -> Coordenadas	
	El sistema le muestra una ventana donde aparecerán todas coordenadas creadas en forma de tabla asociada cada una a un polígono.

Fuente: Elaboración propia

Tabla 24. Caso uso: Listar informes

Caso de uso	Listar Informes
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El usuario ha iniciado sesión y pulsa sobre Acciones -> Informes	
	El sistema le muestra una ventana donde aparecerán todos los informes creados en forma de tabla asociado cada uno a una actividad.

Fuente: Elaboración propia

Tabla 25. Caso uso: Descargar Informe

Caso de uso	Descargar Informe
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El administrador pulsa en el botón de “Descargar” asociado a una fila que representa un informe	
	El sistema recupera el archivo y le muestra la típica ventana de descarga donde establecer el nombre
El usuario establece el nombre del archivo y pulsa “Descargar”	

Fuente: Elaboración propia

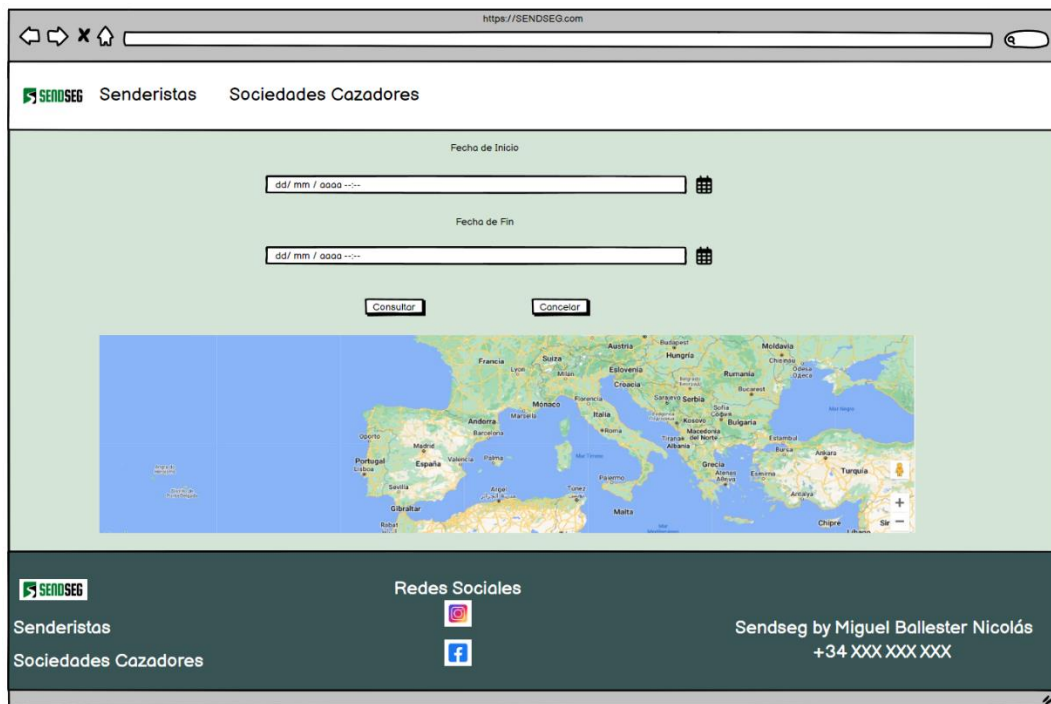
Tabla 26. Caso uso: Eliminar informe

Caso de uso	Eliminar Informe
Precondición	El usuario tiene credenciales de administrador en el sistema después de haber iniciado sesión
Postcondición	-
Proceso	
Usuario	Sistema
El administrador pulsa en el botón de “Borrar” asociado a una fila que representa un informe	
	El sistema elimina en la base de datos el registro del informe asociado a una actividad para que se pueda posteriormente subir uno nuevo. Se le envía un aviso a la Sociedad que creó la actividad y subió el informe.

Fuente: Elaboración propia

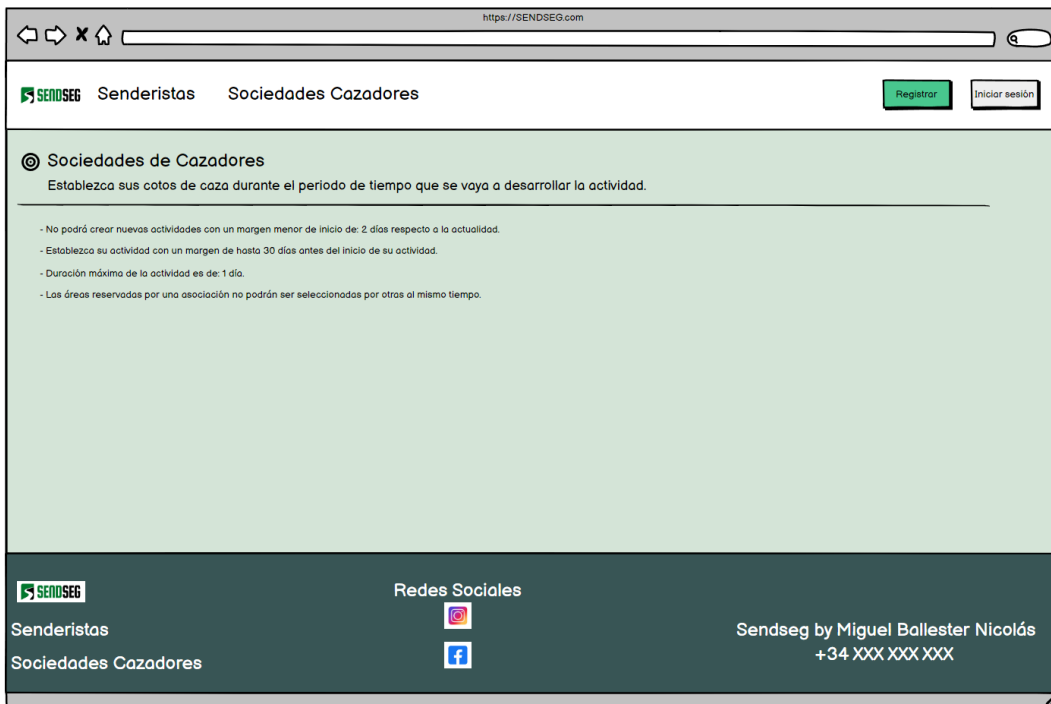
Anexo 2: Mockups Prototipado

Ilustración 45. Prototipo ventana consulta zonas caza Senderistas



Fuente: Elaboración propia

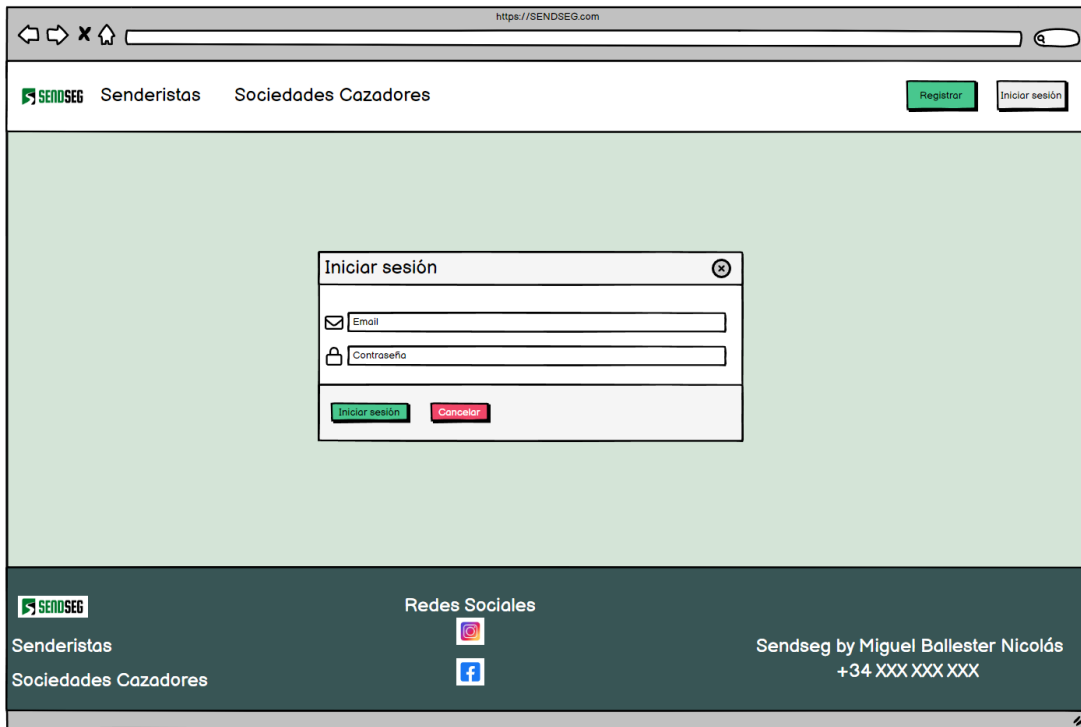
Ilustración 46. Prototipo ventana Sociedades sin credenciales



Fuente: Elaboración propia

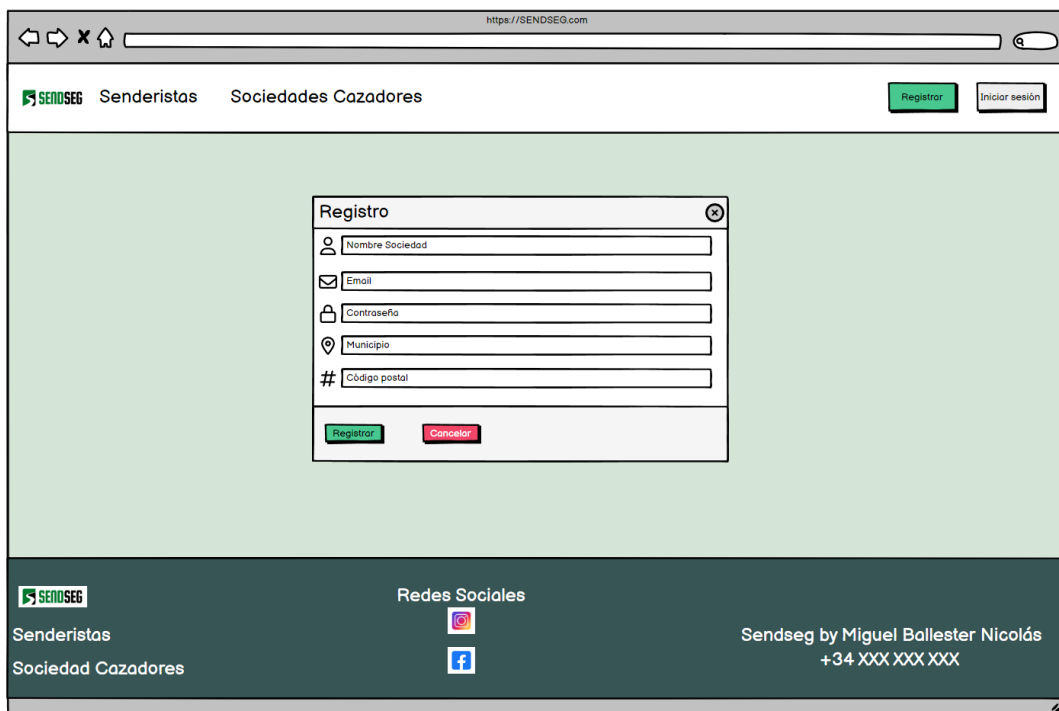
Respecto al inicio de sesión y registro, en principio se intentará hacer uso de modals y no generar ventanas específicas para estas acciones.

