UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITÈCNICA SUPERIOR DE GANDIA

Grado en Tecnologías Interactivas

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

“Una herramienta para la visualización y el apoyo a las decisiones en la gestión de servicios de emergencia sanitarios”

TRABAJO FINAL DE GRADO

Autor/a:

Carlos Tortosa Micó

Tutor/a:

Víctor Sánchez Anguix

Juan Miguel Alberola Oltra

GANDIA, 2021

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Resumen

Los servicios de emergencias sanitarios se enfrentan a una gran variedad de desafíos y problemas relacionados con la gestión de sus limitados recursos. Uno de estos desafíos es decidir la ubicación de los vehículos de emergencias sanitarias como los soportes vitales básicos (SVB) y los SAMU.

La ubicación de estos vehículos debe ser capaz de dar servicio a la mayor parte de la población posible en un tiempo reducido, teniendo en cuenta tanto la demanda presente como demanda futura. La ubicación de estos vehículos determinará qué áreas quedan cubiertas, así como el tiempo de respuesta a las diferentes zonas de la Comunitat Valenciana.

En este proyecto se presenta una herramienta de visualización que permite a los gestores del servicio de emergencia visualizar el resultado de diferentes asignaciones de vehículos a ubicaciones, así como el sector de la población cubierta en rangos de tiempo variables. La herramienta permite tanto la visualización de ubicaciones existentes, como crear nuevas asignaciones dinámicamente.

La herramienta ha recibido retroalimentación de usuarios reales, incluyendo personal sanitario. El grado de satisfacción ha sido alto y ha servido para compilar una lista de mejoras que incluir en el proyecto más allá de su vida como Trabajo Final de Grado.

Palabras clave: visualización; sistemas de apoyo a la decisión; interfaces gráficas; ingeniería del software; experiencia de usuario

Abstract

Emergency health services face a variety of challenges and problems related to the management of their limited resources. One of these challenges is deciding the location of health emergency vehicles such as basic life support (SVB) and SAMUs.

The location of these vehicles must be able to serve as much of the population as possible in a short time, considering both present and future demand. The location of these vehicles will determine which areas are covered, as well as the response time to the different areas of the Valencian Community.

This project presents a visualization tool that allows emergency service managers to visualize the result of different assignments of vehicles to locations, as well as the sector of the population covered in variable time ranges. The tool allows both the visualization of existing locations and the dynamic creation of new assignments.

The tool has received feedback from real users, including healthcare personnel. The degree of satisfaction has been high and has served to compile a list of improvements to be included in the project beyond its life as a Final Degree Project.

Keywords: Visualization; decision supports systems; graphical interfaces; software engineering; user experience

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tabla de contenidos

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice de ilustraciones

Capítulo 1 – Introducción

1.1– Introducción

La crisis del COVID-19 ha remarcado la importancia de una gestión correcta y optimizada de los recursos sanitarios de los que disponen las administraciones. Puesto que el tiempo de reacción de un vehículo sanitario y el tiempo de trayecto hasta el paciente resulta crítico para sus posibilidades de recuperación, y en algunos casos de supervivencia, es necesario estudiar con cautela la localización de estos vehículos.

Para este propósito, es deseable maximizar el número de habitantes que un vehículo es capaz de cubrir en una ventana de tiempo reducida, así como distribuir los servicios acorde al número de habitantes presentes en una zona.

Se han realizado diversos estudios y trabajos, como el que encontramos en [1], con el objetivo de encontrar soluciones optimizadas con las que mejorar la atención que los servicios sanitarios pueden prestar a los ciudadanos.

Sin embargo, estas soluciones requieren por norma general una enorme cantidad de trabajo manual, que puede no resultar tan válido en el futuro si con el tiempo cambian ciertas condiciones (número de vehículos de los que disponen las administraciones, su horario de disponibilidad, localización de bases, etcétera)

Actualmente se encuentra en desarrollo iReves (innovación en Reubicación de Vehículos de Emergencias Sanitarias), un ambicioso proyecto de herramienta inteligente cuyo objetivo es asistir en el proceso de gestión de las emergencias sanitarias.

El proyecto está compuesto por dos componentes principales:

- Reves Data: Componente de recogida de datos obtenidos del mundo real y su procesado mediante técnicas estadísticas y Big Data en información de utilidad.
- Reves Mind: Componente de generación de decisiones basadas en los modelos proporcionados por Reves Data. Se busca que Reves Mind asista de manera reactiva y/o proactiva en la toma de decisiones mediante las sugerencias que puede proporcionar al usuario.

El proyecto iReves, como se ha mencionado, se encuentra en desarrollo y actualmente se están buscando medios para su financiación. Se encuentra en progreso la presentación del proyecto a la convocatoria de ayudas que proporciona la Agencia Valenciana de Innovación[1].

En este documento se describirá el proceso de concepción e implementación del componente visual de Reves Mind.

[1] https://innoavi.es/es/

1.2 – Objetivos

El objetivo que se persigue con este trabajo es el de desarrollar una herramienta ejecutable de ordenador que contenga un mapa interactivo, en el cual puedan representarse una serie de entidades sanitarias (bases y vehículos).

La herramienta debe permitir definir un tiempo en minutos para el cálculo de isócronas y su representación, realizar una estimación de la población que habita en la región de la isócrona y representar la intersección entre isócronas (también con posibilidad de estimar los habitantes en esa región).

Para poder alcanzar este objetivo, comenzaremos con una fase de investigación, con la cual documentarnos sobre el ecosistema de conceptos teóricos referentes a las emergencias sanitarias y las herramientas de apoyo a la decisión.

Se realizará una labor de familiarización con otras aplicaciones ya disponibles orientadas al ámbito del análisis urbano para conocer métodos de interacción y servicios que nos puedan resultar de utilidad.

Se describirá el problema a solventar, extrayendo las necesidades a cubrir. Seguidamente, decidiremos sobre el formato de la aplicación en base a estas necesidades. Analizaremos y seleccionaremos las herramientas que mejor nos permitan realizar la implementación.

Simultáneamente al desarrollo en el sentido técnico, trataremos de alcanzar los siguientes subobjetivos:

- Llevar a cabo el desarrollo bajo una metodología de trabajo ágil, respetando los tiempos de entrega establecidos y desarrollando el producto de una manera incremental en lateral (dentro de lo posible).
- Recoger las impresiones y críticas que usuarios reales tengan sobre nuestra aplicación tras su final como TFG, para, de esa manera, mejorar la aplicación en el proceso de desarrollo futuro.

1.3 – Motivaciones

Se pretende proporcionar una herramienta que pueda potencialmente ser usada en un entorno real de apoyo a la decisión en un contexto de gestión de vehículos sanitarios.

En [1] se propone una propia solución para la localización de estos vehículos mediante la optimización de las áreas de población cubiertas por los servicios sanitarios.

Con este proyecto se busca apelar al mismo problema, si bien abarcándolo desde un ángulo distinto. Se propone y describe en este documento la creación de una aplicación dinámica que itere sobre el trabajo previamente realizado..

Puesto que el objeto de este trabajo será una aplicación empleada por usuarios, es la oportunidad perfecta para poner en práctica los conocimientos impartidos en el Grado en Tecnologías Interactivas.

Las asignaturas del grado cuyo contenido ha resultado de mayor utilidad en el desarrollo de este proyecto han sido las siguientes:

- Desarrollo de un proyecto electrónico utilizando metodología CDIO: Aunque todas las asignaturas de proyecto semestrales del grado han servido para cementar la práctica de la metodología ágil Scrum, destacamos esta en concreto por resultar la primera toma de contacto en la metodología de creación de un producto.
- Programación 1 y 2: Por proporcionar los conocimientos relacionados con el diseño, documentación y testeo de código, adicionalmente a su implementación en lenguaje específico.
- Tecnologías de la información geográfica: Por establecer las bases necesarias para el desarrollo de una aplicación cuyo objetivo es la representación de elementos geolocalizados en un mapa interactivo.
- Proyecto diseño y programación web: Por la instrucción de conocimientos de programación e implementación de aplicaciones web, tal y como es el caso de este proyecto

1.4 – Estructura del documento

Este documento va a estar dividido en las siguientes secciones:

- Capítulo 2. Marco teórico: Se establecerán los conceptos teóricos relativos a las emergencias sanitarias (tipos de emergencias, tipos de vehículos sanitarios que las atienden, etcétera) y las herramientas de apoyo a la decisión (concepto y tipos según varios criterios de clasificación). También se realizará una breve descripción de diversas aplicaciones de visualización de información urbana disponibles públicamente y que han inspirado esta herramienta.

- Capítulo 3. Análisis y diseño: Se realizará un análisis del problema que nuestra aplicación trata de solventar, extrayendo las necesidades a cubrir y las funcionalidades que se han de implementar. Comentaremos además diversos aspectos relacionados con el proceso de selección de tecnologías, diseño del software y metodología de trabajo practicada para el desarrollo.

- Capítulo 4. Implementación: Se comentarán diversos aspectos relacionados con los métodos y herramientas utilizados para la implementación de la aplicación en formato de programa de escritorio.

- Capítulo 5. Conclusiones: Terminaremos este trabajo realizando una reflexión sobre el proceso de desarrollo del proyecto, buenas prácticas realizadas, competencias transversales adquiridas y propuestas de mejora que se podrían seguir en un futuro.

Capítulo 2 - Marco teórico

En este capítulo trataremos de establecer el ecosistema de definiciones y conceptos teóricos que son pertinentes al ámbito de los sistemas de apoyo a la decisión, orientados a los servicios de emergencias sanitarias.

2.1 - Emergencias

El Sistema de Emergencias Médicas (SEM) de la provincia de Valencia realiza una distinción entre 3 posibles tipos de emergencias, basándose en el nivel de riesgo o peligro que amenaza la salud de un paciente [1]:

- El tipo 1 comprende aquellas emergencias donde la vida del accidentado corre peligro mortal en un periodo corto de tiempo, por lo que se le debe atender de manera inmediata.
- El tipo 2 acoge aquellas emergencias donde no se corre peligro mortal, pero se requiere de una atención rápida para evitar el agravamiento de la situación del accidentado.
- El tipo 3 acoge las emergencias que no son clasificadas en los otros dos tipos.

En función de cómo se clasifique la situación de emergencia, el Centro de Información y Coordinación de Urgencias (CICU) puede decidir qué tipo de vehículo enviar para atender y gestionar la situación.

Los vehículos que consideramos de interés y con los que contaremos para los propósitos de este proyecto son los SVA y los SVB, puesto que son generalmente los que son empleados para atender las emergencias.

Se definen ambos tipos de vehículos [2]:

- SVA/SAMU: Unidades de Soporte Vital Avanzado. Son vehículos de vigilancia intensiva, destinados al transporte de pacientes críticos que requieren necesariamente de vigilancia y asistencia constante en ruta. Atienden emergencias de tipo 1.

- SVB: Unidades de Soporte Vital Básico. Son responsables de trasladar a enfermos estables que no presentan riesgo vital durante el transporte, pero que pueden necesitar atención en ruta para evitar el agravamiento de la lesión o la aparición de secuelas. Atienden emergencias de tipo 2 y algunas instancias de tipo 3.

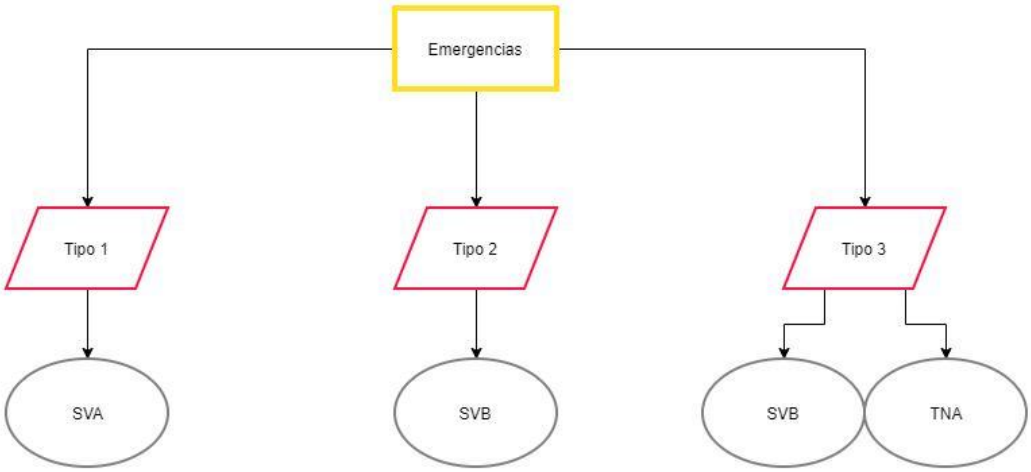


Ilustración 1 - Ambulancias responsables según el tipo de emergencia
Fuente: Elaboración propia, basado en [1]

Los TNA (Transporte no asistido) realizan labores de transporte de pacientes en litera.

2.2 - Sistemas de apoyo a la decisión

Según se explica en [3], ya en la década de los años 60 los investigadores estudiando el uso de modelos computados para asistir de manera sistemática en la toma de decisiones. El primer uso del término “Decision Support System” (DSS) se registra en [3], donde Gorry y Scott sugieren su uso para describir los sistemas de información diseñados para el soporte de decisiones semiestructuradas[2].

El término esencialmente describe aquellos sistemas informáticos interactivos que proporcionan soporte a aquellas personas en la responsabilidad de toma de decisiones, sin llegar a sustituirlos. Estos sistemas emplean los datos de los que disponen para generar un modelo para ayudar en la tarea de toma de decisiones.

En fecha de publicación de [5], alrededor de un 73% de las 210 aplicaciones DSS que contempla el artículo se enmarcaban en el ámbito de la gestión corporativa. Dentro de este ámbito, se puede realizar una clasificación más específica según el área de aplicación:

[2] Aquella decisión donde no todos los factores están predeterminados

- POM (*Production and operation management*, 44.16%)
- Marketing, transporte y logística (17.53%)
- MIS (13.64%)
- Sistemas multifuncionales (8.44%)
- Finanzas (6.49%)
- Estrategias de gestión (3.9%)
- Recursos humanos (3.9%)

En [4] se propone un método de clasificación basado en el grado en el cual el DSS influye en la decisión final.