Ilustración 47. Prototipo ventana Iniciar sesión



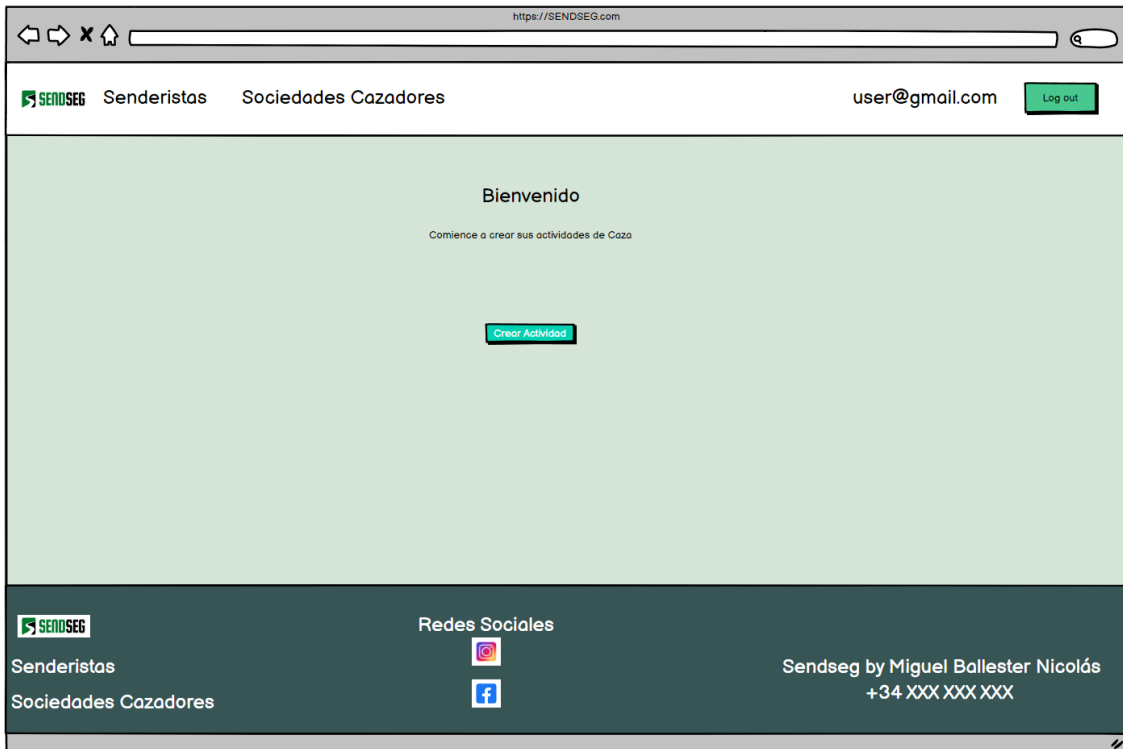
Fuente: Elaboración propia

Ilustración 48. Prototipo ventana Registro sociedades



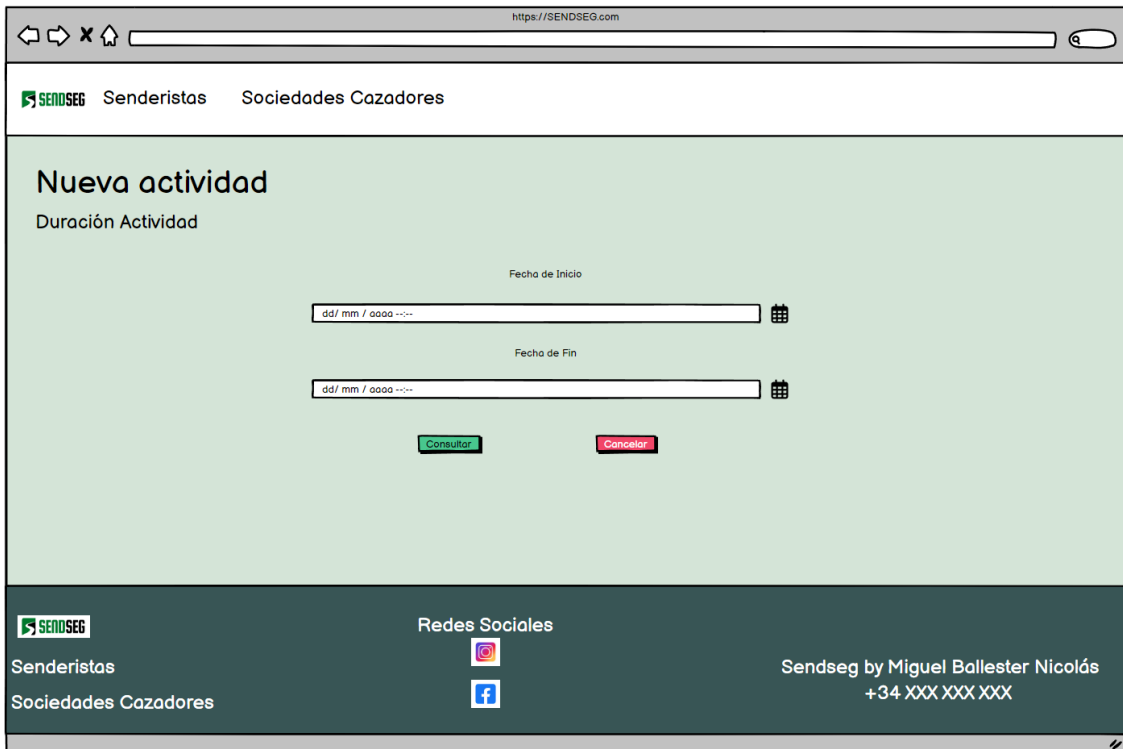
Fuente: Elaboración propia

Ilustración 49. Prototipo ventana Inicio sesión Sociedad sin actividades creadas



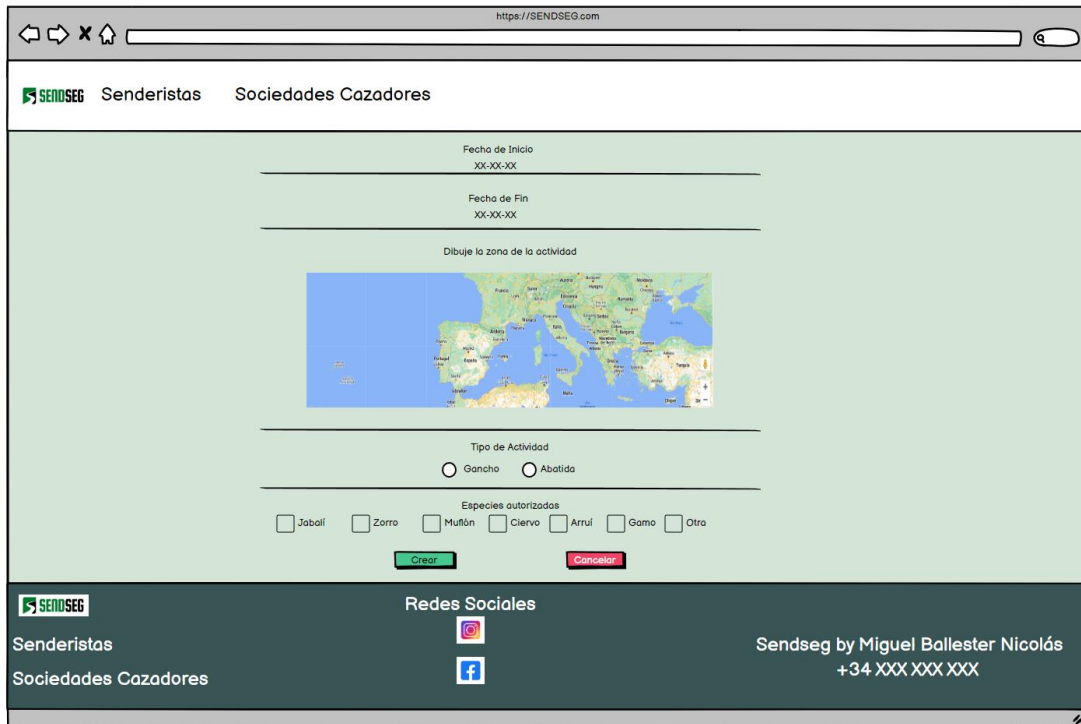
Fuente: Elaboración propia

Ilustración 50. Prototipo ventana selección inicio y fin de actividad de caza



Fuente: Elaboración propia

Ilustración 51. Prototipo ventana creación actividad



Fuente: Elaboración propia

Ilustración 52. Prototipo ventana Sociedad con actividades caza creadas



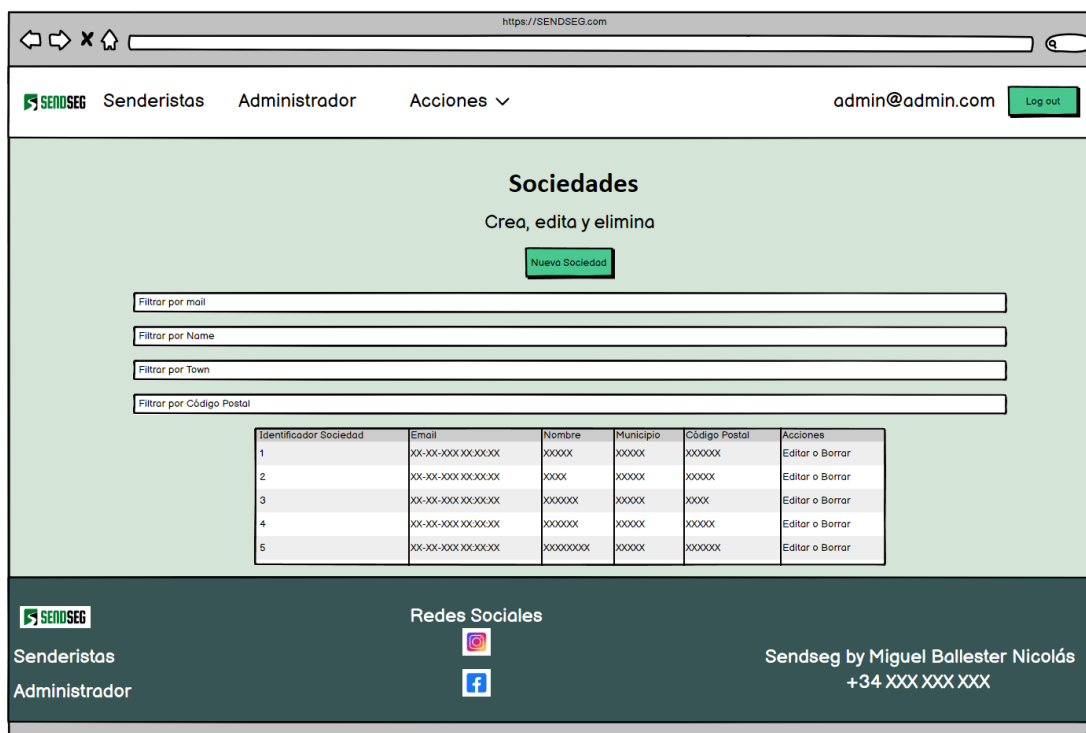
Fuente: Elaboración propia

Ilustración 53. Prototipo ventana Inicio Administrador



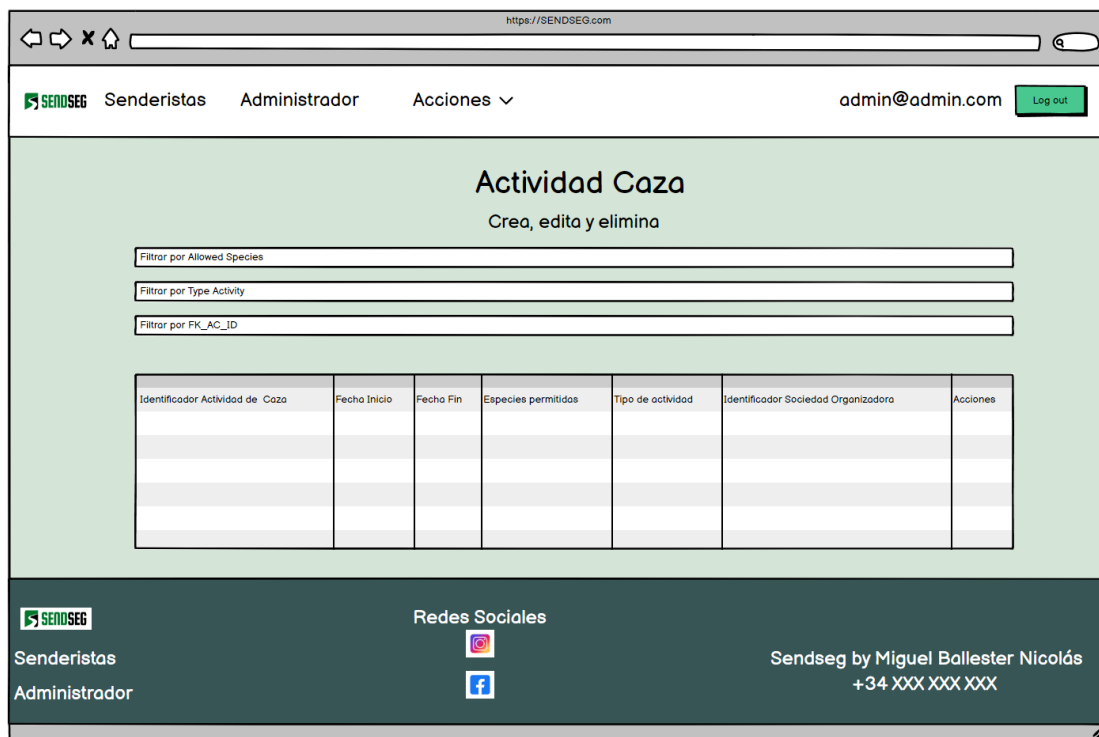
Fuente: Elaboración propia

Ilustración 54. Prototipo ventana Administrador gestión sociedades



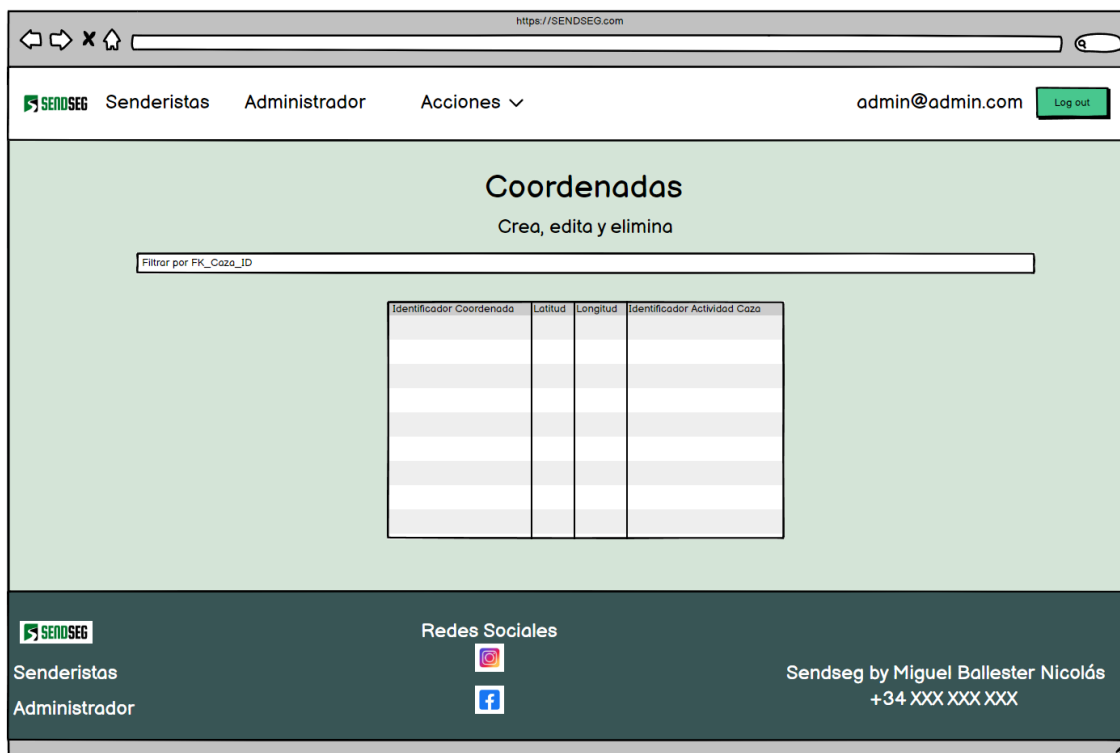
Fuente: Elaboración propia

Ilustración 55. Prototipo ventana Administrador gestión actividades caza



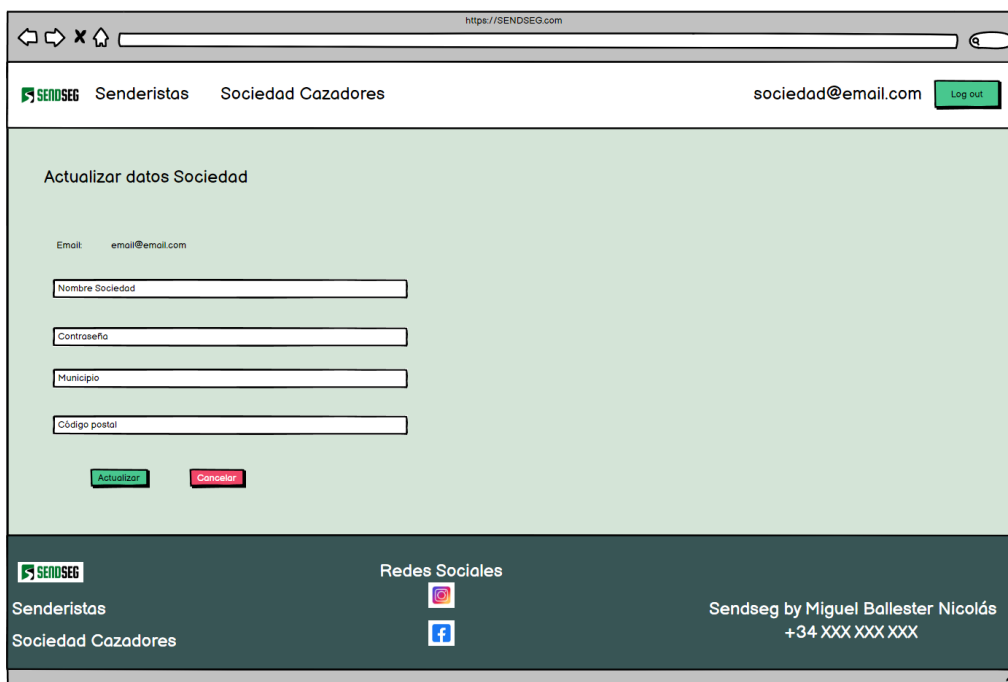
Fuente: Elaboración propia

Ilustración 56. Prototipo ventanas Administrador gestión coordenadas



Fuente: Elaboración propia

Ilustración 57. Prototipo ventana Administrador edición de datos de Sociedad



Fuente: Elaboración propia

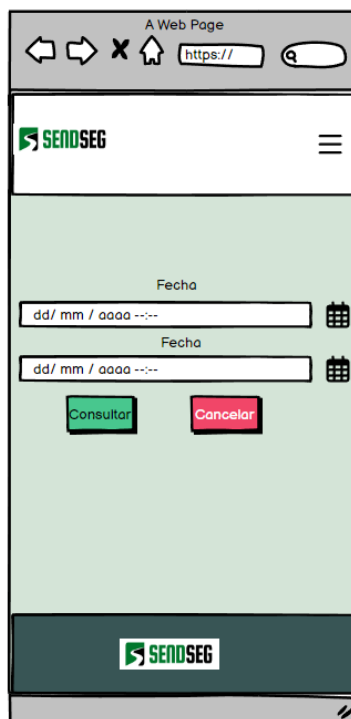
Una vez expuestas todas las imágenes de los prototipos de las ventanas realizadas, se va a mostrar como alguna de las anteriores debería de redimensionar, cumpliendo con la cualidad de la responsiveness, de cada uno de sus elementos en caso de que se acceda a Sendseg desde un dispositivo cuya resolución pueda inferior.

Ilustración 58. Prototipo ventana responsive Inicio Sociedad



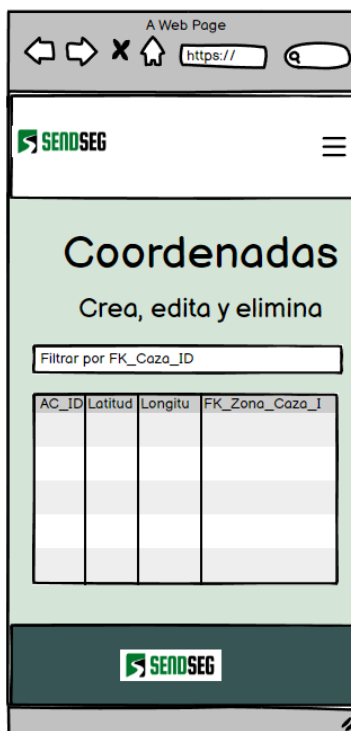
Fuente: Elaboración propia

Ilustración 59. Prototipo ventana responsive consulta senderistas



Fuente: Elaboración propia

Ilustración 60. Prototipo ventana responsive Administrador gestión coordenadas



Fuente: Elaboración propia

Anexo 3: Script base de datos

```
CREATE DATABASE `sendseg` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
```

```
CREATE TABLE `asociacion_cazadores` (
  `AC_ID` int NOT NULL AUTO_INCREMENT,
  `Email` varchar(45) NOT NULL,
  `Password` varchar(60) NOT NULL,
  `Name` varchar(45) NOT NULL,
  `Town` varchar(45) NOT NULL,
  `PostalCode` varchar(45) NOT NULL,
  PRIMARY KEY (`AC_ID`),
  UNIQUE KEY `AC_ID_UNIQUE` (`AC_ID`),
  UNIQUE KEY `Email_UNIQUE` (`Email`)
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `zona_de_caza` (
  `Zona_Caza_ID` int NOT NULL AUTO_INCREMENT,
  `Initial_Date` datetime NOT NULL,
  `End_Date` datetime NOT NULL,
  `Allowed_Species` varchar(70) NOT NULL,
  `Type_Activity` varchar(45) NOT NULL,
  `FK_AC_ID` int NOT NULL,
  PRIMARY KEY (`Zona_Caza_ID`),
  UNIQUE KEY `Zona_Caza_ID_UNIQUE` (`Zona_Caza_ID`),
  KEY `AC_ID_idx` (`FK_AC_ID`),
  CONSTRAINT `AC_ID` FOREIGN KEY (`FK_AC_ID`) REFERENCES `asociacion_cazadores`
(`AC_ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=76 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

```

CREATE TABLE `coordenadas` (
  `Coordenadas_ID` int NOT NULL AUTO_INCREMENT,
  `latitud` float NOT NULL,
  `longitud` float NOT NULL,
  `FK_Zona_Caza_ID` int NOT NULL,
  PRIMARY KEY (`Coordenadas_ID`),
  UNIQUE KEY `Coordenadas_ID_UNIQUE` (`Coordenadas_ID`),
  KEY `FK_Zona_Caza_idx` (`FK_Zona_Caza_ID`),
  CONSTRAINT `FK_Zona_Caza` FOREIGN KEY (`FK_Zona_Caza_ID`) REFERENCES
`zona_de_caza` (`Zona_Caza_ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=236 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `informe_actividad` (
  `Informe_ID` int NOT NULL AUTO_INCREMENT,
  `archivo` longblob NOT NULL,
  `FK_Zona_de_Caza_ID` int NOT NULL,
  PRIMARY KEY (`Informe_ID`),
  UNIQUE KEY `Informe_ID_UNIQUE` (`Informe_ID`),
  KEY `FK_Zona_de_Caza_ID_idx` (`FK_Zona_de_Caza_ID`),
  CONSTRAINT `FK_Zona_de_Caza_ID` FOREIGN KEY (`FK_Zona_de_Caza_ID`)
REFERENCES `zona_de_caza` (`Zona_Caza_ID`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=38 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