Mediante este criterio de clasificación, es posible agrupar los DSS en los siguientes grupos:

- **Sistemas de análisis de la información** (4%), los cuales permiten, mediante el tratamiento de información interna complementada con información externa y el uso de modelos, generar información de utilidad para la toma de la decisión.
- **Modelos de contabilidad** (1%). Permiten calcular las consecuencias de una acción hipotética, generalmente en el ámbito de las finanzas, las cuales pueden cuantificarse en un documento de contabilidad, balances de situación u otros.
- **Modelos de representación** (21%). Buscan calcular las consecuencias de una acción en base a algún tipo de modelo, incluyendo modelos de simulación.
- **Modelos de optimización** (51%). Pretenden generar soluciones óptimas, dentro de las restricciones definidas por el usuario.

Modelos de sugestión (23%). Generan una sugerencia de solución específica, calculada de manera sistemática y con poco margen de interpretación por parte del usuario que emplea el sistema. Acorde a esta definición y adicionalmente la que ofrece [2], consideramos que nuestra herramienta se situaría mayormente en el ámbito de los sistemas de análisis de la información, pues hace hincapié en la manipulación de datos mediante herramientas digitales operadas por el usuario y con el objetivo de generar información de utilidad (como en nuestro caso, el número de habitantes de una región).

Podríamos argumentar que la aplicación se encontraría a mitad de camino entre un sistema de análisis de la información y un modelo de representación, ya que el cálculo de las isócronas representa la consecuencia de un caso hipotético simulado.

El objetivo más importante por el cual se realizó este proyecto fue el de proporcionar una herramienta de apoyo a la decisión a aquellos organismos que deben coordinar y gestionar los recursos sanitarios.

Se han realizado estudios en la Comunidad Valenciana para tratar de analizar su situación particular, así como diseñar soluciones que permitan optimizar el uso de los recursos sanitarios en nuestro territorio (como es el caso de [1])

Sin embargo, estos estudios tienen la desventaja de ser una instantánea del momento en el que se realizan. Es deseable un sistema que permita explorar diferentes opciones y que nos permita afrontar situaciones futuras.

Consideramos necesaria una herramienta que pudiera cargar datos de manera dinámica y proporcionar un soporte para la visualización de los vehículos.

Debido a la importancia de que estos vehículos puedan atender situaciones de emergencia en ventanas de tiempo reducidas, necesitamos que el sistema brinde herramientas para calcular y representar gráficamente aquellas áreas que los vehículos pueden abarcar en un tiempo estipulado (isócronas). Estas isócronas deben poderse calcular de manera dinámica, para que el usuario pueda manipular los vehículos sobre el mapa y realizar juicios según los resultados.

2.3 - Visualización en aplicaciones urbanas

La democratización del conocimiento, el desarrollo de los medios de comunicación y el aumento exponencial de nuestros bancos de datos en volumen y complejidad hacen imperiosa la necesidad de modernizar nuestras herramientas y apostar por los proyectos que implementan algún tipo de interfaz visual como principal medio de interacción.

El artículo [5] concluye realizando gran énfasis en la importancia de desarrollar las tecnologías digitales interactivas como motor del progreso en la gestión y planificación de la sociedad del futuro.

Habiendo establecido la importancia de este tipo de herramientas, en esta sección se describirán diversas plataformas de visualización y tratamiento de información urbana para tener en cuenta para el desarrollo de este proyecto.

Consideramos importante observar aplicaciones del mundo real para ver que tecnologías se emplean y que vías de interacción se suelen ofrecer al usuario en este tipo de aplicaciones. A continuación, se describen algunas de las más relevantes.

NASA SEDAC Population Estimation Service[4]

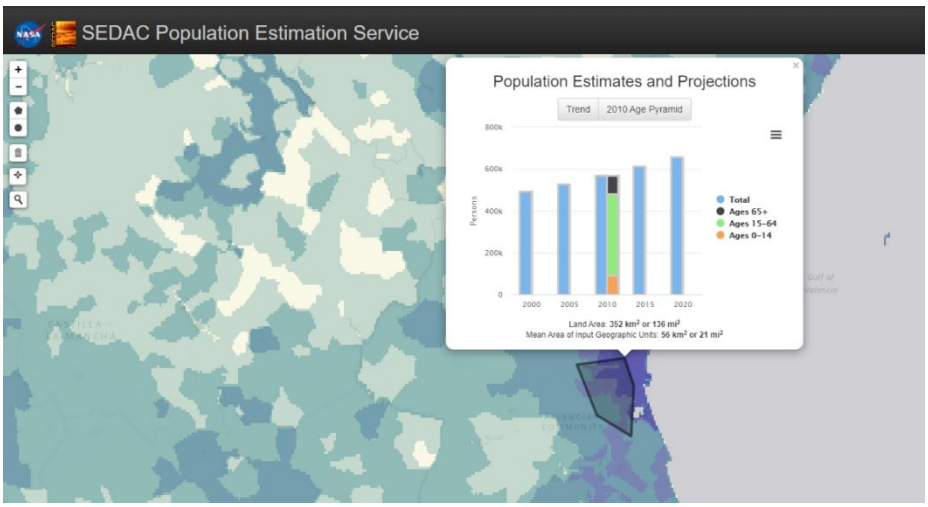


Ilustración 2 - Interfaz del SEDAC Population Estimation Service

Se trata de una herramienta web del SEDAC (*Socioeconomic Data and Applications Center*) para la estimación de población dentro de áreas definidas por el usuario sobre un mapa interactivo (Ilustración 2).

El SEDAC es un centro de datos de la NASA que pone a disposición del público todo tipo de recursos, artículos y bases de datos relacionados con datos socioeconómicos del mundo.

La herramienta es gratuita y sencilla de utilizar. Pero, más importante, dispone de una API[5] para emplearla de manera programática mediante peticiones por Internet.

Dicha API fue incorporada en un estado más temprano de este proyecto para la estimación de población, gracias a su facilidad de uso, conveniencia y gratuidad indiferente del número de peticiones realizadas.

Los datos que podemos obtener con esta API concreta incluyen:

- El número de habitantes total presente en una región
- El número de habitantes por segmento de edad
- Área total de la región en kilómetros cuadrados
- Área de la región que es tierra sólida (por descarte, se puede calcular el área cubierta por agua)

Podemos consultar estos datos acordes al año que deseemos, siendo el más temprano el año 2000 y 2020 el más actual, pudiendo consultar en intervalos de 5 años (2005, 2010 y 2015).

Ventajas:

[4] *https://sedac.ciesin.columbia.edu/mapping/popest/pes-v3/)*
[5] *Application Programming Interface,* en este contexto, servicio en línea que puede invocarse desde el código

- La API es muy sencilla de utilizar, con ejemplos completos disponibles en la propia página.
- Totalmente gratuita y sin límites de peticiones diarias o mensuales

Desventajas:

- Respuestas del servidor con gran cantidad de metainformación que no es estrictamente necesaria. Si bien esto no es un problema grave para nuestra aplicación, podría dificultar la conversión de nuestro programa a otra plataforma o entorno donde el ancho de banda de Internet sea limitado, o donde el dispositivo donde corre el programa sea de prestaciones ajustadas.
- Gran imprecisión en regiones de menor tamaño. Si bien las estimaciones de población para regiones del tamaño de comunidades autónomas eran aceptables, a nivel de municipio o ciudad dejaba mucho que desear.

WoprVision (beta)[6]

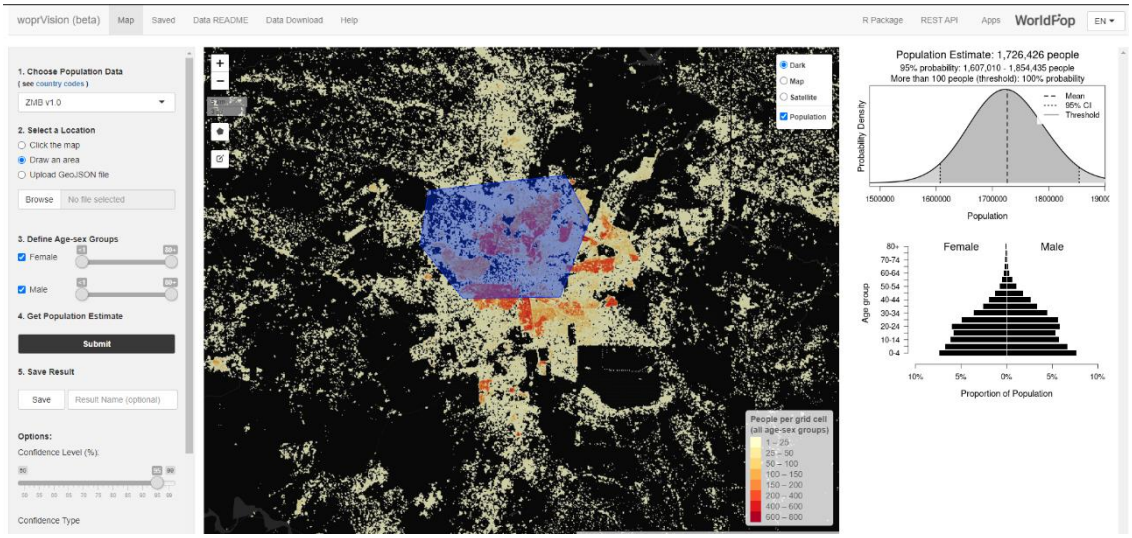


Ilustración 3 - Interfaz de la herramienta WoprVision

Herramienta similar a la proporcionada por el SEDAC, en este caso de la mano de WorldPop.

Permite visualizar la distribución poblacional por sexo y edades de la región que dibujemos o indiquemos sobre el mapa. El panel de control de la izquierda (Ilustración 3) nos da algunas opciones adicionales para personalizar la búsqueda. Tiene el inconveniente de que actualmente solo proporciona datos referentes a algunos países de África.

[6] https://apps.worldpop.org/woprVision/

OpenRoute[7]

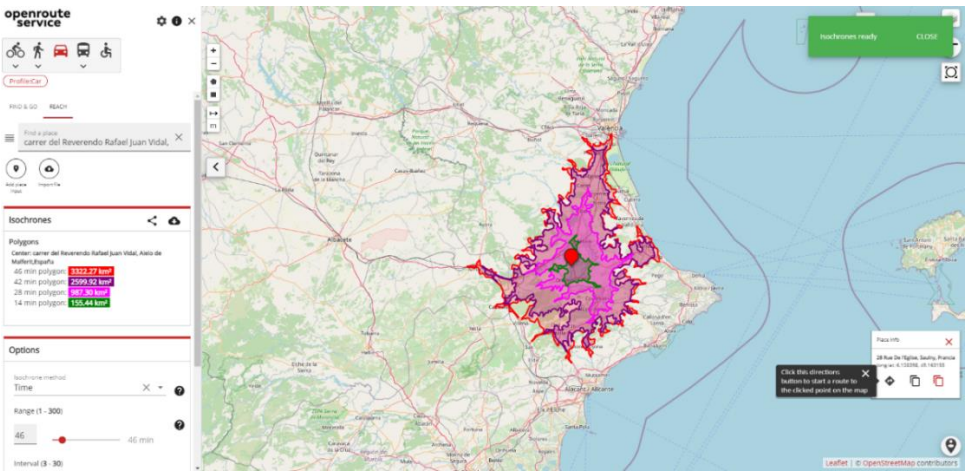


Ilustración 4 - Interfaz de la herramienta de OpenRoute

OpenRoute es un servicio gratuito que ofrece diversas herramientas para el cálculo de distancias entre puntos, cálculo de rutas para vehículos, cálculo de isócronas según parámetros, etcétera.

Podemos personalizar cualquier parámetro de nuestras consultas mediante las herramientas del panel de control izquierdo (Ilustración 4)

Entre otras, las funcionalidades que ofrece son:

- Cálculo y representación de isócronas según los parámetros definidos (tipo de vehículo, distancia, tiempo de recorrido...)
- Cálculo y representación de la vía óptima entre punto A y B, con opción para evitar puntos indeseados (peajes, puntos específicos y otras restricciones).
- Cálculo de distancia entre varios puntos indicados, así como el tiempo de recorrido necesario para un vehículo hipotético.

[7] *https://maps.openrouteservice.org/#/*

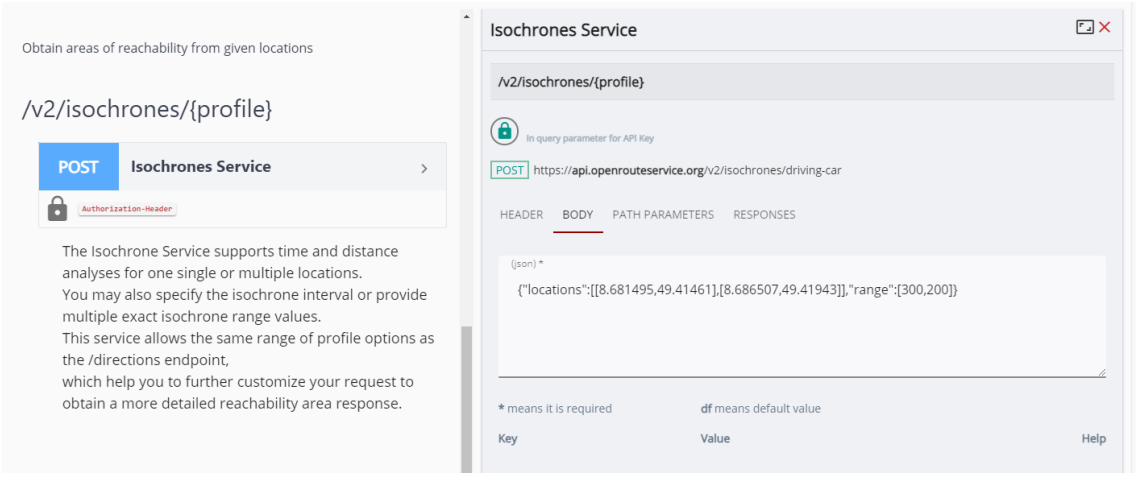


Ilustración 5 - API REST de OpenRoute para el cálculo de isócronas
Fuente: Página oficial de OpenRoute, subsección de API

OpenRoute pone a disposición de manera gratuita consultas REST para el cálculo y procesado de rutas e isócronas definidas según los parámetros del usuario. Si bien su uso es gratuito, el número de consultas diarias se encuentra limitado (aunque consideramos que es suficiente).