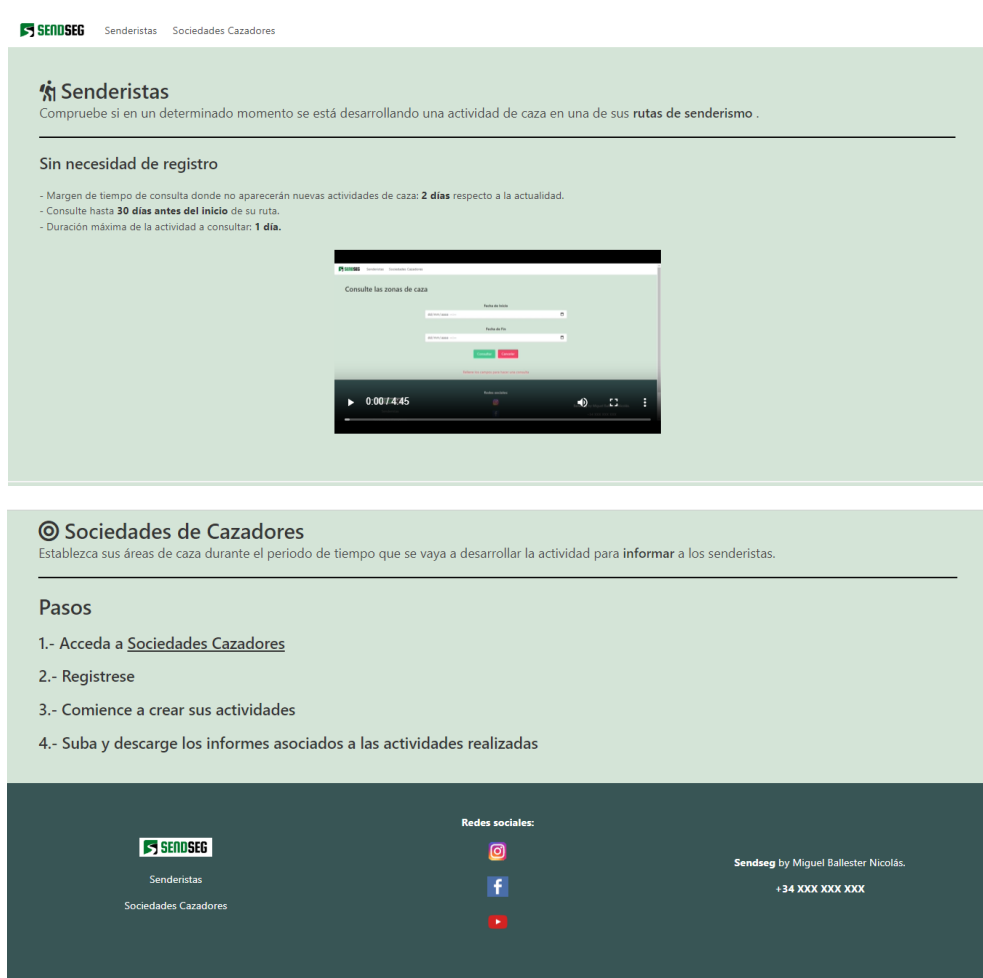
```

Anexo 4: Manual de usuario

En este manual de usuario, se van a mostrar las distintas funciones que se pueden desarrollar en cada una de las pantallas que se han desarrollado y así se verá cómo se puede realizar todas y cada una de las funcionalidades que cada uno de los tipos de usuarios puede hacer.

La página de Inicio de Sendseg sirve para informar a Senderistas y a Sociedades de Cazadores sobre el sistema y su uso, en ella además de texto descriptivo se ha incluido un video explicativo para que los senderistas puedan ver el funcionamiento de las consultas. El resultado de la vista de Inicio es:

Ilustración 61. Vista Inicio Sendseg



Fuente: Elaboración propia

Los senderistas pueden realizar las consultas de las áreas afectadas por actividades de caza durante el periodo de tiempo en el que deseen realizar una actividad. Si en la ventana anterior pulsan sobre "Senderistas", pueden acceder a ella y es la siguiente:

Ilustración 62. Vista Senderistas

SENDSEG Senderistas Sociedades Cazadores

Consulte las zonas de caza

Fecha de Inicio
dd/mm/aaaa --:--

Fecha de Fin
dd/mm/aaaa --:--

Consultar Cancelar

Rellene los campos para hacer una consulta

Redes sociales:
SENDSEG
Sendseg by Miguel Ballester Nicolás.
+34 XXX XXX XXX

Fuente: Elaboración propia

Para hacer la consulta se muestra este formulario donde se introduce la fecha de inicio y fin de actividad, que deben de cumplir una serie de condiciones para que Sendseg permita la consulta, al introducir estos valores en caso de error se indica cuál se ha cometido, y en caso contrario, se carga un mapa con las áreas de las actividades representadas por polígonos dibujados de color rojo. Puede verse ambos casos en las siguientes capturas de manera respectiva:

Ilustración 63. Vista error consulta senderistas

Consulte las zonas de caza

Fecha de Inicio
dd/mm/aaaa --:--

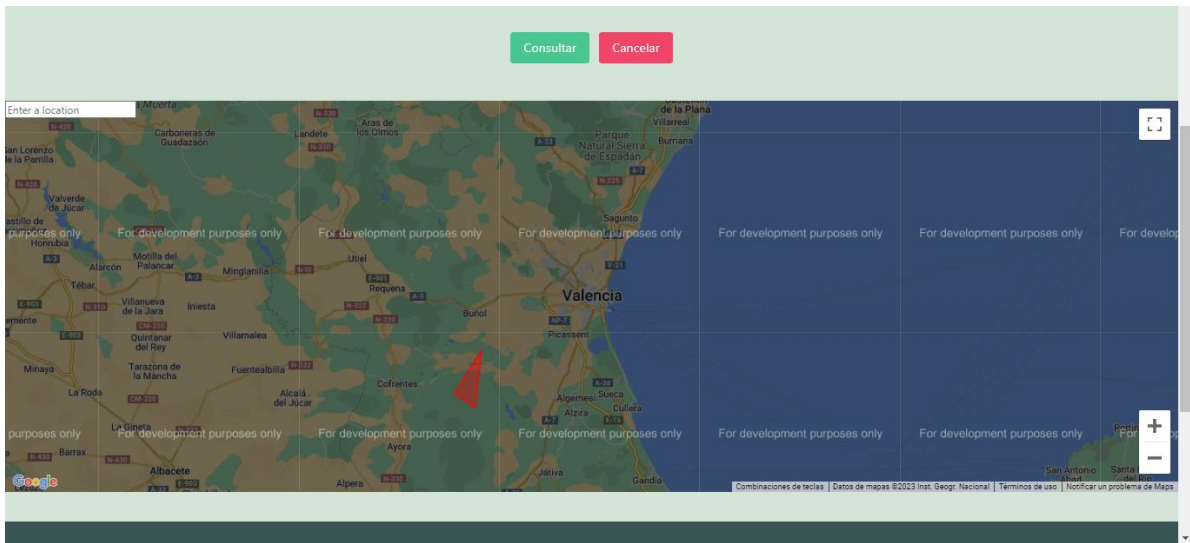
Fecha de Fin
dd/mm/aaaa --:--

Consultar Cancelar

Introduzca Fecha de Inicio y de Fin

Fuente: Elaboración propia

Ilustración 64. Vista consulta senderistas correcta



Fuente: Elaboración propia

Cabe indicar al lector de esta memoria que en el momento en que se toman estas capturas la duración del plan gratuito de la API de Google Maps JavaScript ha expirado y pese a que se puede seguir usando con normalidad en el sistema, se muestra la marca de agua de que es para fines o con propósitos de desarrollo. Con lo cual, cuando se acceda a un plan de pago estas marcas serán eliminadas y tendremos el mapa visible a la perfección sin ningún tipo de problema.

Mostrado lo anterior, si desde la ventana de Inicio pulsamos sobre “Sociedades Cazadores vemos la siguiente ventana explicativa de manera complementaria para este tipo de usuarios sobre el sistema:

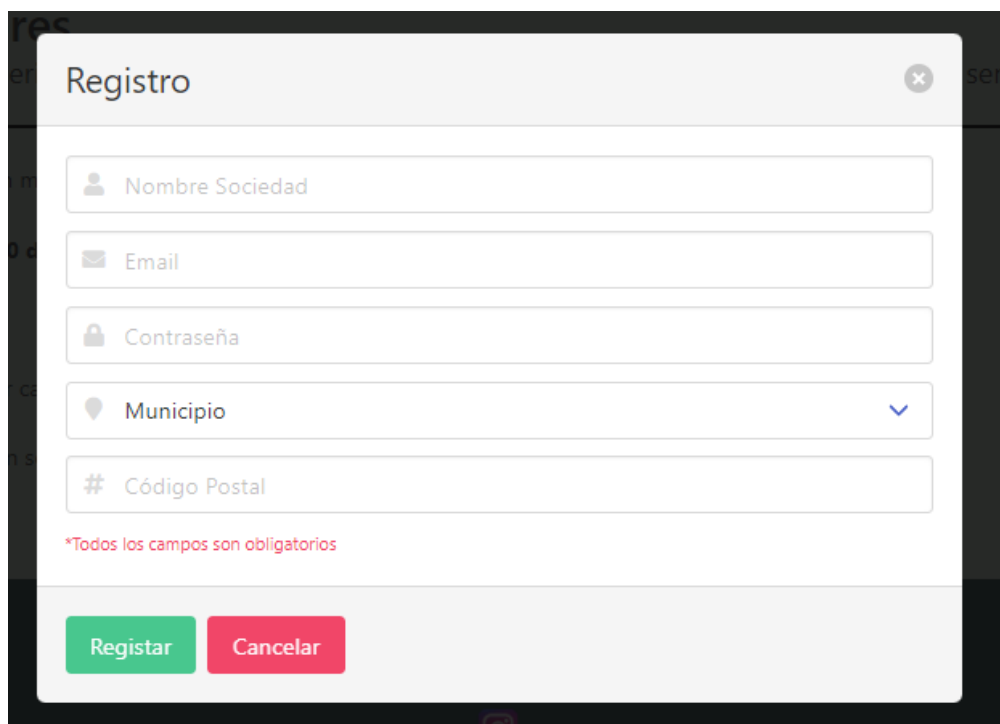
Ilustración 65. Vista Sociedades Cazadores



Fuente: Elaboración propia

Ahora pasamos a mostrar dentro de esta ventana el modal de Registro y de Inicio de sesión para aquellos usuarios que deseen adquirir o identificarse como Sociedades de Cazadores:

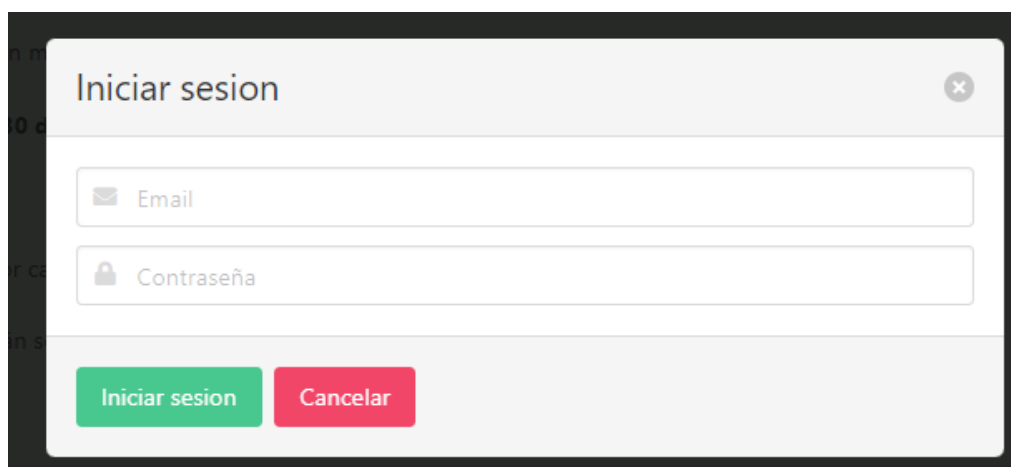
Ilustración 66. Vista Registro



The image shows a mobile application modal titled "Registro". It contains five input fields: "Nombre Sociedad" (with a person icon), "Email" (with an envelope icon), "Contraseña" (with a lock icon), "Municipio" (with a location pin icon and a dropdown arrow), and "Código Postal" (with a hash icon). Below the fields is a red asterisk warning: "*Todos los campos son obligatorios". At the bottom are two buttons: a green "Registrar" button and a red "Cancelar" button.

Fuente: Elaboración propia

Ilustración 67. Vista Inicio sesión



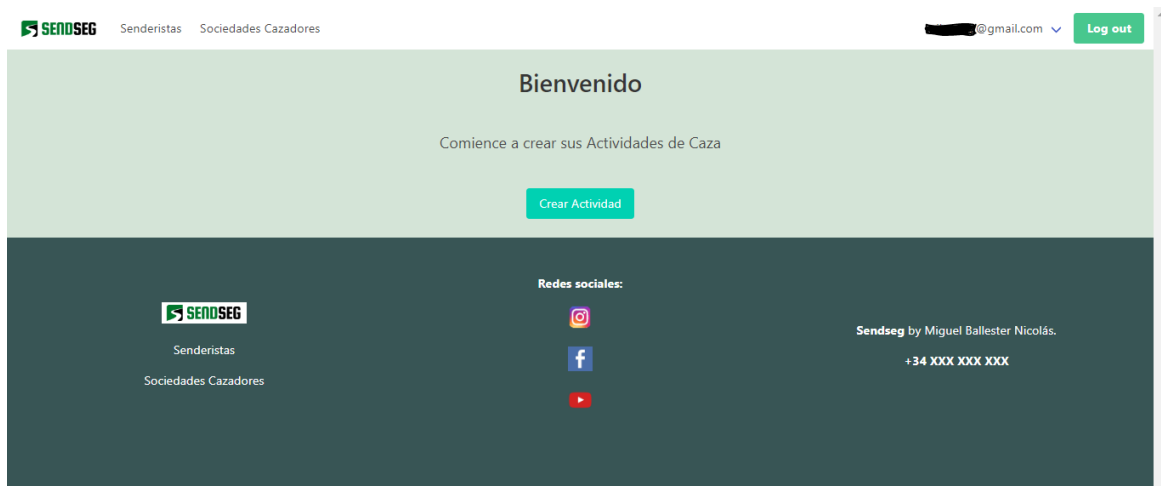
The image shows a mobile application modal titled "Iniciar sesion". It contains two input fields: "Email" (with an envelope icon) and "Contraseña" (with a lock icon). At the bottom are two buttons: a green "Iniciar sesion" button and a red "Cancelar" button.

Fuente: Elaboración propia

Estas son todas las vistas y acciones que pueden realizar los usuarios sin credenciales en el sistema de Sendseg que se corresponden como Senderistas. Cuando realicen correctamente una de las dos anteriores acciones, en caso de registro, se envía un correo al email introducido por el nuevo usuario, y en ambos casos se da acceso a una nueva vista donde la sociedad de cazadores identificada puede ver la lista con las actividades de caza que haya creado.

La vista cuando un usuario no tenga ninguna actividad creada todavía es:

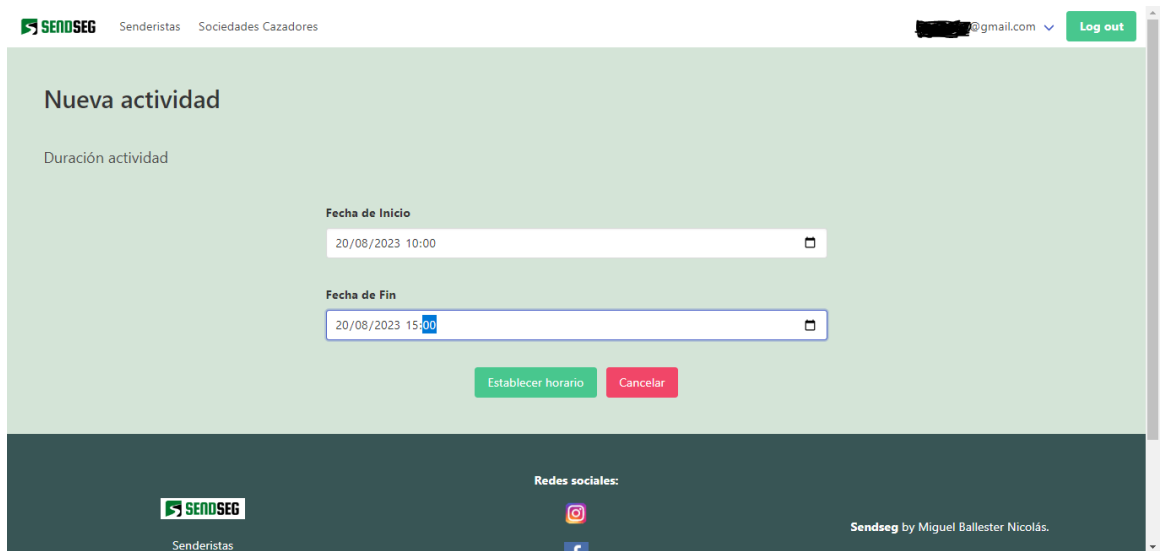
Ilustración 68. Vista sociedad cazadores con sesión iniciada sin actividades



Fuente: Elaboración propia

Pulsando en el botón “Crear actividad” se lleva al usuario a la vista en la que definirá el horario de la actividad que va a crear. Véase en la siguiente vista:

Ilustración 69. Vista creación actividad sociedad cazadores I



Fuente: Elaboración propia

Una vez se establece el horario, se muestra el formulario para terminar la definición de la actividad, mostrando las otras actividades que suceden durante cualquier momento del horario de la actividad a crear, esto se hace en la siguiente vista:

Ilustración 70. Vista creación actividad sociedad cazadores II

SENDSEG Senderistas Sociedades Cazadores

Crear actividad

Fecha de Inicio
20/08/2023 10:00

Fecha de Fin
20/08/2023 15:00

Dibuje la zona de actividad

Enter a location

Mantén pulsada la tecla Ctrl mientras te desplazas para acercar o alejar el mapa

Tipo de Actividad
 Gancho Abatida

Especies autorizadas:
 Jabalí Zorro Mufón Ciervo Arruf Gamo

Crear actividad Cancelar

Redes sociales:
SENDSEG

Fuente: Elaboración propia

En el mapa cargado aparecerán los polígonos que coinciden temporalmente, también se puede observar una barra de búsqueda que permite buscar lugares y haciendo click sobre el sitio te lleva a dicha ubicación. Una vez definida la actividad, se redirige a la vista con la lista de actividades donde podemos observar los datos introducidos.

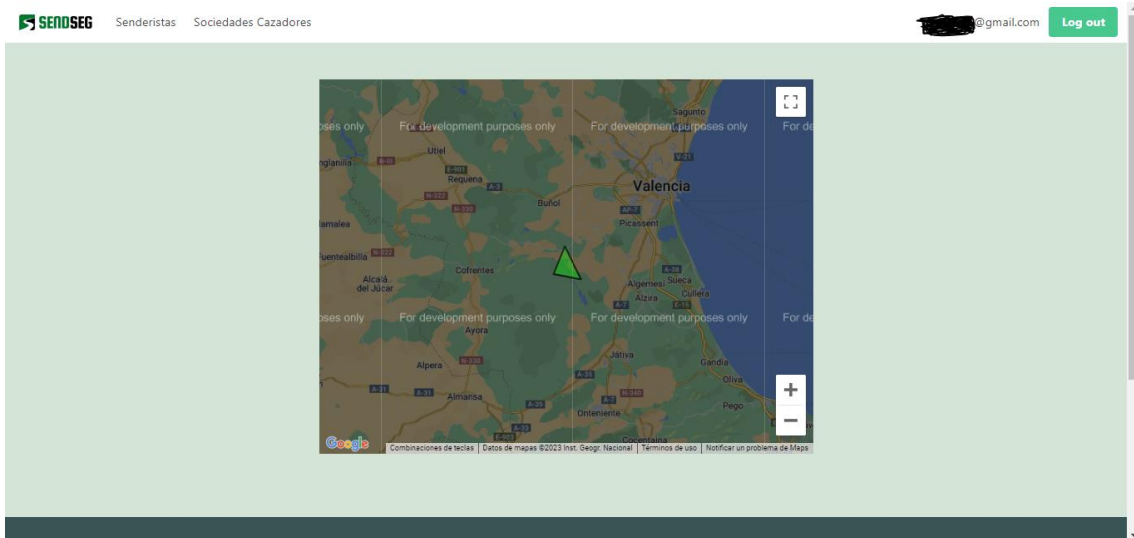
Ilustración 71. Vista sociedad cazadores con actividad creada



Fuente: Elaboración propia

Además dentro de cada actividad, podemos pulsar sobre “Ver” para observar el área dibujada donde se desarrollará esta para asegurarnos que se dibujó de manera correcta o simplemente para recordarla o consultarla:

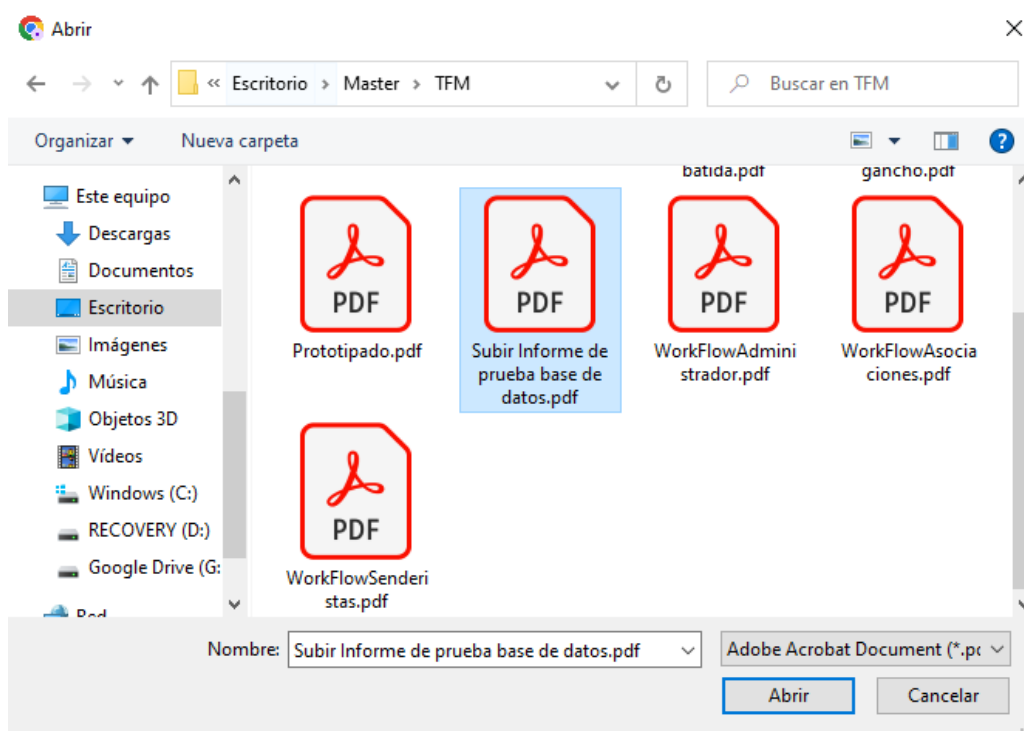
Ilustración 72. Consulta del polígono o área de una actividad de sociedad de cazadores



Fuente: Elaboración propia

La otra acción que pueden hacer es subir un informe asociado que será un archivo pdf, primero se adjunta un documento de esta extensión:

Ilustración 73. Subida informe de una actividad I



Fuente: Elaboración propia

Al pulsar abrir se carga en la vista el documento a subir:

Ilustración 74. Subida de informe de una actividad II

Identificador Actividad Caza	Fecha Inicio	Fecha Fin	Especies Permitidas	Tipo Actividad	Poligono	Informe Actividad
78	20/8/2023, 10:00:00	20/8/2023, 15:00:00	Ciervo	Gancho	Ver	<div style="display: flex; justify-content: space-between; align-items: center;"> Seleccione informe Subir Informe de prueba base... </div> <div style="text-align: right; margin-top: 5px;"> Subir </div>
Identificador Actividad Caza	Fecha Inicio	Fecha Fin	Especies Permitidas	Tipo Actividad	Poligono	Informe Actividad

Fuente: Elaboración propia

En caso de equivocación, el usuario puede seleccionar otro archivo, y cuando el usuario pulsa sobre subir, se sube el último adjunto, de forma en que una vez subido la actividad quedará así:

Ilustración 75. Informe de actividad subido y posibilidad de descarga

Identificador Actividad Caza	Fecha Inicio	Fecha Fin	Especies Permitidas	Tipo Actividad	Poligono	Informe Actividad
78	20/8/2023, 10:00:00	20/8/2023, 15:00:00	Ciervo	Gancho	Ver	<div style="text-align: right; margin-top: 5px;"> Descargar </div>
Identificador Actividad Caza	Fecha Inicio	Fecha Fin	Especies Permitidas	Tipo Actividad	Poligono	Informe Actividad

Fuente: Elaboración propia

La sociedad de cazadores podrá pulsando sobre “Descargar” obtener el informe que subió para su consulta o cualquier otra necesidad.

Respecto al usuario Sociedad de cazadores estas son todas las acciones que puede realizar relacionadas con las actividades de caza, adicionalmente si pasa el cursor por encima de donde está el email, puede pulsar sobre perfil:

Ilustración 76. Posibilidad de modificar el perfil de sociedad cazadores

The screenshot shows the SENOSEG web application interface. At the top, there is a navigation bar with the SENOSEG logo, the text "Senderistas Sociedades Cazadores", and a user profile dropdown menu showing an email address and a "Log out" button. Below the navigation bar, there is a "Crear Actividad" button. The main content area is titled "Actividades de caza creadas anteriormente" and contains a table with the following data:

Identificador Actividad Caza	Fecha Inicio	Fecha Fin	Especies Permitidas	Tipo Actividad	Poligono	Informe Actividad
78	20/8/2023, 10:00:00	20/8/2023, 15:00:00	Ciervo	Gancho	Ver	Descargar

Below the table, there is a footer section with the SENOSEG logo, social media icons for Instagram, Facebook, and YouTube, and contact information: "Sendseg by Miguel Ballester Nicolás. +34 XXX XXX XXX".

Fuente: Elaboración propia

Al pulsar “Perfil” se le redirige a la siguiente vista:

Ilustración 77. Modificación del perfil.

The screenshot shows the "Modificar datos Sociedad" form in the SENOSEG web application. The form is titled "Modificar datos Sociedad" and includes a subtitle: "Consulte y modifique cualquier dato a excepción del Email que deberá ser cambiado por el administrador." The form fields are:

- Email: [Redacted]@gmail.com
- Nombre: mibani
- Contraseña: [Redacted]
- Municipio: Abegondo (dropdown menu)
- Postal Code: 12345

At the bottom of the form, there are two buttons: "Actualizar" (green) and "Cancelar" (red).

Fuente: Elaboración propia

En esta, la sociedad de cazadores puede modificar todos sus datos con excepción del email, que deberá ser modificado por el administrador. Por último, pulsando sobre el botón de “Log out” el usuario podrá cerrar su sesión y volver a ser considerado un usuario senderista sin credenciales para acceder a lo visto ahora.

Para terminar con este tipo de usuario, ahora se va a mostrar en caso de crear la actividad y que coincida con alguna otra temporalmente se mostrará su polígono o área, con el que se comprobará la intersección respecto al nuevo que se vaya a crear, se puede observar en la siguiente captura:

Ilustración 78. Creación de actividad con la aparición de otras definidas previamente



Fuente: Elaboración propia

Todo lo anterior es lo que puede hacer una sociedad de cazadores que inicia sesión, pero nos falta todas las vistas y acciones que puede realizar un usuario que sea considerado como administrador. Cuando este inicia sesión se le muestra la siguiente vista:

Ilustración 79. Vista del Administrador para comprobar las entidades y acciones que se pueden hacer

The screenshot shows the SENDSEG administrator interface. At the top, there is a navigation bar with 'Senderistas', 'Administrador', and 'Acciones' (with a dropdown arrow). The user is logged in as 'admin@admin.com' with a 'Log out' button. The main content area displays a green banner with the text 'Bienvenido administrador' and 'Trabaje con la base de datos usando las acciones disponibles'. Below this is a table with the following structure:

Tabla	Descripción	Acciones disponibles			
Sociedades Cazadores	Accede a las asociaciones. Los cambios como eliminación harán que las filas de otras tablas que dependen de esta se actualicen en cascada en caso de ser necesario.	Listar	Crear	Actualizar	Borrar
Actividades de Caza	Accede a las actividades de caza. Consulta, edita y elimina actividades de caza.	Listar		Actualizar	Borrar
Coordenadas	Accede a las coordenadas de caza para cada actividad. No se realizan nuevas inserciones, edición o modificación de coordenadas de manera aislada, se hace desde la tabla Zona de Caza.	Listar			
Informes	Accede a los informes subidos por las sociedades de caza asociados a las actividades que han sido realizadas. Se proporciona la posibilidad de listar todos los documentos, descargarlos y borrarlos.	Listar		Descargar	Borrar
Tabla	Descripción	Acciones disponibles			

At the bottom of the page, there is a dark green footer with the SENDSEG logo and the text 'Senderistas'.

Fuente: Elaboración propia

En esta vista puede observar las distintas entidades o recursos para ver una descripción y que acciones puede realizar sobre cada uno de estos. Después, pasando el cursor por encima de "Acciones" se le muestran las mismas entidades para que indique sobre cuál quiere operar pulsando sobre esta:

Ilustración 80. Despliegue de Acciones con cada recurso

The screenshot shows the SENDSEG administrator interface with the 'Acciones' dropdown menu open. The menu lists 'Sociedades Cazadores', 'Actividades de Caza', 'Coordenadas', and 'Informes'. The main content area displays a green banner with the text 'Bienvenido administrador' and 'Trabaje con la base de datos usando las acciones disponibles'. Below this is a table with the following structure:

Tabla	Descripción	Acciones disponibles			
Sociedades Cazadores	Accede a las asociaciones. Los cambios como eliminación harán que las filas de otras tablas que dependen de esta se actualicen en cascada en caso de ser necesario.	Listar	Crear	Actualizar	Borrar
Actividades de Caza	Accede a las actividades de caza. Consulta, edita y elimina actividades de caza.	Listar		Actualizar	Borrar
Coordenadas	Accede a las coordenadas de caza para cada actividad. No se realizan nuevas inserciones, edición o modificación de coordenadas de manera aislada, se hace desde la tabla Zona de Caza.	Listar			
Informes	Accede a los informes subidos por las sociedades de caza asociados a las actividades que han sido realizadas. Se proporciona la posibilidad de listar todos los documentos, descargarlos y borrarlos.	Listar		Descargar	Borrar
Tabla	Descripción	Acciones disponibles			

At the bottom of the page, there is a dark green footer with the SENDSEG logo and the text 'Senderistas'.

Fuente: Elaboración propia

Ahora, vamos a ver las acciones que puede realizar sobre cada una en orden. Al pulsar sobre "Sociedades Cazadores" ve lo siguiente:

Ilustración 81. Vista de Acciones sobre Sociedades Cazadores del Administrador

The screenshot shows the 'Sociedades Cazadores' admin interface. At the top, there is a navigation bar with the SENDSEG logo, 'Senderistas', 'Administrador', and 'Acciones' dropdown. The user 'admin@admin.com' is logged in, with a 'Log out' button. The main heading is 'Sociedades Cazadores' with the subtext 'Crea, edita y elimina'. A green 'Nueva Sociedad' button is prominently displayed. Below this are four search filters: 'Filtrar por Email', 'Filtrar por Nombre', 'Filtrar por Municipio', and 'Filtrar por Código Postal'. A table lists four existing societies with their details and edit/delete actions.

Identificador Sociedad	Email	Nombre	Municipio	Código Postal	Acciones
2	user1@gmail.com	user1	Torrent	46901	Editar Borrar
3	user2@gmail.com	user2	Alaquàs	46970	Editar Borrar
4	user3@gmail.com	user3	Alaquàs	46970	Editar Borrar
5	user4@gmail.com	user4	Torrent	46900	Editar Borrar

Fuente: Elaboración propia

El administrador observa una tabla con todas las Sociedades de Cazadores creadas y puede filtrar por valores en la tabla usando los distintos buscadores. Además, puede crear nuevas al pulsar sobre el botón “Nueva Sociedad”:

Ilustración 82. Vista Administrador para crear nueva sociedad cazadores

The screenshot shows the 'Introduzca datos de nueva Sociedad' form. It includes input fields for 'Email', 'Nombre', 'Password', 'Municipio' (a dropdown menu), and 'Postal Code'. At the bottom, there are two buttons: 'Crear Sociedad' (green) and 'Cancelar' (red).

Fuente: Elaboración propia

Y luego si pulsa sobre el botón “Editar” asociado a una sociedad en la tabla puede modificar todos sus datos incluido el email:

Ilustración 83. Vista modificación de una sociedad de cazadores por Administrador

The screenshot shows a web application interface for editing a society's data. At the top, there is a navigation bar with the SENDSEG logo, the user role 'Administrador', and a 'Log out' button. The main heading is 'Edición de datos Sociedad'. The form contains the following fields: 'Email' (with a masked address), 'Nombre' (containing 'mibani'), 'Password' (empty), 'Municipio' (a dropdown menu with 'Abegondo' selected), and 'Postal Code' (containing '12345'). At the bottom of the form are two buttons: 'Actualizar' (green) and 'Cancelar' (red). The browser's address bar shows 'localhost:3000'.