Puede emplearse usando la herramienta que pone a disposición en su web (Ilustración 5)

La funcionalidad de representación de isócronas de este proyecto se basa en esta herramienta.

Capítulo 3 – Análisis y diseño

En este capítulo se realizará una descripción del proceso de concepción de la aplicación, la metodología de trabajo empleada y proceso de selección de software de apoyo.

3.1 - Características del problema

En [1] se realiza el estudio y cálculo de una solución optimizada para la relocalización de los vehículos sanitarios de los que dispone el territorio valenciano.

El objetivo principal es el de recolocar dichos vehículos de manera acorde a la distribución de la población y las distancias que estos vehículos pueden cubrir en un tiempo de actuación crítica. De esta manera, los servicios de emergencia pueden cubrir al mayor número de habitantes en el menos tiempo posible.

Como es evidente, dicho trabajo final de grado es un documento estático. Por la naturaleza de este, al igual que otros artículos y estudios, se trata de una instantánea del tiempo en el que se realiza.

Otro problema relevante es la cantidad de trabajo manual que se debe realizar para diseñar una solución optimizada. Este proceso involucra una gran inversión de tiempo y esfuerzo en recolección de datos de cada base y vehículos,

La propuesta con la que justificar y fundamentar la realización de este proyecto fue el de desarrollar una herramienta que pudiese cargar datos de un fichero y/o mediante una petición HTTP con los datos (incluyendo la localización) de los vehículos de los que nuestro cuerpo sanitario dispone en el territorio.

Buscamos proporcionar una herramienta digital de fácil uso con los que explorar escenarios alternativos y permitir al usuario tomar decisiones informadas.

Se decidió confeccionar una lista de funcionalidades que deben encontrarse en el programa:

- Se debe poder ver en un mapa una representación de los vehículos y bases, extraída su información de un fichero y/o base de datos.
- Se debe poder interactuar con los vehículos, preferiblemente de una manera sencilla (como pueda ser un simple *click* del ratón o un *tap* en una pantalla táctil).
- El programa debe ser capaz de plasmar en el mapa interactivo las isócronas de cada vehículo.
- El programa debe ser capaz de estimar la población contenida en cada región de isócronas
- De ser posible, el programa ha de ser compatible con la ejecución de instrucciones Python. El motivo principal de este requisito es el de poder incorporar esta aplicación al resto de la solución completa que se encuentra en desarrollo en colaboración con los tutores de este trabajo y otros alumnos, los cuales trabajan principalmente en Python.

En una sección posterior se realiza un estudio sobre el software a emplear para el desarrollo del proyecto en función de los criterios establecidos.

3.2 - Diseño de la solución

Puesto que la solución que proponemos es una aplicación software, el proyecto atravesó las siguientes fases:

1. Definición y caracterización de la necesidad o problema a solventar
2. Compilación de las funcionalidades para cubrir dicha necesidad
3. Elección de las tecnologías y entornos de trabajo que ofrecen mejor soporte y herramientas para implementar una solución
4. Diseño de los componentes del programa

5. Inicio de scrum (Realizado de manera cíclica por cada iteración):
    a. Selección de funcionalidades a implementar en el sprint
    b. Diseño de la funcionalidad
    c. Implementación
    d. Testeo
6. Empaquetamiento del software y distribución

Una vez reunidos los requisitos deseados para el producto, se procedió a formalizar el método de trabajo y monitorización del progreso. Priorizamos el empleo de métodos y herramientas que fuesen afines a la metodología impartida en el grado en tecnologías interactivas.

Se empleó metodología ágil Scrum con este propósito, el cual se desarrolla en más detalle en el apartado siguiente.

Como todo producto de software, la parte programática está formalizada y documentada mediante esquemas UML realizados previamente y actualizados conforme se detectaban necesidades adicionales desde el punto de vista del programa.

En cuanto a la elección de tecnologías para el desarrollo del proyecto, se realiza un estudio en la sección 3.4, el cual se encuentra más adelante.

Aunque se decidió desarrollar el programa como un programa de escritorio, también está diseñado para poder portarse al entorno web sin demasiados cambios.

3.3 - Metodología de trabajo

La realización de este proyecto se ha llevado a cabo mediante Scrum.

"*Scrum es un framework iterativo e incremental para proyectos o desarrollo de aplicaciones. Estructura el desarrollo en ciclos de trabajo llamados sprints*". (Sutherland, 2012, página 14)

Esencialmente, el desarrollo del producto se divide en una sucesión de incrementos de regular duración (típicamente, entre dos semanas y un mes). A cada incremento lo denominamos sprint.

Previo al inicio del desarrollo, se compilan en una lista los requisitos, las funcionalidades y otros aspectos del producto. Esta lista se denomina el *product backlog.*

Al inicio de cada sprint, se escogen las características sobre las que se trabajarán durante la duración del dicho, lo que conforma el *sprint backlog*.

Acto seguido, se realiza la asignación de las tareas a los miembros del Scrum (el equipo de desarrollo). Este proceso se suele realizar en una reunión de equipo denominada *sprint planning*.

Al final de un sprint, se realiza un sprint review, una reunión done se valoran los resultados alcanzados y se realiza una reflexión sobre aspectos a mejorar.

Es habitual la celebración diaria de reuniones llamadas *daily Scrum*, donde cada miembro del equipo informa sobre las tareas realizadas hasta el momento, las que está realizando, las que planea realizar a continuación, así como los problemas que haya tenido o esté teniendo en ese momento.

Como es evidente, este trabajo se ha realizado en solitario, por lo que el proceso de Scrum no se ha realizado de manera exacta a como se suele realizar dentro de un equipo.

En este caso, al inicio de cada sprint los tutores y el alumno han pactado las funcionalidades a implementar, así como su orden de prioridad. El alumno ha realizado las labores de formalización de las tareas en el tablero Kanban (descrito en mejor detalle más adelante), desarrollo de las tareas y presentación de los avances en los *sprint reviews*.

El alumno expone entonces las funcionalidades que se han conseguido implementar satisfactoriamente, así como aquellas que se encuentran a medio desarrollo y que requieren mayor atención o tutoría.

Tras la retroalimentación recibida de los tutores respecto al trabajo realizado, se procede a planificar el nuevo sprint y la nueva fecha de reunión.

La ilustración 6 muestra un esquemático resumido sobre la estructura de un *sprint.*

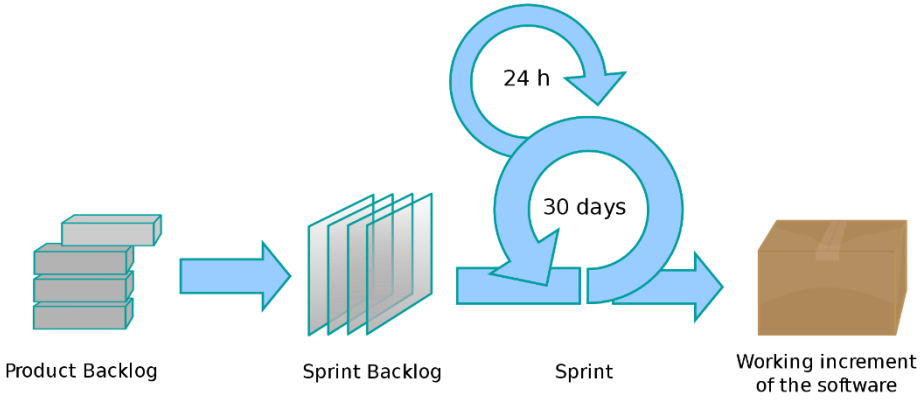


Ilustración 6 - Iteración de un sprint en el modelo Scrum
Fuente: Wikipedia (Creative Commons)

Generalmente, Scrum es se realiza complementándolo con herramientas digitales, siendo los tableros Kanban los más usuales.

Estos permiten plasmar las tareas a realizar en un formato que permite clasificarlas de manera sencilla según estén pendientes, en proceso o completadas, así como información relevante a estas, como la persona que las realiza, requisitos, duración estimada, etcétera.

Una herramienta Kanban muy popular, así como la que se ha usado para el desarrollo de este proyecto ha sido Trello.

Trello[8] es una herramienta ampliamente conocida y de uso gratuito, si bien se puede obtener acceso a algunas características y/o mejoras convirtiéndonos en usuarios de pago.

Permite la creación y manejo intuitivo de tarjetas virtuales que representan nuestras tareas.

[8] https://trello.com

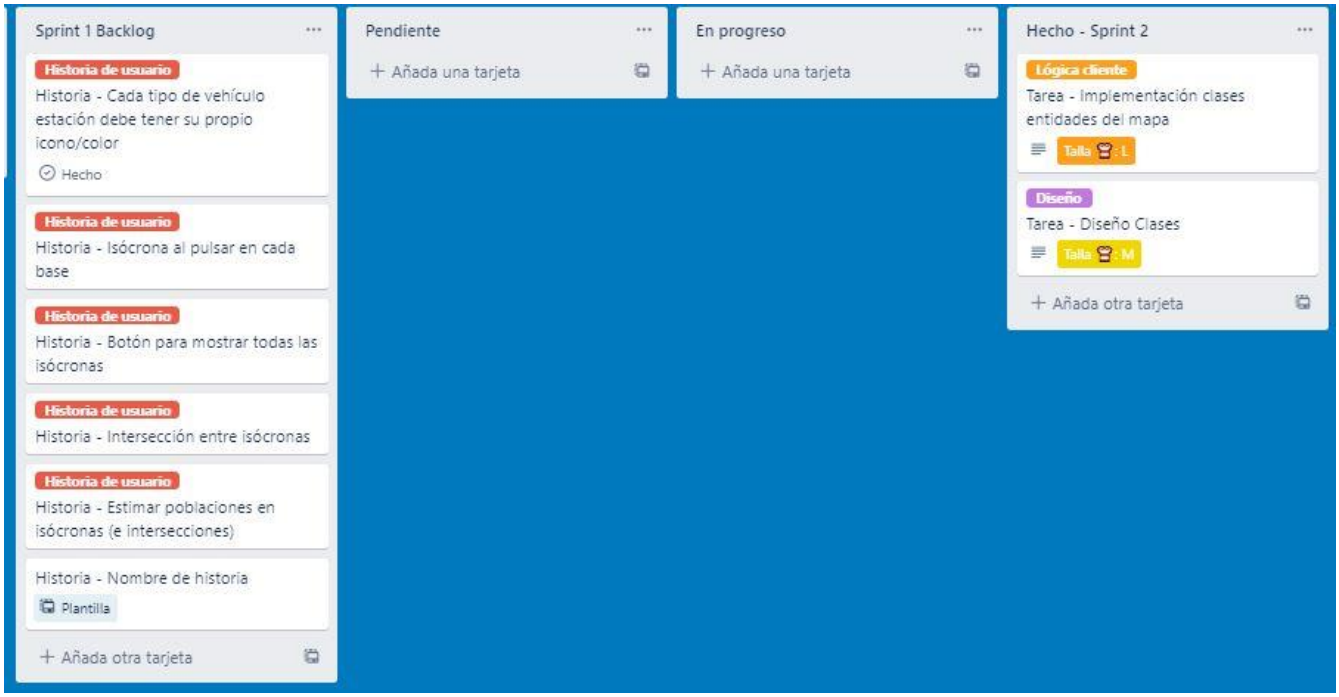


Ilustración 7 - Captura del tablero del proyecto durante el segundo sprint

En la ilustración 7 observamos el tablero del proyecto al final del segundo sprint. Cada tarjeta muestra el nombre de la tarea a realizar, una etiqueta describiendo la sección del programa que modifica (estilo de la interfaz, lógica del cliente, diseño, lógica del *backend*[9], …) y un tamaño de tarea, con la que se pretende estimar la complejidad de implementarla (desde una talla XS a una XL).

El desarrollo de la aplicación ha tomado un total de 10 *sprints*, de los cuales los 7 primeros han durado 2 semanas y los 3 últimos una semana.

3.4 - Criterios y proceso de elección de tecnologías.

De manera previa al inicio del desarrollo, se especificaron una serie de requisitos y de características que preferiblemente debían estar disponibles en el programa.

- El usuario debe poder visualizar en un mapa los elementos de la simulación (situación de los vehículos y bases, tipos de vehículos, isócronas, …)
- Debe poder manipularlos directamente en la propia interfaz con sencillez y de manera directa
- Los contenidos y datos representados en el mapa deben actualizarse de manera automática y dinámicamente, ya sea accediendo a los contenidos de un fichero o mediante peticiones a un servidor.
- De manera preferible, la tecnología empleada debe ser compatible con el ecosistema de Python, ya que es en este lenguaje donde se encuentran disponibles las librerías de procesado de datos más potentes y extendidas.

[9] *Backend* se refiere a la lógica del programa que transcurre de manera oculta a la vista

Tras la recopilación de estas recomendaciones y requisitos, se realizó la evaluación de diversos entornos de trabajo y tecnologías, muchas de ellas de libre uso y disponibles en internet.

Se evaluaron bajo una serie de parámetros:

- Gratuidad de uso/Código libre
- Potencial para crear interfaces de usuario adecuadas para las especificaciones detalladas anteriormente, así como la facilidad para alcanzar este objetivo.
- La herramienta cuenta con librerías con las que acelerar el proceso de desarrollo, así como documentación actualizada y complementada por aportaciones de usuarios de esta.