Fuente: Elaboración propia

En caso de pulsar sobre el botón de “Eliminar” esa sociedad se eliminará y en cascada se eliminan todas sus actividades creadas, así como los vértices de cada polígono asociado a una actividad e informes.

Se debe de destacar el hecho de que las acciones realizadas sobre sociedades de cazadores por el administrador en el momento en que se realizan se envía un correo al email de la sociedad para que este esté informado de que acciones se realiza por alguien que no es él mismo. Por tanto, estas son todas las acciones que puede realizar el administrador respecto a las sociedades, ahora pasamos a la siguiente entidad. Pulsamos en “Actividad Caza”. Al administrador se le muestra la siguiente vista:

Ilustración 84. Vista Acciones de Actividades Caza por el Administrador

Identificador Actividad	Fecha Inicio	Fecha Fin	Especies permitidas	Tipo Actividad	Identificador Sociedad	Acciones
1	Sun May 14 2023 19:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 22:00:00 GMT+0200 (hora de verano de Europa central)	Zorro	Gancho	2	Editar Borrar
2	Sun May 14 2023 20:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Murfón	Gancho	3	Editar Borrar
3	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Jabalí Conejo	Abatida	2	Editar Borrar
4	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Jabalí	Abatida	4	Editar Borrar
5	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Amulí	Abatida	5	Editar Borrar
6	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Amulí	Abatida	5	Editar Borrar
7	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Ciervo	Abatida	4	Editar Borrar
8	Sun May 14 2023 21:00:00 GMT+0200 (hora de verano de Europa central)	Sun May 14 2023 23:00:00 GMT+0200 (hora de verano de Europa central)	Ciervo Amulí Conejo	Abatida	5	Editar Borrar

Fuente: Elaboración propia

Respecto a esta entidad, el administrador puede observar en una tabla todas las actividades creadas, para filtrar y buscar actividades concretas puede usar las barras de búsqueda asociada a distintos campos para la búsqueda más rápida.

El administrador no puede crear actividades, pero pulsando sobre “Editar” puede modificar una ya definida. La modificación, al igual que la creación se divide en dos pasos, la definición del horario y rellenar el resto de los datos incluida el área donde se desarrollará.

Ilustración 85. Vista edición actividad de caza por Administrador I

Fuente: Elaboración propia

Cuando se muestra el mapa, se calcula el punto central del polígono o área de la actividad y el mapa se centra en dicha posición de modo en que el administrador no debe preocuparse por buscarlo y simplemente puede ya cambiar su posición, añadir, eliminar o modificar vértices.

Ilustración 86. Vista edición actividad de caza por Administrador II

SENDISEG Senderistas Administrador Acciones admin@admin.com Log out

Editar actividad

ID Actividad Caza editando
78

Fecha de Inicio
20/08/2023 10:00

Fecha de Fin
20/08/2023 15:00

Dibuje la zona de actividad

Enter a location

Tipo de Actividad
 Gancho Abatida

Especies autorizadas:
 Jabalí Zorro Mufi6n Ciervo Arruf Gamo

Confirmar Edici6n Cancelar

Fuente: Elaboraci6n propia

Al pulsar sobre "Confirmar Edici6n" se modifica la actividad y se manda un aviso a la sociedad que cre6 esta, en caso contrario o de error, se queda como estaba definido. Si entramos

Ilustración 88. Vista Acciones del recurso coordinadas por Administrador

Identificador Coordenada	Latitud	Longitud	Identificador Zona de Caza
1	25.774	-80.19	1
2	18.466	-66.118	1
3	32.321	-64.757	1
4	39.395	-0.358	2
5	39.297	-0.3	2
6	39.329	-0.393	2
7	39.199	-0.771673	14
8	39.3117	-0.642584	14
9	39.1564	-0.634344	14

Fuente: Elaboración propia

Esta es la entidad o recurso sobre la que menos puede operar ya que solo puede ver todas las coordenadas de todas las actividades de caza creadas. Puede filtrar usando el buscador por la actividad que desee para ver las coordenadas de los vértices del polígono que define el área de esta. El motivo por el que no puede operar es porque las coordenadas son una consecuencia de los polígonos que definen las actividades, y por tanto, si ya se permite su modificación en las actividades no tiene sentido trabajar con coordenadas de manera aislada.

Finalmente, si pulsamos sobre “Informes” el administrador tiene la siguiente vista:

Ilustración 89. Vista Acciones del recurso Informes por Administrador

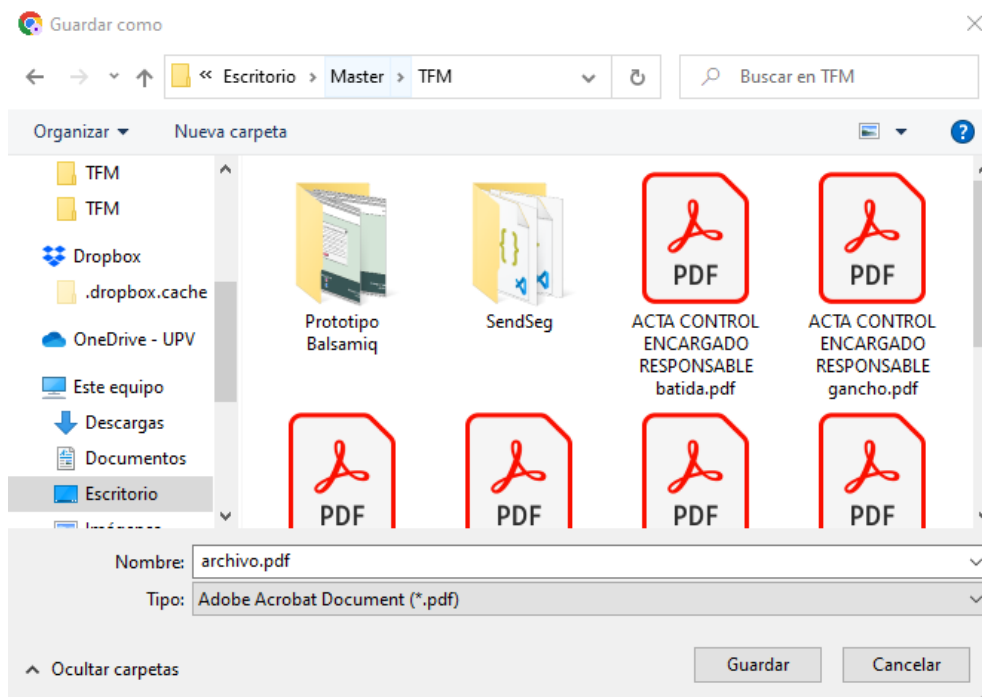
Identificador Informe	Identificador Actividad caza asociada	Acción
38	78	Descargar Borrar
Identificador Informe	Identificador Actividad caza asociada	Acción

Fuente: Elaboración propia

El administrador puede usar ese campo para filtrar el informe por la actividad a la que va asociada. Respecto a las acciones que puede hacer respecto a un informe solo son dos: descargar o eliminar un informe.

Si pulsa sobre “Descargar” se le muestra el panel típico de descarga donde decide el nombre que quiere ponerle, que por defecto se pone en “archivo.pdf”:

Ilustración 90. Descarga de Informe de actividad de caza



Fuente: Elaboración propia

La otra acción, “Eliminar”, hace que el informe adjunto desaparezca de modo en que la sociedad de cazadores que crea la actividad y subió un informe podrá sustituirlo por otro.

Con esta última acción queda mostrado todo lo que en la versión actual implementada de Sendseg cada uno de los tipos de usuario puede realizar, que se ajusta a los distintos casos de uso que se plantearon para desarrollar.

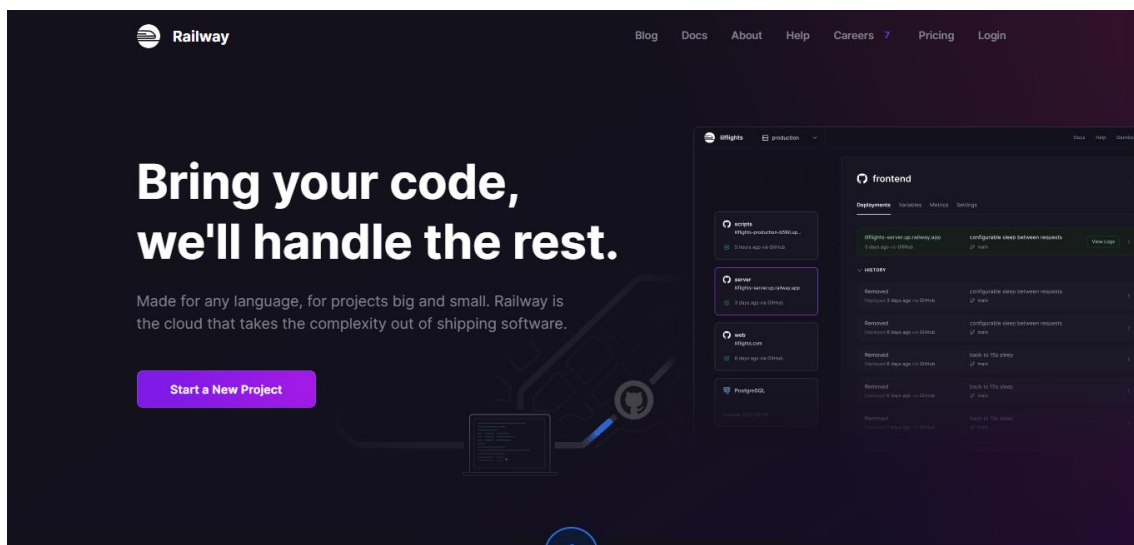
Anexo 5: Despliegue de Sendseg

En este anexo se va a explicar cómo y de qué manera, se realizó el proceso de despliegue Sendseg. Este paso es muy importante debido a que toda la fase de implementación se realizó de manera local, esto no permite evidentemente que otras personas como las sociedades de cazadores, senderistas, o el propio tutor del proyecto pudiesen usar la aplicación para probarla y dar *feedback*.

La idea inicial para realizar el despliegue era tener una máquina propia que hiciera de servidor donde alojar la aplicación, debido a las fechas en las que se desarrolló el proyecto esto no fue posible, y se buscaron otras opciones alternativas. A causa de que en un futuro se optará por la opción inicial, de manera provisional se decidió buscar distintos proveedores de servicios en la nube que tengan algún plan gratuito para poder alojar la aplicación y tener la base de datos necesaria.

Después de hacer una búsqueda y comparar opciones, se decidió usar Railway. La decisión de usar este proveedor frente a otros radica principalmente en las facilidades que proporciona para el despliegue así como la oferta de que cualquier usuario obtiene un crédito gratuito inicial de 5 dólares que pueden usarse para desplegar proyectos. Además, no es necesario introducir una tarjeta de crédito para acceder a ello.

Ilustración 91. Página de Railway

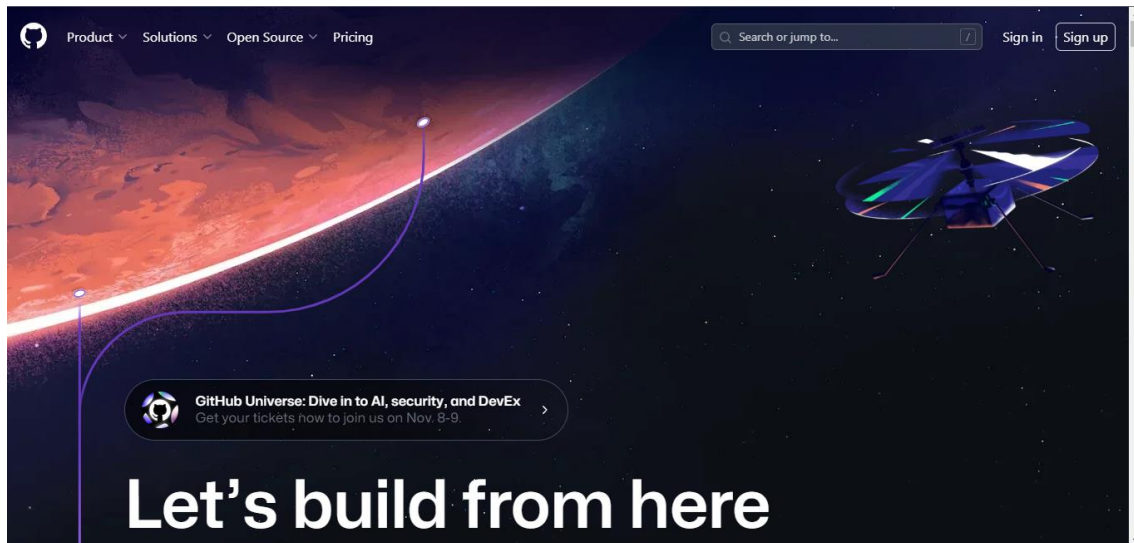


Fuente: Railway

Una vez seleccionado el proveedor, se comenzó con el proceso para hacer el despliegue de Sendseg, que fue el siguiente:

En primer lugar, se debe de crear una cuenta de Github en caso de no tenerla. En mi caso concreto, ya contaba con ella así que no fue necesario hacer este paso.

Ilustración 92. Página de Github



Fuente: Github

Una vez se tiene la cuenta en Github, se debe de crear un nuevo repositorio donde se subirá todo el código realizado. El formulario para crear el repositorio es el siguiente:

Ilustración 93. Creación nuevo repositorio Github

Fuente: Github

En este caso, el repositorio que creé es privado para que no pueda ser visible salvo por aquellas personas, o herramientas como Railway, a las que se le proporcione acceso posteriormente. Después de crear el repositorio este está vacío, para subir todos los archivos del proyecto desde Visual Studio, se introducen una serie de comandos en la consola.

El primer comando sirve para inicializar en el proyecto en Visual Studio Code Git:

Ilustración 94. Comando para inicializar Git en Visual Studio

```
git init
```

Fuente: Elaboración propia

Una vez inicializado se añade todos los archivos del proyecto para proceder a su subida al repositorio:

Ilustración 95. Comando para añadir archivos al siguiente commit

```
git add .
```

Fuente: Elaboración propia

Añadidos todos los archivos, se da nombre al commit que se va a hacer:

Ilustración 96. Comando para establecer el commit y su nombre

```
git commit -m "first commit"
```

Fuente: Elaboración propia

Después Github proporciona el siguiente comando que especifica la url que se corresponde con el repositorio creado y que es donde se subirá todo.

Ilustración 97. Comando para definir el origen y destino del commit

```
git remote add origin https://github.com/username/tfm-sendseg.git
```

Fuente: Elaboración propia

Una vez hecho todo lo anterior se hace la subida con el siguiente comando en la rama principal:

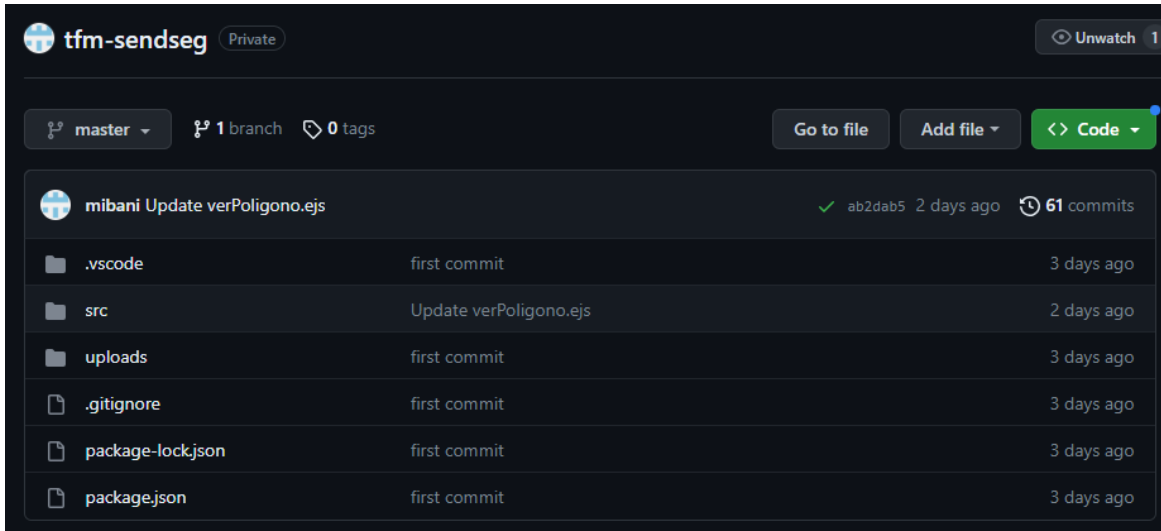
Ilustración 98. Comando para hacer push del commit

```
git push origin master
```

Fuente: Elaboración propia

De esta forma, si vamos a Github observaremos como aparecen los archivos subidos:

Ilustración 99. Repositorio después de commit



Fuente: Elaboración propia

En la captura anterior podemos observar cómo no se encuentra la carpeta node_modules que había en el proyecto en Visual Studio Code, esto se debe a que no es necesario para el despliegue y para ello se creó un archivo llamado “.gitignore” que permite establecer que archivos no se suben al repositorio.

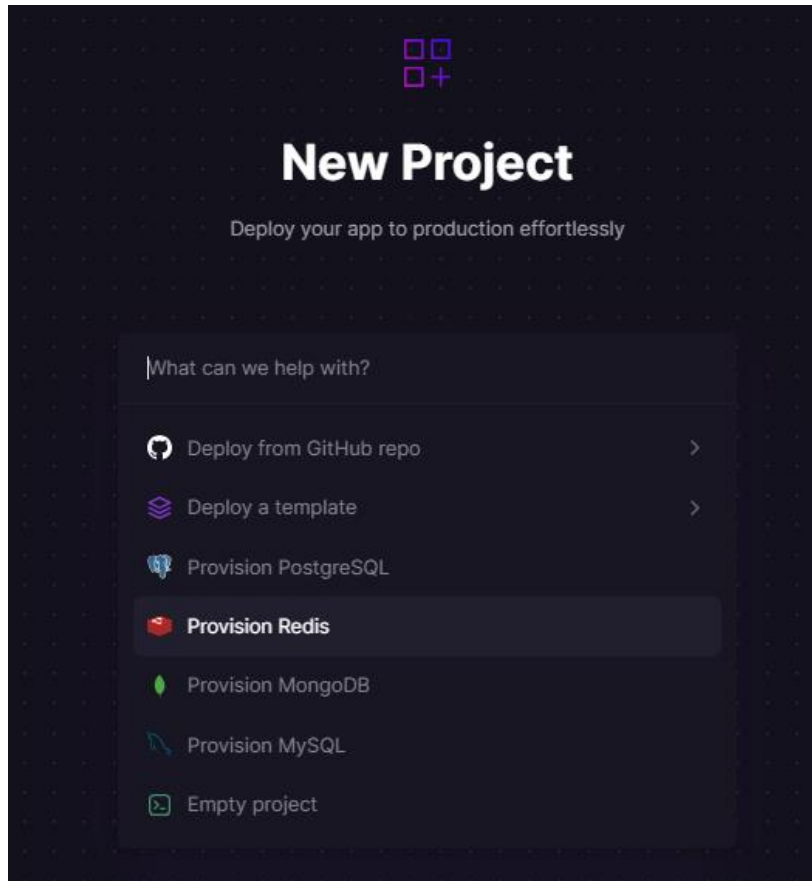
Ilustración 100. Archivo .gitignore