A continuación, se detalla el análisis realizado sobre las tecnologías candidatas:

Tabla 1 - Comparación entre tecnologías candidatas para la implementación de la aplicación

Nombre	**Descripción**	**Gratuidad**	**Interfaz de usuario**	**Disponibilidad de librerías/documentación**	**Valoración**
Leaflet + NodeJS/Django	Combinación de la librería para generación de mapas interactivos Leaflet con un backend compatible con ejecución de código Python (NodeJS o Django directamente)	Código libre	Gran flexibilidad para crear interfaces de usuario, con cualquier estilo y funcionalidad que se desee	Muy amplio, el ecosistema web es quizá el más rico en librerías y documentación, tanto por parte de los autores de las librerías como las guías generadas por usuarios En concreto, tanto Leaflet como NodeJS (o alternativamente Django) son ampliamente utilizado y disponen de documentación de calidad disponible	Uno de los candidatos más potentes, pues es tecnología con la que estoy familiarizado y que puede potencialmente ofrecer un mejor resultado tanto desde el lado del desarrollador como el usuario. El uso de Django haría la ejecución de código Python muy sencilla y de fácil inclusión Por otro lado, habría que alojar la herramienta en algún servicio en la nube, lo cual podría añadir dificultades para integrarlo con el resto de componentes del sistema.
Frameworks de Python (PySimpleGUI, PyQt, Remi,…)	Emplear uno de los diversos frameworks que permiten generar interfaces con programas Python nativos.	Mayormente si, con la excepción de PyQT	Flexibilidad considerablemente más limitada que otras opciones, con especial mención del mapa interactivo	Generalmente amplio y suficiente. La inmensa mayoría de la documentación es la propiamente proporcionada por los autores Dispondríamos de acceso a todo el ecosistema de librerías Python.	Se trata de la opción que mejor casa la opción de generar una herramienta de escritorio con la capacidad de integrar el proyecto con el resto de los componentes. Sin embargo, la gran falta de opciones en el lado de la interfaz supone una gran desventaja, especialmente en comparación a otras opciones de la lista.
.NET	Framework de desarrollo de aplicaciones	Código libre	Similar o idéntica capacidad a	Se trata de un framework popular de	En general una herramienta muy atractiva. La documentación, como ya se

	multiplataforma de Microsoft		las tecnologías web convencionales si se trata de una aplicación web de .NET Más limitado si se trata de una opción de escritorio	Microsoft, con lo que posee documentación oficial de gran calidad, así como un gran número de soluciones de usuarios. Hay disponibles un número amplio de librerías, aunque no tienen una integración del todo plena entre la parte web y de escritorio de .NET	ha mencionado, es de muy alta calidad y la flexibilidad en el apartado web es idéntico al de otras tecnologías. Es notable, sin embargo, la dificultad para integrar Python. Existen algunas soluciones en forma de librerías, pero están disponibles principalmente en el apartado de escritorio y no son del todo sencillos de usar. Destacar la dificultad para desarrollar aplicaciones que puedan ser usadas tanto en escritorio como en web (habría que realizar dos desarrollos paralelos o decantarse por uno)
Eel + Leaflet	Se trata de una librería Python para generar de manera sencilla y sin muchas complicaciones un servidor local Python que dispense contenido web en local, así como facilitar la comunicación entre el componente web y el componente backend de Python	Código libre	La flexibilidad y capacidad coinciden con la primera herramienta descrita en la tabla.	La documentación referida a Eel no es muy extensa, pero es suficiente dado la sencillez de la librería. Por lo demás, léase este mismo apartado del primer elemento de la tabla.	Esta herramienta es esencialmente la misma que la primera opción, con el añadido de ser más sencilla en estructura, pues Eel define toda la infraestructura de intercambio de información entre Python y Javascript sin tener que generarla nosotros de cero. Adicionalmente, se integra muy bien con Pyinstaller, por lo que una build de escritorio no solo es posible, sino que apenas diverge de la versión web.

Las opciones web de la lista anterior resultaban las más prometedoras, gracias a su sencillez, familiaridad, flexibilidad e integración de Python.

Nos decantamos por el uso de Eel, que simplifica el proceso de generar un servidor Python que interactúe de manera sencilla con nuestro *frontend* Javascript.

3.4.1 - Leaflet/Eel como framework de desarrollo

Tras el anterior análisis de las herramientas disponibles, se hará a continuación una breve descripción de las dos herramientas seleccionadas para el desarrollo del proyecto.

Leaflet

Leaflet[10] es una muy conocida librería para la implementación y personalización de mapas interactivos para entornos web. Es de código libre y de uso gratuito.

Con Leaflet disponemos de una amplia gama de herramientas y opciones de personalización y configuración.

Scaracteriza por ser una librería de poco peso relativo a su funcionalidad.

En la página de la librería encontramos una detallada sección de documentación con ejemplos claros, así como una gran cantidad de preguntas de la comunidad respondidas por veteranos de la herramienta.

Podemos extender las funcionalidades de nuestro mapa mediante numerosos *plugins* creados por la comunidad. Por ejemplo, se ha incorporado el plugin Leaflet Shapefile, para permitir la carga de archivos shapefile sobre el mapa interactivo.

Eel

Tal y como lo define Chris Knott, autor de la librería, en [5]:

"*Eel is a little Python library for making simple Electron-like offline HTML/JS GUI apps, with full access to Python capabilities and libraries*."

El funcionamiento de la librería es relativamente sencillo (Ilustración 8):

[10] https://leafletjs.com/

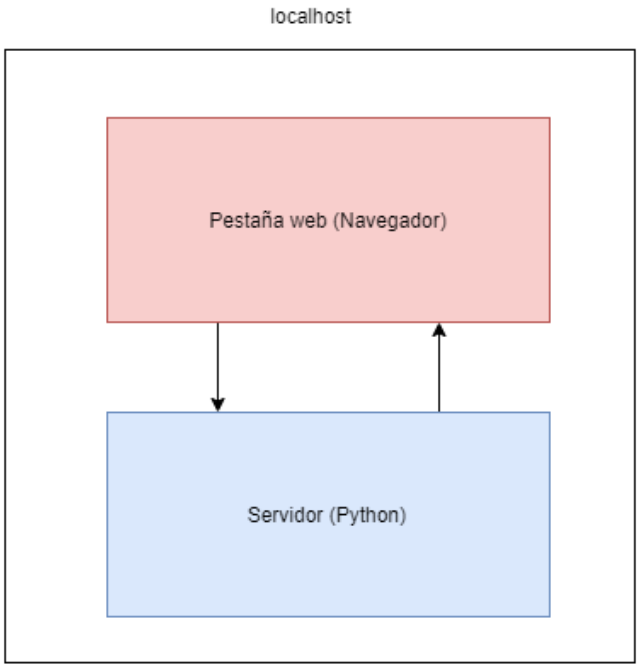


Ilustración 8- Arquitectura de EEL
Fuente: Elaboración propia

En resumidas cuentas, nos permite fácilmente comunicar un servidor Python con un cliente web generado en la ejecución del programa mediante un sistema asíncrono de peticiones. De esta manera, podemos combinar el entorno de ejecución Python con su rico ecosistema de librerías de procesado de datos con la flexibilidad y tecnologías frontend disponibles para web.

Aunque la librería no es necesaria para realizar esta infraestructura, si nos permite ahorrar tiempo, ya que nos libra de la necesidad de escribir código para el servidor y una API para interconectar cliente web y backend Python.

Se muestra a continuación (Ilustración 9) unos ejemplos reales del código para mostrar el funcionamiento.

```python
@eel.expose
def obtenerGeoJson(lng, lat, tiempoEnMinutos):
    body = {"locations":[[lng, lat]],"range":[tiempoEnMinutos*60]}

    headers = {
        'Accept': 'application/json, application/geo+json, application/gpx+xml, img/png; charset=utf-8',
        'Authorization': [redacted],
        'Content-Type': 'application/json; charset=utf-8'
    }
    call = requests.post('https://api.openrouteservice.org/v2/isochrones/driving-car', json=body, headers=headers)
    return call.json()
```

Ilustración 9- Función "obtenerGeoJson" del archivo principal Python
Fuente: Elaboración propia

Aquí vemos un fragmento del fichero principal en Python. Mediante la sentencia *@eel.expose* antes de la función, *exponemos* esta función para ser invocada desde el frontend de JavaScript.

En este caso, la función recibe una localización geográfica donde se quiere centrar una isócrona en forma de latitud y longitud, así como el tiempo en minutos que debe abarcar la isócrona deseada. Se realiza una petición HTTP mediante la librería *requests* con los parámetros proporcionados.

Se devuelve entonces el cuerpo de la respuesta para ser gestionada por el frontend.

```
/**
 * Obtener isocrona en la localización indicada con extensión
 * del tiempo proporcionado
 * @param {number} lat
 * @param {number} lng
 * @param {number} tiempo
 * @param {function} callback
 */
getIsocrona(lat, lng, tiempo, callback) {
    eel.obtenerGeoJson lng, lat, tiempo)().then((resultado) => {
        if (!resultado.type === 'FeatureCollection') {
            callback(null, resultado);
            return;
        }
        callback(resultado , null);
    }, (rejected) => {
        callback(null, rejected)
    })
}
```

Ilustración 10- Método getIsocrona de la clase "EnlaceABackend", implementada en JavaScript
Fuente: Elaboración propia

Para invocar estas funciones de Python desde el lado del *frontend* es tan sencillo como invocar al método del mismo nombre del objeto *eel,* el cual es instanciado al inicio de ejecución.

La respuesta es gestionada de manera asíncrona mediante la API de promesas de JavaScript.

Capítulo 4 – Implementación del proyecto

En esta sección se comentarán diversas cuestiones referentes a la implementación, el testeo y la documentación del proyecto como aplicación software.

4.1- Implementación

En este apartado nos concentraremos en lo referente a las distintas técnicas y distintos integrantes del programa en lo que a materia de código fuente se refiere.

4.1.1 - Los componentes del programa

Como se ha comentado anteriormente, nuestra aplicación se basa en la ejecución de un script Python. También se ha mencionado que el motivo principal para que esto sea así es para facilitar la integración de esta herramienta en un futuro ecosistema de componentes que se desarrollarán principalmente en Python, en gran parte gracias a las facilidades que el lenguaje ofrece en el ámbito de *data science.*

El script esencialmente genera un servidor local e intenta abrir una instancia de navegador Chrome donde mostrar el contenido web incluido en el programa.

De no estar Chrome instalado en el sistema del usuario, se abre en el navegador por defecto que este tenga (en el caso de Windows 10, con seguridad estará disponible el navegador Edge).

El script Python también realiza la labor de *backend,* esencialmente actuando como conexión entre la parte del navegador, la base de datos con la información de las bases y las APIs a través de las cuales se calculan las isócronas y las estimaciones de población. Pueden apreciarse de manera esquematizada las funciones que este script ejerce en la Ilustración 11.

Ilustración 11 - Esquema con las funciones del script principal
Fuente: Elaboración propia

Al iniciarse el navegador automáticamente se muestra en pantalla el contenido web. Dicha parte web consta de los siguientes componentes:

- main.html: Página html donde se encuentra el mapa Leaflet interactivo y los diferentes controles de los que dispone el usuario.

- Sim_RS.js: Código Javascript principal del cliente.

- EnlaceABackend.js: Una clase cuya principal utilidad es aislar las funciones principales del cliente de las tareas de comunicación con el *backend* Python. De esta manera el código queda mejor compartimentado y en un futuro donde se desee portar esta aplicación al contexto web en línea resultaría más sencillo.

Las **bases**, **vehículos** y **solapes** entre isócronas son tratados en el código en forma de objetos, cada uno de ellos con una clase JavaScript para representarlos. A continuación, se muestra un ejemplo de diseño de una de las clases componentes del programa (Ilustración 12).

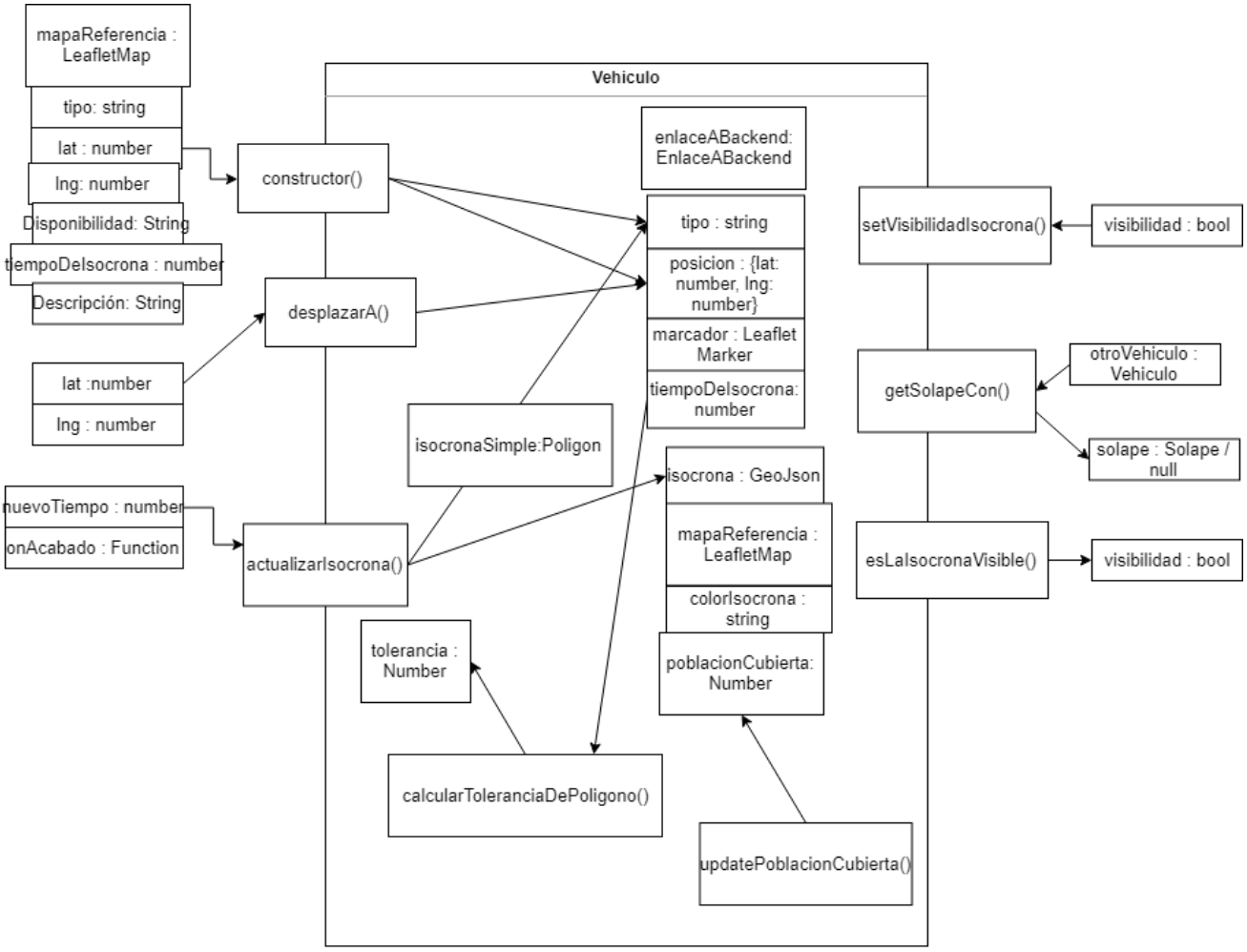


Ilustración 12 - Esquema de la clase vehículo
Fuente: Elaboración propia

La mayor parte del código del cliente web sirve para gestionar el estado de estos objetos que representan a las diferentes entidades del mapa.

Como se ha mencionado previamente, se ha implementado una clase EnlaceABackend para evitar acoplamiento entre la parte que gestiona la interfaz y las labores de contacto con el backend Python.

En una hipotética portación de esta aplicación al formato web online podría modificarse este fichero para facilitarlo.

En el siguiente esquema (Ilustración 13) puede apreciarse el resultado final, así como las relaciones que los componentes guardan entre sí:

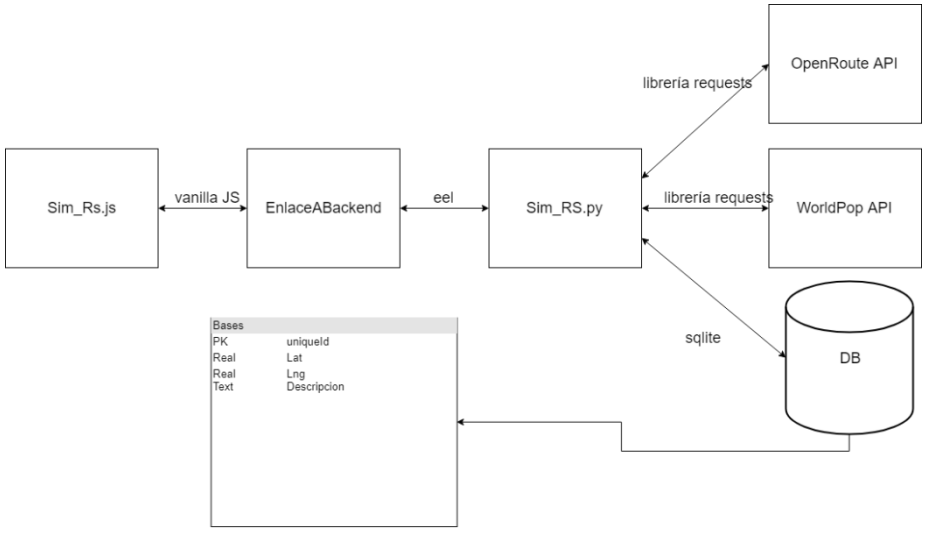


Ilustración 13 - Componentes integrantes del programa
Fuente: Elaboración propia

4.1.2 – Interfaz gráfica

En el apartado de la interfaz gráfica se ha optado por una sencilla maquetación HTML, empleando **Bootstrap**[11] como biblioteca de estilización.

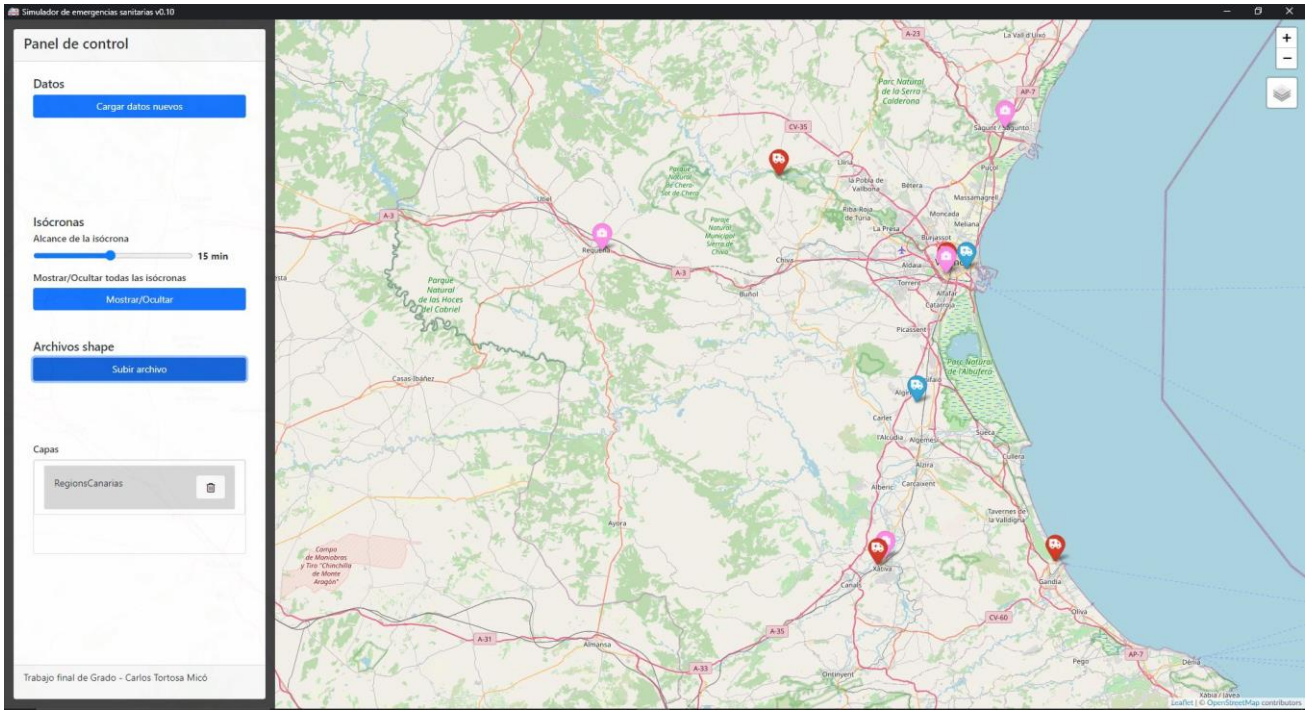


Ilustración 14 - Interfaz gráfica del programa, vista desde Chrome

[11] https://getbootstrap.com/

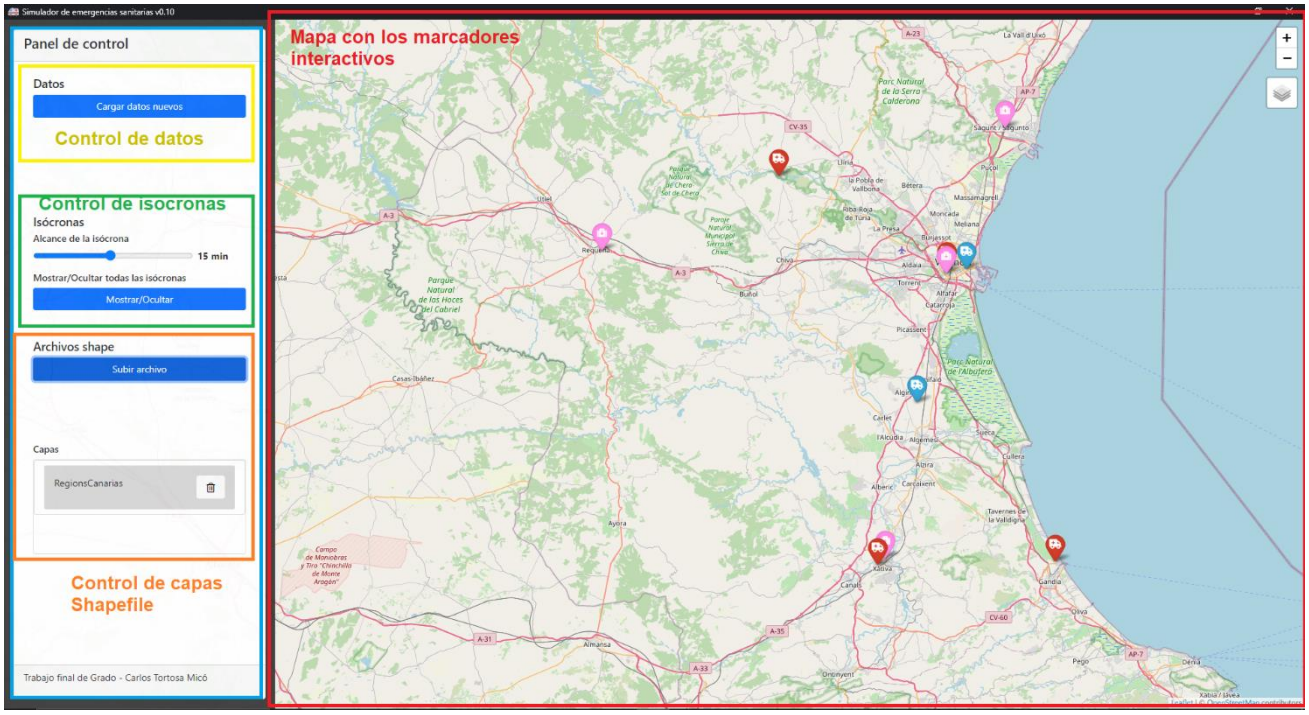


Ilustración 15 - Interfaz gráfica con las diferentes secciones remarcadas

La interfaz, representada en las ilustraciones 14 y 15, consta de los siguientes elementos:

- El mapa: Implementación de mapa interactivo basado en Leaflet donde por defecto se muestra una capa base obtenida en OpenStreetMap. La herramienta es pública, gratuita y de licencia abierta. En el mapa es donde se encuentran los marcadores donde quedan representados los vehículos y bases.

- La sección de carga de datos: Al interactuar con el botón se nos insta a seleccionar un fichero csv que contenga los datos de los vehículos. Al hacerlo, y si el formato del archivo es correcto, se generan las entidades en el mapa.

- Los controles de isócronas: Por defecto están deshabilitadas. Se habilitan en el instante que se cargan vehículos. El *slider* permite definir el tamaño de isócronas que se desea calcular, y el botón de "Mostrar/Ocultar" permite alternar la visibilidad de estas.

- Los controles de capas: Al usar el botón se nos permite cargar un archivo *shapefile* comprimido en zip. *Shapefile* es un formato empleado para almacenar información relativa a la localización, geometría y atributos de una entidad geográfica. Al cargarlo, queda representado en el mapa. La subsección de "Capas" permite eliminarlo si así se desea.

Los vehículos SVA quedan representados como marcadores rojos, mientras que los SVB quedan representados en azul. Ambos se muestran con el icono de una ambulancia. Las bases quedan representadas en rosa y con un icono de cruz médica.

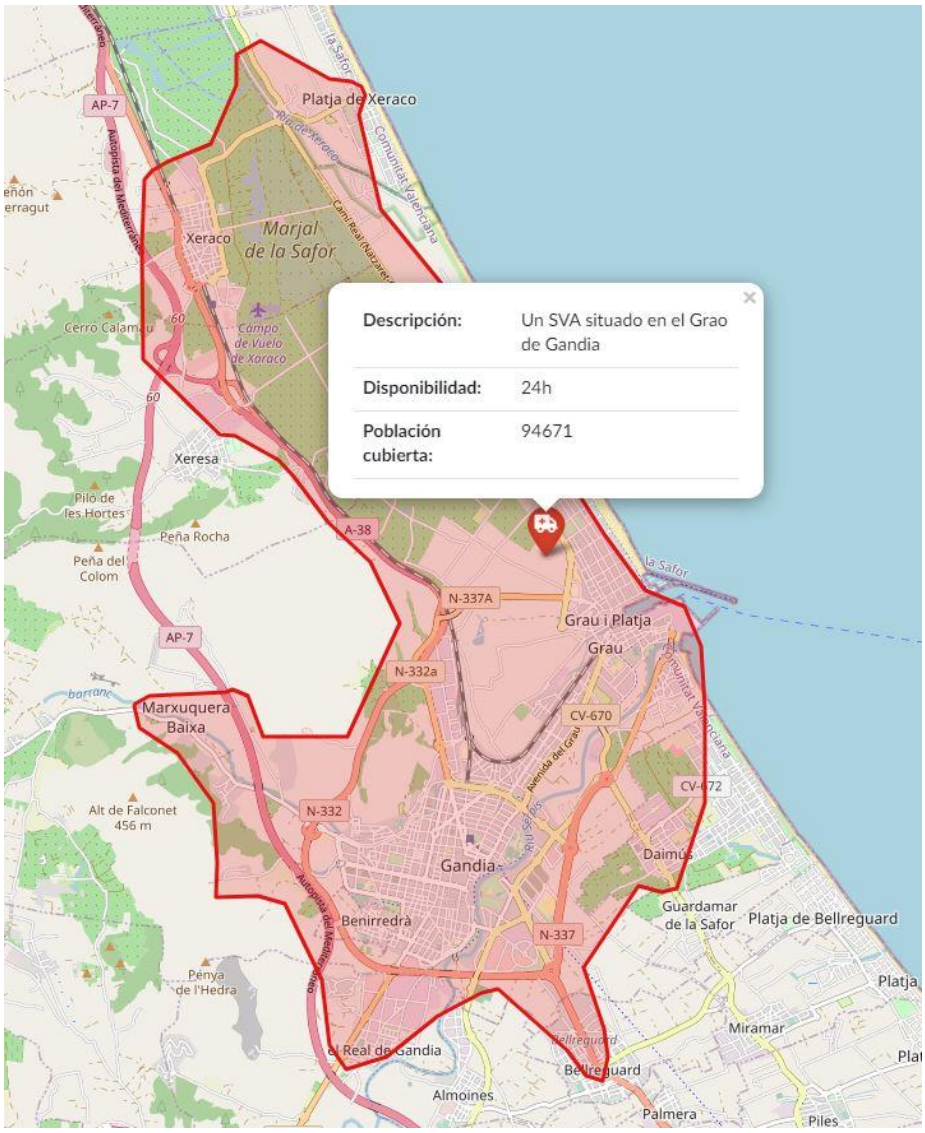


Ilustración 16 - Información desplegada de un vehículo, incluyendo la isócrona

Haciendo *click* en un marcador de vehículo (tal y como se muestra en la ilustración 16) se despliega la información de este con los siguientes campos:

- Descripción: Una descripción que se haya decidido proporcionar. En el ejemplo se describe simplemente la localización del vehículo.
- Disponibilidad: El horario de disponibilidad del vehículo.
- Población cubierta: El número de habitantes que se estima que está contenida en la isócrona del vehículo.