```
.gitignore
1 node_modules
```

Fuente: Elaboración propia

Una vez se tiene en el repositorio el proyecto subido, pasamos a Railway donde se crea una cuenta usando la creada en Github. Pulsamos en “New Project” y tenemos las siguientes opciones:

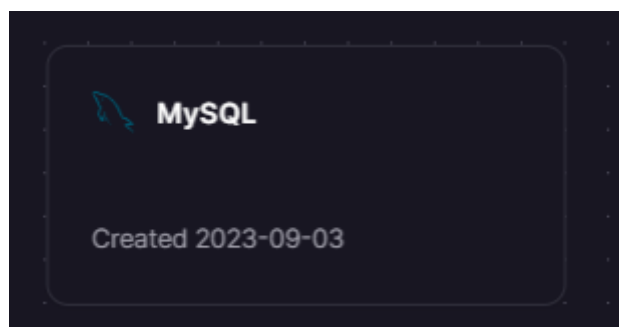
Ilustración 101. Railway opciones New project



Fuente: Railway

En primer lugar se hizo el despliegue de la base de datos, para ello pulsamos en “Provision MySQL” y tendremos la siguiente vista:

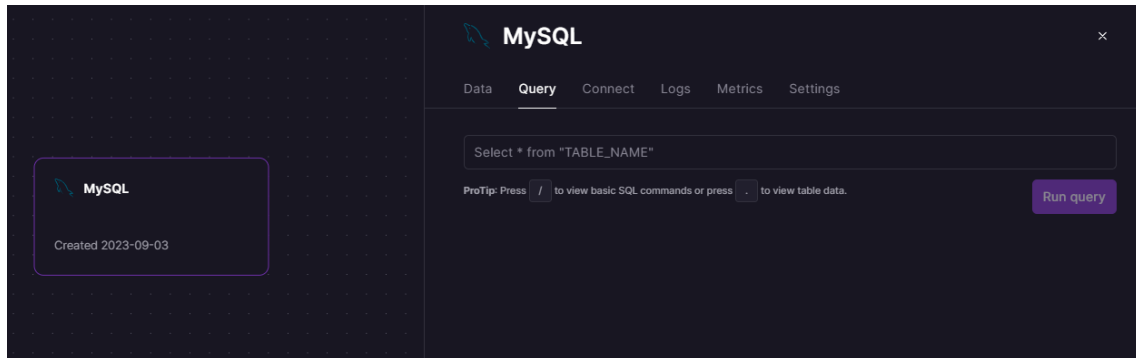
Ilustración 102. Provision MySQL



Fuente: Railway

Al pulsar sobre ella se muestra un menú y vamos a “Query”, donde se copia y pegan los scripts de MySQL que se obtuvieron para crear cada una de las tablas:

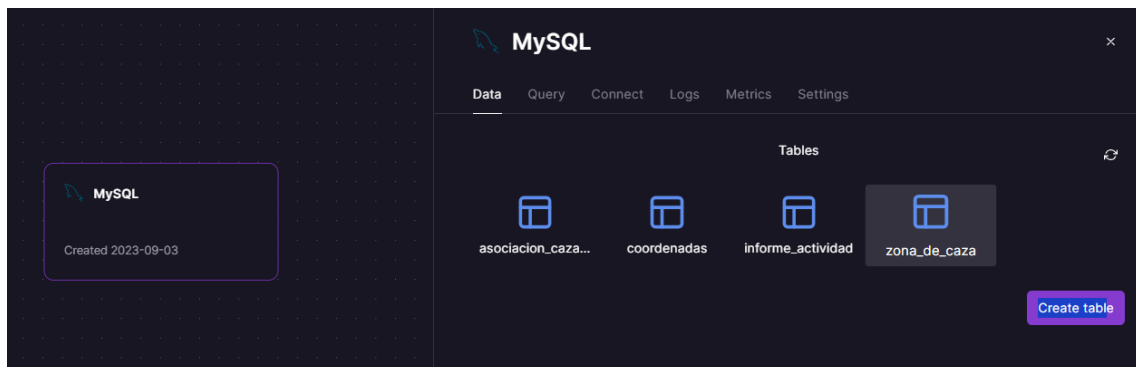
Ilustración 103. Railway MySQL query



Fuente: Railway

Una vez copiado y pegado el script de creación de cada una de las tablas en el apartado “Data” tenemos lo siguiente:

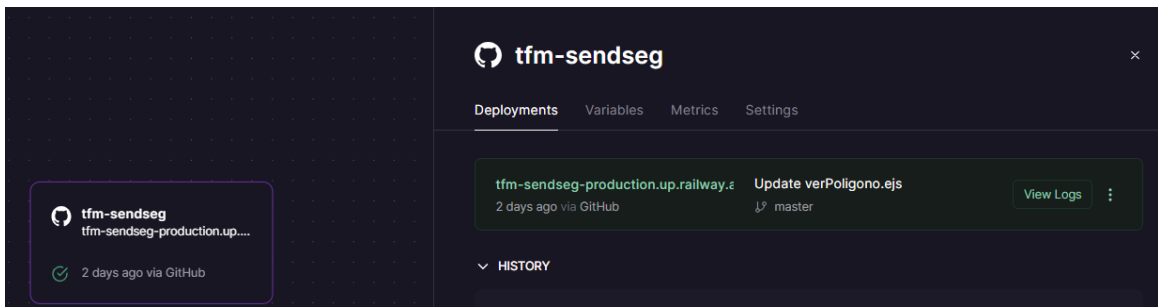
Ilustración 104. Tablas base de datos creadas en Railway



Fuente: Railway

Realizado este paso ya tenemos la base de datos relacional que teníamos implementada localmente desplegada. Ahora pasamos a desplegar la aplicación en sí, para ello volvemos a crear un nuevo proyecto pero esta vez pulsamos en “Deploy from Github repo”, donde seleccionamos el repositorio creado con todos los archivos del proyecto y tenemos la siguiente vista:

Ilustración 105. Despliegue aplicación a partir del repositorio en Railway

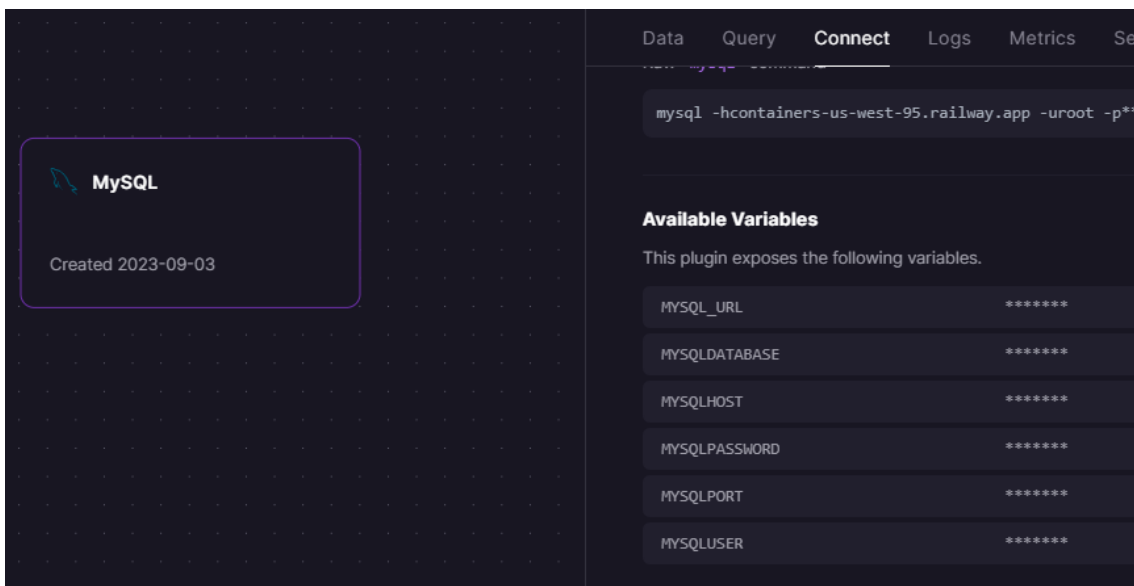


Fuente: Railway

Después de esto, pulsamos sobre “Settings” para, o bien hacer el despliegue en un dominio que hayamos adquirido previamente por nuestra cuenta, o bien en uno que nos genere Railway. En este caso, se usó la segunda opción ya que no se ha adquirido todavía ningún dominio para el despliegue de Sendseg.

Por último respecto a Railway, falta hacer la conexión entre la base de datos desplegada y la aplicación, para ello se usaron variables de entorno. Para ver los valores de estas variables se accede a la base de datos y en el apartado “Connect” se observan los valores:

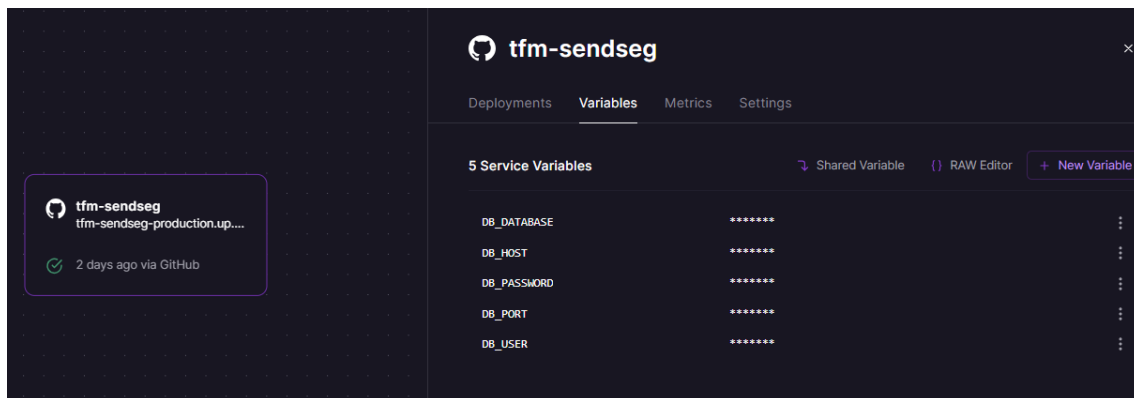
Ilustración 106. Variables entorno base de datos Railway



Fuente: Railway

Obtenidos estos valores, vamos a la aplicación y en el apartado de “Variables” introducimos todos los valores necesarios para establecer la conexión, quedando del siguiente modo y teniendo la conexión ya correctamente establecida:

Ilustración 107. Establecer variables en despliegue aplicación Railway



Fuente: Elaboración propia

Una vez alcanzado este punto, tenemos desplegados los dos elementos necesarios para el correcto funcionamiento de Sendseg, pese a esto, se tuvieron que hacer algunas modificaciones entre las que destacan:

- Links y redireccionamientos debieron ser cambiados de “localhost” a la url del dominio proporcionado por Railway en todo el proyecto.
- Una consulta concreta en base de datos que usaba el operador MAX() tuvo que ser modificada ya que daba problemas y era necesario hacer uso del GROUP BY para que funcionase correctamente.
- Modificar los valores de las variables para la conexión a la base de datos y el puerto donde se aloja la aplicación de valores locales a valores con variables de entorno.

Ilustración 108. Uso de variables de entorno

```
user: process.env.DB_USER || 'root',  
password: process.env.DB_PASSWORD || 'root',  
host: process.env.DB_HOST || 'localhost',  
port: process.env.DB_PORT || 3306,  
database: process.env.DB_DATABASE || 'sendseg',
```

Fuente: Elaboración propia

- En las credenciales de la API key creadas para el uso de los mapas de Google Maps en Sendseg, se tuvo que añadir en las “Restricciones de sitios web” la url proporcionada por Railway para permitir su uso en dicho dominio. En caso de no hacer esto los mapas no cargaban y por tanto la aplicación no podía ser usada.

Realizados estos cambios, se logró el objetivo de desplegar la aplicación, aunque es cierto que existen algunas limitaciones que no se han podido solventar al depender de un proveedor de este tipo y que en un futuro con la máquina propia se solventará de manera muy sencilla.