Simultáneamente, al clicar se realiza el cálculo y representación de la isócrona.

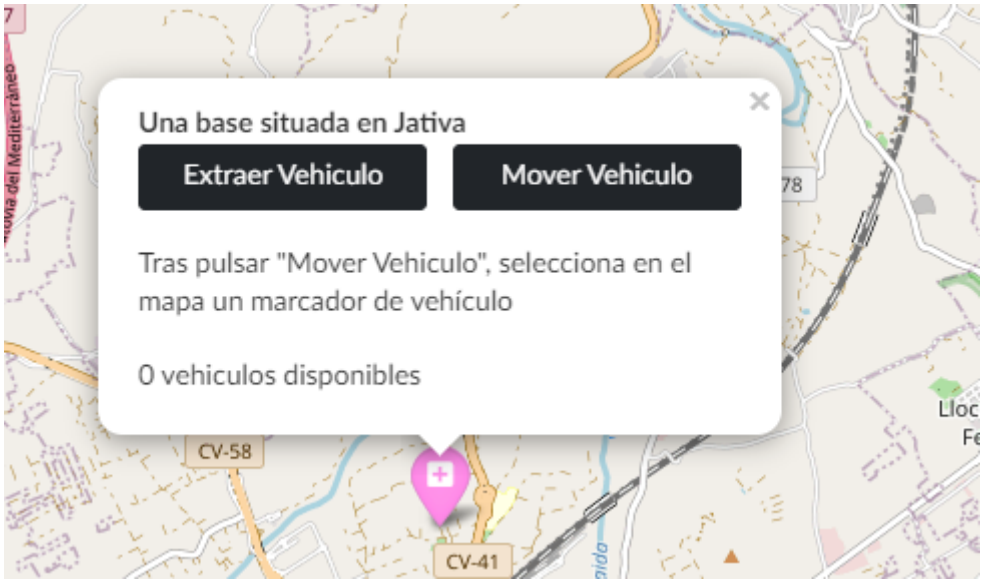


Ilustración 17 - Información de una entidad base desplegada

Al clicar una base (como muestra el ejemplo de la ilustración 17) se nos permite “Mover” o “Extraer” un vehículo. Al “Mover” y seleccionar un vehículo del mapa este queda almacenado en la base. Esto significa que desaparece del mapa y aumenta el contador de vehículos disponibles.

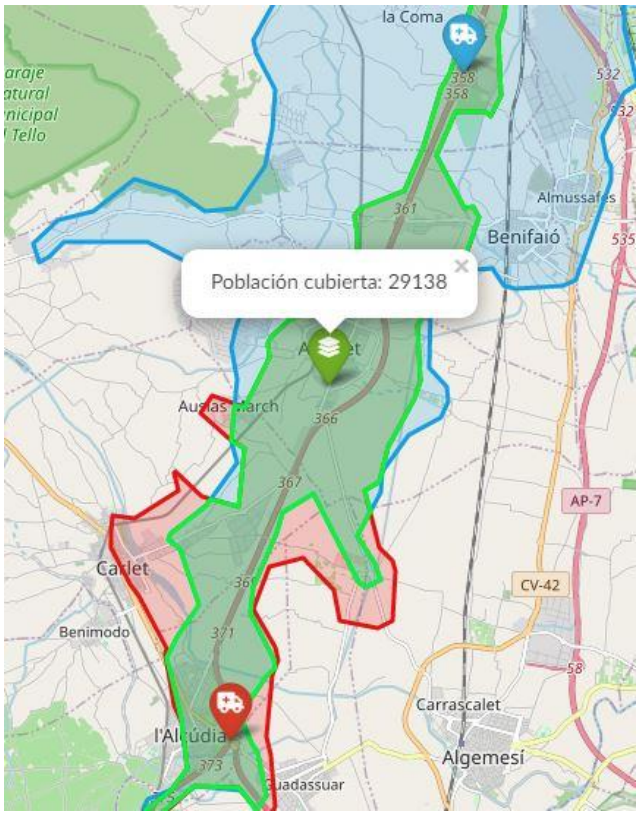


Ilustración 18 - Entidad "Solape" del mapa, en aquellas zonas donde dos isócronas comparten región

Al solaparse la isócrona de dos vehículos, se remarca como una entidad diferente en el mapa (ilustración 18). Interactuando con su marcador se nos muestra la estimación de población de esa región.

4.1.3 – APIs empleadas

La aplicación hace uso de dos servicios con APIs disponibles para dos de las funciones más críticas:

- **Cálculo de isócronas:** Basado en la API de OpenRoute.
- **Estimación de población:** Basada en la API de WorldPop.

El cálculo de isócronas puede efectuarse a través de OpenRoute de manera sencilla proporcionado los siguientes parámetros:

- Latitud de la posición
- Longitud de la posición
- Tiempo de la isócrona en segundos.

Adicionalmente se puede proporcionar el tipo de agente que efectúa el hipotético traslado. Esto quiere decir que se puede calcular la isócrona para un humano, una bicicleta, un coche, etcétera. En este caso y como es evidente se asume una isócrona para un coche (Ambulancia).

El uso de la API se encuentra limitado por motivos de banda ancha a no más de 500 peticiones por día y 20 por minuto. Adicionalmente, la extensión de las isócronas no debe superar la hora o los 120km de trayecto (en coche).

WorldPop limita a 1000 peticiones al día a aquellos usuarios que no estén identificados en su plataforma y tengan una clave de desarrollo. Actualmente esta limitación no supone un gran problema por el volumen de usuarios y peticiones realizadas.

Para realizar la estimación de la población, adjuntamos en el *query*[12] de la petición HTTP los siguientes parámetros:

- Dataset: En este caso el valor “wpgpop”, tal y como indica la documentación.
- Year: Año que representan los datos de población. Empleamos los datos del año 2020.
- GeoJson: Objeto JSON que respeta el formato estándar GeoJson y que contiene los datos de posición y geometría. Se puede serializar en texto.
- *Runasync:* Parámetro opcional para indicar si se quiere que el servidor responsa tan rápido como haya terminado de realizar la estimación, o si por otro lado se desea consultar el progreso del cálculo mediante peticiones HTTP. Para este proyecto indicamos este parámetro como falso.

[12] *Query* es el término comúnmente utilizado para referirse a la parte de una URL que contiene parámetros.

Originalmente se empleaba el servicio de la ESAC para estimar la población. Como se ha mencionado en el apartado 2.3, la precisión del servicio para escala comparables a la superficie de un municipio dejaba que desear y se optó por buscar una alternativa.

4.1.4 – Simplificación de isócronas

Durante la implementación de la API de WorldPop en el proyecto encontramos un problema técnico bastante relevante.

Puesto que la información de la geometría de la isócrona se ha de adjuntar en el propio *query* de la petición HTTP, podría ocurrir que, si la isócrona es de gran extensión o muy compleja (y por lo tanto contiene una gran cantidad de puntos), el servidor retornase un estatus de error 414. Este error es típico de los servidores Apache, que están configurados por defecto para rechazar *queries* de demasiada extensión.

Tratamos de contactar con soporte técnico describiéndoles este problema, pero no hubo respuesta. En condiciones normales hubiésemos considerado la posibilidad de omitir la funcionalidad, pero dada la importancia y el nivel de prioridad de esta se decidió indagar en alguna posible solución.

Finalmente diseñamos una solución que, a cambio de un pequeño sacrificio en precisión en la estimación, permite evitar este problema. La solución consiste en realizar la estimación sobre una versión simplificada de la isócrona.

Este proceso de simplificación se ha implementado en este proyecto mediante la inserción de la biblioteca Turf.js[13], la cual también es empleada para el cálculo de intersecciones entre isócronas.

Esta biblioteca incluye una función *simplify* que acepta como parámetros un polígono (en nuestro caso, la isócrona a simplificar) y un parámetro *e* conocido como tolerancia.

El proceso de simplificación se efectúa mediante el uso del algoritmo Ramer–Douglas–Peucker. Este algoritmo fue introducido en [6] y fue diseñado para la aproximación de curvas bidimensionales para propósitos de procesamiento de imagen. En [7] podemos encontrar una adaptación de este algoritmo para la obtención de algoritmos simplificados.

Son necesarios dos parámetros: La lista de puntos que forman el polígono a simplificar (P) y el parámetro de tolerancia *e.*

La aplicación del algoritmo resulta en un listado de puntos P', donde cada punto ha sido extraído del listado P original, cumpliendo la condición de que la distancia que ha de separar cualquier pareja de puntos sea igual o superior al parámetro de tolerancia *e.* A mayor sea el parámetro *e*, más puntos se excluirán y mayor será la simplificación.

[13] https://turfjs.org/. – Biblioteca de análisis geoespacial para navegadores y NodeJS

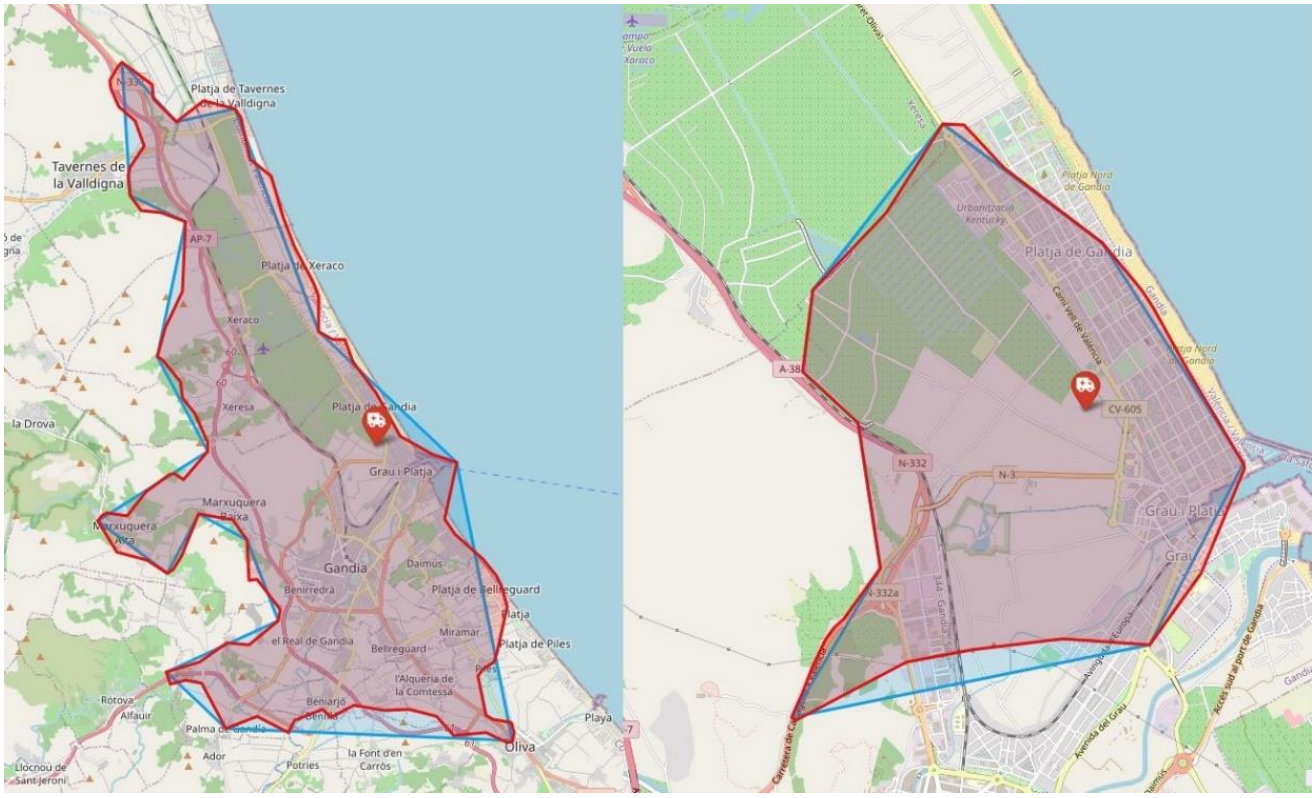


Ilustración 19 - Comparativa entre una isócrona real (rojo) y su versión simplificada (azul). Isócrona de 20 minutos a la izquierda, de 5 minutos a la derecha

En la ilustración 20 se observan dos ejemplos de simplificación de isócrona. Aunque la isócrona en rojo es la que se muestra gráficamente en la interfaz, es el polígono azul sobre el cual se realiza la estimación de población.

El programa calcula de manera dinámica el valor de tolerancia *e,* pues excluir demasiados puntos en una isócrona de por si pequeña podría suponer una gran pérdida de precisión. Con isócronas mayores nos podemos permitir excluir más puntos, pues el nivel de solapamiento entre la isócrona y la versión simplificada continúa siendo suficiente para una estimación aceptable.

Con este proceso podemos obtener un polígono suficientemente simple como para poder insertarse en nuestras peticiones HTTP y sortear el error de sobrecarga del *query*.

4.2- Documentación de código

Con la intención de continuar el desarrollo de este proyecto más allá de su vida como trabajo final de grado (ya sea de la mano del mismo autor o un contribuidor diferente) se han tomado medidas para la correcta documentación del código.

Se han realizado esquemas UML de aquellos componentes que forman parte del programa. Adicionalmente a los ejemplos del apartado 4.1.1, adjuntamos el siguiente ejemplo de la clase que representa las entidades base en el mapa:

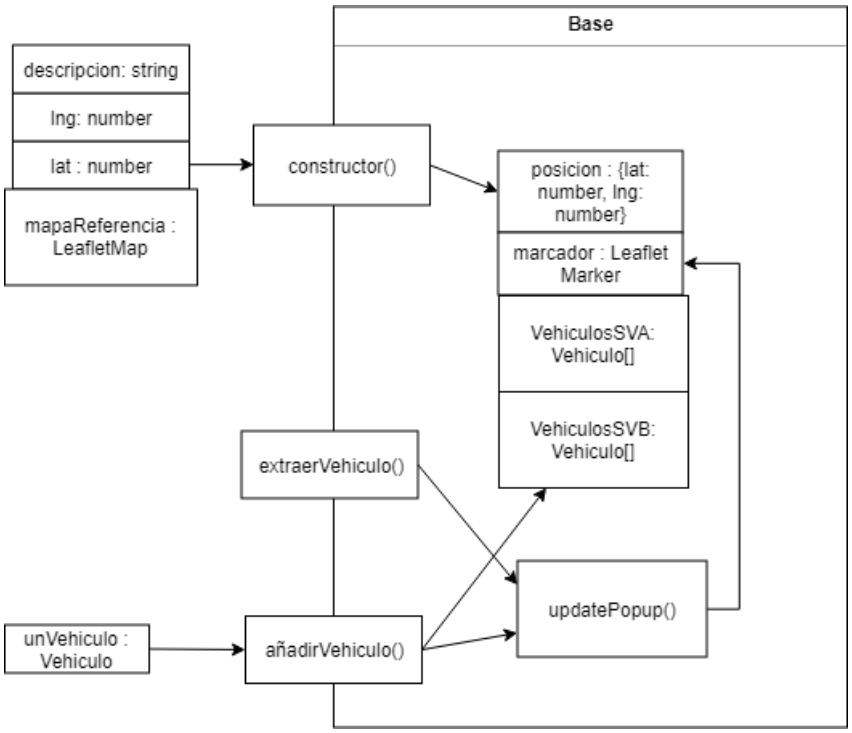


Ilustración 20 - Esquema UML de una entidad "Base"
Fuente: Elaboración propia

Para cada lenguaje empleado en el proyecto (JavaScript y Python) se ha realizado el comentado del código, siguiendo las formas estándar de cada lenguaje. En la ilustración 21 encontramos un ejemplo del código del proyecto en JavaScript.

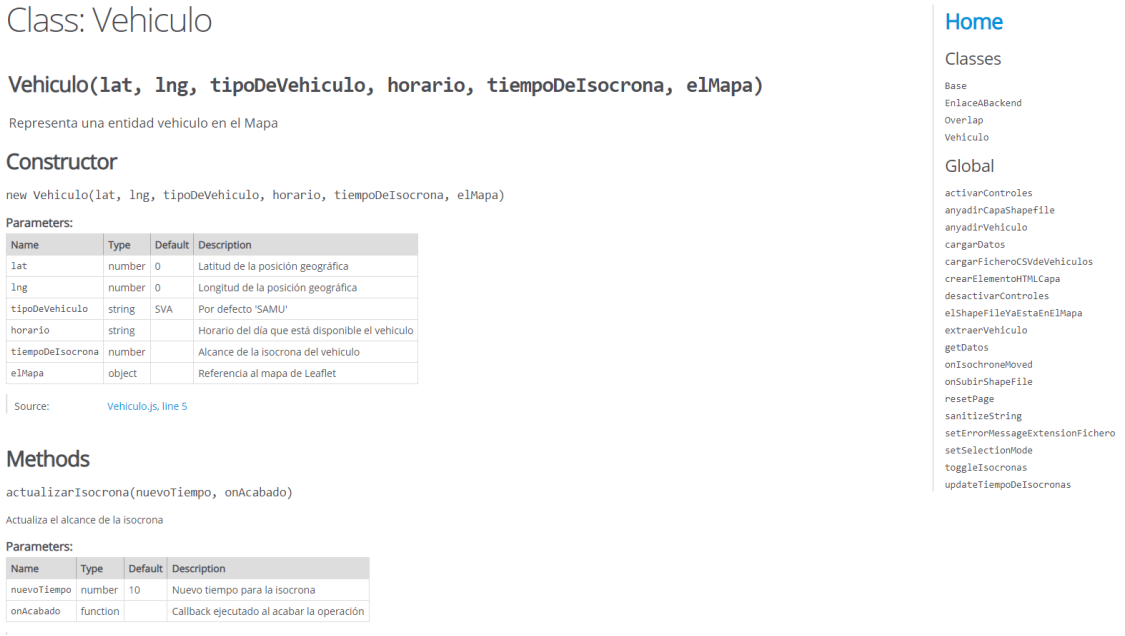
```javascript
/**
 * Obtener isocrona en la localización indicada con extensión
 * del tiempo proporcionado
 * @param {number} lat
 * @param {number} lng
 * @param {number} tiempo
 * @param {function} callback
 */
getIsocrona(lat, lng, tiempo, callback) {
    eel.obtenerGeoJson(lng, lat, tiempo)().then((resultado) => {
        if (!resultado.type === 'FeatureCollection') {
            callback(null, resultado);
            return;
        }
        callback(resultado , null);
    }, (rejected) => {
        callback(null, rejected)
    })
}
```

Ilustración 21 - Ejemplo de código JavaScript comentado.

Se emplean bloques de comentarios estandarizados como descriptores del uso de las clases y métodos. Un beneficio adicional de emplear este formato es el poder utilizar muchos de las librerías o programas que, de manera automática, generan documentación interactiva y formateada a partir de los ficheros.

Para este proyecto se ha generado documentación interactiva en formato web, la cual será entregada junto al programa y esta memoria. Se ha conseguido con la herramienta para generación de documentación JavaScript JSDoc[14].

[14] https://github.com/jsdoc/jsdoc



Class: Vehiculo

Vehiculo(lat, lng, tipoDeVehiculo, horario, tiempoDeIsocrona, elMapa)

Representa una entidad vehiculo en el Mapa

Constructor

new Vehiculo(lat, lng, tipoDeVehiculo, horario, tiempoDeIsocrona, elMapa)

Parameters:

Name	Type	Default	Description
lat	number	0	Latitud de la posición geográfica
lng	number	0	Longitud de la posición geográfica
tipoDeVehiculo	string	SVA	Por defecto 'SAMU'
horario	string		Horario del día que está disponible el vehiculo
tiempoDeIsocrona	number		Alcance de la isocrona del vehiculo
elMapa	object		Referencia al mapa de Leaflet

Source: Vehiculo.js, line 5

Methods

actualizarIsocrona(nuevoTiempo, onAcabado)

Actualiza el alcance de la isocrona

Parameters:

Name	Type	Default	Description
nuevoTiempo	number	10	Nuevo tiempo para la isocrona
onAcabado	function		Callback ejecutado al acabar la operación

Home

Classes

Base
EnlaceABackend
Overlap
Vehiculo

Global

activarControles
anyadirCapaShapefile
anyadirVehiculo
cargarDatos
cargarFicheroCSVdeVehiculos
crearElementoHTMLCapa
desactivarControles
elShapeFileYaEstaEnElMapa
extraerVehiculo
getDatos
onIsochroneMoved
onSubirShapeFile
resetPage
sanitizeString
setErrorMessageExtensionFichero
setSelectionMode
toggleIsocronas
updateTiempoDeIsocronas

Ilustración 22 - Interfaz de la documentación interactiva, página de la clase Vehículo

En la ilustración se muestra la página de la clase Vehículo como ejemplo de lo que se puede encontrar en esta documentación interactiva y el formato en el que se presenta.

4.3- Pruebas unitarias

Como es propio del desarrollo de una aplicación software, se han implementado diversas pruebas automáticas para comprobar la integridad del programa en cada incremento.

Se han preparado *tests* para comprobar dos componentes:

- Comprobación del status de las APIs
- Comprobación del funcionamiento del cliente

Se ha implementado un breve test en Python para comprobar la disponibilidad de los servicios de WorldPop y OpenRoute, independientemente de nuestro programa.

El objetivo no es otro que el de distinguir durante el desarrollo si un fallo del programa se debe a la incorrecta implementación de código o simplemente un momento de no disponibilidad del servicio.

Este test sencillamente realiza una petición a cada servicio (un cálculo de isócrona y una estimación de población, respectivamente) y comprueba la validez de la respuesta y la presencia de los datos deseados.

Se ha empleado la biblioteca Python Unittest[15] para implementar la prueba.

[15] https://docs.python.org/3/library/unittest.html

Para la comprobación del correcto funcionamiento del cliente se ha empleado la herramienta **Selenium.**

Selenium es un proyecto que acoge diferentes librerías y herramientas para la automatización de navegadores, incluyendo la automatización de interacción con elementos de la interfaz gráfica, como botones, *sliders* y otros componentes.

En un inicio se intentó implementar tests unitarios más convencionales y comúnmente utilizados en el entorno de JavaScript. Sin embargo, hubo complicaciones técnicas que dificultaron esta tarea.

La mayoría de estos problemas técnicos están relacionados con la manera en la que los componentes son importados en el script web principal. Descrito brevemente, debíamos elegir entre implementar cada clase como un módulo (con lo cual no podíamos emplearlo en el script principal) o declararlo sencillamente como una clase JavaScript tradicional importable a HTML (con lo cual no puede incluirse en un test automático sin tener que duplicar la clase).

Adicionalmente, gran parte del programa depende de bibliotecas y/o componentes que se importan durante *runtime*[16] tras ser dispensados por el servidor *eel*.

Sintetizado brevemente, necesitamos de la ejecución del servidor y la disponibilidad del apartado web para poder emplear las clases y objetos implementados en la parte del cliente.

Afortunadamente, se encontró una solución empleando Selenium tras revisar la documentación [8]. Esencialmente, la herramienta nos permite, además de interactuar con los elementos de la interfaz, ejecutar comandos JavaScript en el navegador automatizado y recibir el resultado del comando en el propio script Python.

Ejecutando nuestra aplicación, y una vez iniciado el servidor local, puede abrirse una segunda instancia de navegador (automatizado mediante Selenium) donde realizar comprobaciones sobre el estado de las entidades del mapa y los objetos del código. De esta manera podemos comprobar de manera automatizada que el código se comporta de la manera deseada.

Un beneficio adicional es la oportunidad que Selenium ofrece para correr la misma batería de pruebas en navegadores diferentes. En nuestro caso se realizan las pruebas sobre Chrome y Edge. Firefox quedó descartado debido a un bug de ejecución de scripts desde Python que no se pudo solventar. Sin embargo, si se ha podido testear sobre Firefox de manera manual.

[16] Durante la ejecución del programa

4.4- Evaluación cualitativa

Se puso a disposición de usuarios reales el uso de nuestra aplicación, y recogimos sus impresiones con una encuesta. Queríamos comprobar si el desarrollo de nuestra herramienta se dirigía en buena dirección, así como recopilar una lista de adiciones y mejoras que incorporar en el futuro.

El grupo de encuestados incluye miembros del alumnado, profesorado de la UPV y un profesional del ámbito sanitario. Las preguntas, respuestas y participantes pueden encontrarse en el anexo de este documento.

La encuesta contempla 3 aspectos principales del programa a valorar por el usuario:

- La sencillez, facilidad de uso y utilidad de la interfaz gráfica
- Una serie de cuestiones tratando la utilidad y beneficios de la aplicación como herramienta de apoyo a la decisión en materia sanitaria, así como recolección de funcionalidades que puedan añadirse en un futuro.
- Un par de preguntas más técnicas recogiendo opiniones sobre el formato de la aplicación y su estabilidad como software.

En cuestión de interfaz, los usuarios han reportado ser capaces de entender y emplear fácilmente los diferentes controles con los que manejar la herramienta.

En líneas generales, se ha apreciado el potencial de la aplicación como herramienta aplicable al mundo real. Las respuestas incluyen una serie de sugerencias de gran valor para mejorar la oferta de funcionalidades de la aplicación.

Una de las respuestas incluyó el siguiente comentario:

"*Me ha parecido fantástico el hecho de que se puede desplazar cada vehículo a cualquier punto del mapa, de esta forma se pueden simular diferentes estados del sistema y ver la cobertura desde diferentes ubicaciones. Considero el programa absolutamente útil para tomar las decisiones de ubicación y reubicación de ambulancias.*"

Una sugerencia habitual ha sido el incluir una mayor oferta de vehículos, como los TNA, o incluso vehículos no estrictamente necesarios como camiones de bomberos o vehículos policiales.

La percepción sobre el formato del programa (aplicación de escritorio con lectura de ficheros CSV) resulta suficiente para la mayoría de encuestados.

Finalmente, Los usuarios han experimentado y reportado conductas no deseadas del programa (algunos artefactos visuales relacionados con la representación de isócronas) que no resultan fáciles de reproducir en el equipo de desarrollo.

Se han recogido sugerencias de mejora para la estabilidad del programa y experiencia de usuario, que resultan valiosas teniendo en cuenta que el desarrollo de la aplicación se ha realizado sobre un solo equipo y con una conexión Internet de características concretas.

Capítulo 5 – Conclusiones

Con este capítulo finalizaremos esta memoria, realizando una serie de reflexiones sobre las buenas prácticas realizadas, los conocimientos adquiridos, los aspectos mejorables, la adquisición de prácticas transversales y propuestas con las que mejorar nuestra aplicación.

5.1 – Conclusiones y reflexiones sobre el proyecto

En esta memoria se ha expuesto la relevancia de una correcta localización de los vehículos sanitarios, de tal manera que permita a los servicios sanitarios cubrir las emergencias que puedan acontecer en una región.

Se han recopilado y sintetizado diversos conceptos relacionados con los tipos de emergencias, los vehículos que las atienden y los sistemas de apoyo a la decisión, con el fin de definir y enfocar nuestra aplicación.

Se ha descrito el proceso de generación de un producto, incluyendo las siguientes fases:

- Análisis y caracterización de un problema
- Descomposición de este problema en necesidades a cubrir
- Diseño de una solución que atiende estas necesidades
- Proceso de elección del formato de la aplicación, así como de las herramientas que más facilidades ofrecen para la implementación de la solución
- Descripción de la metodología de trabajo empleada durante el desarrollo.
- Descripción del proyecto como aplicación software, incluyendo el proceso de implementación de los diversos componentes, resultado final, problemas solventados, documentación del código y las pruebas automáticas realizadas.

Los sistemas de apoyo a la decisión orientados a las emergencias sanitarias son, ciertamente, una materia que no resultaba familiar. Ha sido necesario un trabajo de investigación con el objetivo de familiarizarse con la definición del concepto, la categorización de las distintitas herramientas de apoyo a la decisión y una serie de conceptos referentes al mundo de las emergencias los cuales eran desconocidos previos a la realización de este proyecto.

En esta fase de investigación también se ha tenido la oportunidad de llegar a conocer otras herramientas de la vida real disponibles al público. Estas aplicaciones han resultado de inmensa utilidad para definir el *kit* de tecnologías que acabarían siendo integradas en nuestro programa, así como para definir la manera con la que se representan los diferentes elementos del mapa interactivo de nuestra aplicación.

Con la contribución de los tutores de este proyecto y testimonios reales de personas que realizan la labor de optimizar el uso de los recursos sanitarios, se ha podido recopilar una lista real de necesidades que una aplicación con un objetivo como la nuestra debe cumplir.

Los profesionales involucrados en este proceso de optimización a menudo han de realizar grandes cantidades de laborioso trabajo manual para generar este tipo de

soluciones. Generar y representar isócronas para cada vehículo, calcular la población cubierta por los servicios sanitarios y generar modelos optimizados requiere una gran labor de recolección de datos, manejo de hojas de cálculos y presentación. Hemos procurado que nuestra aplicación ofrezca una serie de funcionalidades de fácil acceso que alivie la carga de trabajo de estos profesionales.

En el proceso de implementar esta herramienta se han tenido que sortear desafíos técnicos que, consideramos, nos han ayudado a crecer como desarrolladores resilientes y han derivado en la adquisición de conocimientos variados sobre los diferentes lenguajes de programación y librerías que integran el programa.

Si bien teníamos metas un poco más ambiciosas para la fecha de entrega de este proyecto, estamos satisfechos con el punto alcanzado, especialmente con las buenas opiniones que hemos recibido de usuarios reales.

Finalmente, remarcar el potencial de nuestra aplicación, la cual ha recibido retroalimentación positiva de parte de personal sanitario real pese a ser únicamente el primer componente de un proyecto mayor.

5.2 – Reflexiones sobre la adquisición y práctica de competencias transversales

En esta sección se realizará un breve comentario sobre las competencias transversales que, creemos, han sido desarrolladas durante la realización de este proyecto.

Estas competencias incluyen:

- **Comprensión e integración**: Por la adquisición de una serie de conceptos sobre emergencias sanitarias y herramientas de apoyo a la decisión que eran desconocidos y su integración en un producto real que busca solventar un problema de dicha materia.

- **Análisis y resolución de problemas**: Por el diseño de una aplicación que busca dar una solución real a un problema, tras su análisis y disección en pequeños objetivos a lograr. Este punto es aplicable también al desarrollo de la parte software, en la que se han tenido que solventar diversos problemas de carácter técnico mediante el razonamiento y el diseño de diversas soluciones.

- **Diseño y proyecto**: Por la concreción de un concepto de aplicación en un producto real mediante la planificación, la aplicación de una metodología de trabajo regular y el uso efectivo de los recursos disponibles.

- **Conocimiento de los problemas contemporáneos:** Por el proceso de familiarización con el campo de las emergencias sanitarias y el análisis y caracterización de un problema relacionado con la gestión de los recursos de los que disponen las entidades sanitarias.

- **Planificación y gestión del tiempo:** Por haber desarrollado este proyecto bajo las pautas de entrega de progreso pactadas con los tutores y una satisfactoria distribución de la carga de trabajo.

5.3 – Propuestas de mejora

Aunque este proyecto ha sido finalizado como trabajo final de grado, todavía se contemplan diversas mejoras con las que diversificar las funcionalidades de la aplicación y ofrecer una mejor herramienta.

Queda pendiente la definición del modelo de negocio del proyecto. Actualmente está en proceso la propuesta de nuestra aplicación para optar a las ayudas que oferta la Agencia Valenciana de Innovación, pero a largo plazo sería relevante definir mejor el modelo de distribución y acceso a la herramienta.

Una de las mejoras más relevantes que quedan pendientes es la integración de un modelo real de bases y vehículos por defecto al programa. Actualmente la aplicación se testea y se inicia sobre una hoja de datos que incluye una serie de bases y vehículos colocados artesanalmente, a modo de ejemplo y para comprobar la correcta funcionalidad de la herramienta.

Una funcionalidad que nos gustaría haber desarrollado en más profundidad es la carga y representación de capas *Shapefile* sobre el mapa interactivo. Aunque la base actual es sólida, solo permite cargar capas proyectadas en WGS84, y únicamente carga el aspecto visual de la capa.

El motivo principal es que preparar el programa para cualquier tipo de formato que pudiese encontrar en el banco de datos de la capa se vuelve muy complejo algorítmicamente. Para un futuro nos gustaría introducir una primera versión de esta funcionalidad con una estructura de datos concreta.

Actualmente la herramienta puede emplearse un número limitado de veces diarias antes de no poder utilizarse (Unas 500 consultas de isócronas al día, máximo 20 por minuto y 1000 estimaciones de población). Para un solo cliente y con propósitos de testeo ha resultado suficiente, pero en un futuro donde la herramienta este distribuida a gran escala esto sería insuficiente.

OpenRoute impone una limitación diaria máxima para asegurarse de que su infraestructura puede proporcionar servicio a todos los usuarios, pero también proporciona las herramientas para que estos implementen su proprio *backend* del servicio si desean trabajar sin restricciones. Es una opción relevante a tener en cuenta para el futuro de nuestra aplicación.

Durante la evaluación de la herramienta por los usuarios, se compiló una lista de funcionalidades que resultaron interesantes de cara a una futura implementación:

- Localizar los vehículos en tiempo real, es decir, que su marcador del mapa se actualizara en tiempo real siguiendo la posición de las ambulancias.
- Aumentar el número de intersecciones que se pueden tener en pantalla
- Representar sobre el mapa la suma de todas las isócronas (cobertura total de población)
- Aumentar la oferta de tipos de vehículos representables, sin limitarse necesariamente a vehículos sanitarios (bomberos, coches policiales…)
- Representación de isócronas con origen en centros sanitarios.

Bibliografía

[1] M. Á. V. García, «Resolución de un problema real de relocalización de los vehículos de emergencia sanitaria,» 2020.

[2] D. J. Power, «A Brief History of Decision Support Systems,» 10 March 2007. [En línea]. Available: https://dssresources.com/history/dsshistory.html.

[3] G. A. Gorry y M. S. Scott-Morton, «A framework for information systems,» de *Sloan Management Review*, 1971, pp. 56-79.

[4] S. Alter, «Decision Support Systems: Current Practice and Continuing Challenges,» Addison-Wesley: Reading,MA, 1980.

[5] S. H. Williams, «Eel,» 2017-2020. [En línea]. Available: https://github.com/ChrisKnott/Eel.

[6] R. Urs, «An iterative procedure for the polygonal approximation of plane curve,» *Computer Graphics and Image Processing,* pp. 244-256, 1972.

[7] A. Mutlu, F. Göz y O. Akbulut, «IFIT: an unsupervised discretization method based on the Ramer-Douglas-Peucker algorithm,» 17 12 2019. [En línea]. Available: http://journals.tubitak.gov.tr/elektrik/.

[8] B. Muthukadan, «Selenium with Python,» 2018. [En línea]. Available: https://selenium-python.readthedocs.io/.

[9] M. Martínez García y J. Recasens Aloy, «Dotación sanitaria. Ciclo formativo: Emergencias Sanitarias,» Secretaría General Técnica. Centro de Publicaciones. Ministerio de Educación, 2011.

[10] T. H. Peucker y D. H. Douglas, «Algorithms for the reduction of the number of points required to represent a digitized,» pp. 112-122, 2006.

[11] The Python Software Foundation, «DocsPython,» 2021. [En línea]. Available: https://docs.python.org/3/.

[12] Refsnes Data, «W3Schools,» 1999-2021. [En línea]. Available: https://www.w3schools.com/.

[13] Mozilla, «MDN Web Docs,» 2005-2021. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript.

[14] V. Agafonkin, «Leaflet,» 2010-2021. [En línea]. Available: https://leafletjs.com/.

[15] J. Sutherland, «The Scrum Papers: Nult, Bolts, and Origins of an Agile Framework,» Cambridge, 2012, p. 217.

[16] The Python Software Foundation, «Python - Style Guide for Python Code,» 2001-2021. [En línea]. Available: https://www.python.org/dev/peps/pep-0008/.

[17] Google, «Google JavaScript Style Guide,» 2021. [En línea]. Available: https://google.github.io/styleguide/jsguide.html.

[18] NASA, «Socioeconomic Data and Applications Center,» 1997-2021. [En línea]. Available: https://sedac.ciesin.columbia.edu/.

[19] S. Eom y E. Kim, «A survey of decision support system applications(1995–2001),» *Journal of the Operational Research Society (2006) 57, 1264–1278,* 2005.

[20] X. Hao, M. Wang y X. Liu, «Application of Big Data Visualization in Urban Planning,» *IOP Con. Ser: Earth Environ. Sci.,* 2020.

Anexo

Encuesta cualitativa

[Y[Yulia Karpova Kyrlova, Doctorando

[A] Anónimo, Médico Coordinador,

[F] Mª Fulgencia Villa Juliá, Profesora e Investigadora UPV

1. ¿Qué opinión te merece la interfaz gráfica? (Se comprende lo que hace cada elemento, los controles están situados de manera cómoda, el estilo es adecuado...)

[Y] Se entiende perfectamente el uso de cada elemento de la interfaz. El estilo es adecuado y permite diferenciar cada elemento. La posición de los botones de control es muy cómoda y hay que resaltar su sencillez para el uso. No sé si sería posible cambiar el tono de las capas shape, que salen en un azul muy fuerte, a algo más suave.

[A] Todo correcto, el borrado a veces se cuelga.

[F] Es clara y se comprende todo.

2. ¿Te resulta sencillo interactuar con el mapa? (El manejo de los marcadores es intuitivo, la información se muestra con buen formato, no hay errores visuales en las isócronas). De no ser así, indica que mejorarías.

[Y] La interacción con el mapa resulta ser muy sencilla y muy intuitiva. Los botones +/- claramente indican el aumento/disminución del zoom y la manita permite realizar el desplazamiento por el mapa. El botón para seleccionar los elementos visibles en el mapa es muy importante, porque permite la adaptación de la información a cada usuario.

[A] Si

[F] Es sencillo interactuar. La única dificultad es el borrado de una simulación.

3. El mapa permite distinguir las posiciones de vehículos SVA, SVB y bases estacionarias. ¿Considerarías útil incluir algún tipo de servicio más? De ser así indica cual.

[Y] Creo que la gran ventaja de la herramienta es la posibilidad de incluir otros tipos de vehículos: policías, bomberos... Incluso, si la usara una empresa de logística con una flota heterogénea, también le sería útil. A efectos del actual estudio que se realiza en colaboración con la Conselleria de Sanitat, sería útil incluir las ambulancias convencionales (TNAs), las que realizan transporte programado.

[A] Me gustaría que se incluyesen TNA y la identificación de las unidades

[F] La posibilidad de visibilizar más de una isócrona por vehículo.

4. El programa permite cargar archivos CSV donde el usuario ha definido el tipo, localización y descripción de cada vehículo previamente. ¿Crees que es una funcionalidad útil o sería suficiente cargar un estado predeterminado todas las veces?

[Y] Dado que el programa se puede usar en diferentes áreas, sería más cómodo utilizar un archivo para cargar las posiciones iniciales de cada vehículo. Además, es posible que el mismo usuario en diferentes momentos quiere partir de diferentes estados iniciales.

[A] No lo sé

[F] Creo que es una funcionalidad útil

5. Por contra al caso anterior, la posición de las bases no las altera directamente el usuario, debido a su naturaleza estacionaria y permanente. ¿Crees que sería una funcionalidad útil poder personalizar su localización?

[Y] Creo que debería haber una opción de cargar la posición de las bases, porque diferentes usuarios tienen diferentes bases. Y otra vez, el mismo usuario puede necesitar valorar diferentes combinaciones de bases en uso. La funcionalidad de poder personalizar localización de las bases le daría al programa mayor versatilidad.

[A] No así está bien

[F] No, yo creo que así está perfecta

6. Con el estado y funcionalidades actuales del programa, ¿Crees que resulta útil como herramienta de apoyo a la decisión? Si no es así, indica que adición o corrección podría cambiar esto.

[Y] Me ha parecido fantástico el hecho de que se puede desplazar cada vehículo a cualquier punto del mapa, de esta forma se pueden simular diferentes estados del sistema y ver la cobertura desde diferentes ubicaciones. Considero el programa absolutamente útil para tomar las decisiones de ubicación y reubicación de ambulancias.

[A] Si, puede ayudar a la movilidad de las unidades

[F] Creo que está fenomenal y es muy útil. De una forma muy sencilla se puede visibilizar el impacto de las decisiones.

7. ¿Qué beneficio crees que esta herramienta puede aportar a una persona responsable de gestionar la distribución de los vehículos sanitarios? (Alivio de carga de trabajo, acceso a decisiones mejor informadas, permite generar discusión con otros responsables...)

[Y] La posibilidad de visualizar el estado de las ambulancias puede confirmar una buena decisión o, por el contrario, puede disuadir de tomar una decisión errónea. Esta herramienta es muy útil para la toma de decisiones objetivas, basadas en el análisis numérico, que algunas veces faltan.

[A] Permite visibilizar y simular qué pasaría si se moviesen unidades o se incorporasen nuevas de una forma muy visual

[F] Permite simular de una forma sencilla y muy visual el impacto de decisiones sobre ubicación de vehículos.

8. ¿Crees que el formato en programa es cómodo? ¿O sería preferible algún otro? (Por ejemplo, tener la herramienta disponible en formato web)

[Y] En este momento veo muy cómodo el formato del programa.

[A] Es adecuado

[F] Así lo encuentro perfecto.

9. Comenta sobre la estabilidad del programa. Esto es, si no han habido errores, los que han habido los ha gestionado correctamente, o por el contrario, has encontrado errores que dificultaban usar el programa.

[Y] Creo que sería necesario hacer un reseteo cada vez que se carguen datos nuevos para que desaparezcan las isócronas que se habían pintado.

[A] A veces se bloquea el borrado de las isócronas

[F] No es cómodo borrar.

10. Comenta cualquier aspecto que te resulte relevante (opcional)

[Y] Gran trabajo!!! Enhorabuena!!!!

[F] Me parece una herramienta muy sencilla para ayudar a tomar decisiones complejas de una manera muy visual.