UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Modelos de Mezcla

Presentado por Humberto Marijuán Santamaría
en Universidad de Burgos — 22 de septiembre de 2022
Tutores: Luis R. Izquierdo y José Manuel Galán

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Luis R. Izquierdo y D. José Manuel Galán, profesores del departamento Ingeniería de Organización, área de Organización de Empresas.

Expone:

Que el alumno D. Humberto Marijuán Santamaría, con DNI 71757204X, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Modelos de Mezcla.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 22 de septiembre de 2022

Vº. Bº. del Tutor:

D. Luis R. Izquierdo

Vº. Bº. del co-tutor:

D. José Manuel Galán

Resumen

En este documento se expone el proceso de desarrollo de una herramienta de cálculos estadísticos sobre *Mixing Models* (modelos de mezcla). Este tipo de herramientas suelen resolver problemas en el ámbito de la ecología, permitiendo calcular las contribuciones de diferentes fuentes a una determinada mezcla a partir de trazadores. Algunos ejemplos de aplicaciones son el cálculo de la dieta de un animal a partir de la composición de sus sus heces, o la estimación de la procedencia geográfica de diferentes tipos de tierra que componen un sedimento.

Se ha creado una aplicación web que utiliza las librería GLPK para resolver problemas de maximización y minimización por medio de una interfaz en *Typescript*. Esta aplicación web permite visualizar y exportar los resultados. El código de la aplicación web esta desarrollado en *Angular* y utiliza otros lenguajes habituales en las aplicaciones web como HTML y CSS. La aplicación esta internacionalizada a los idiomas Español e Inglés.

La aplicación permite la carga de datos a partir de archivos .csv e incluye un vídeo tutorial como guía de uso. Por último se realizó el despliegue del proyecto en Netlify con url: https://mixingmodels.netlify.app

Descriptores

Aplicación web, single-page aplication, composición dieta, ecuaciones lineales con restricciones.

Abstract

This document presents the development process of a statistical calculation tool based on so-called *Mixing Models*. This type of tool is often used to solve problems in the field of ecology, allowing to calculate the contributions of different sources to a certain mixture using tracers.

A web application has been created that uses the GLPK library to solve maximisation and minimisation problems by means of a Typescript interface, allowing the results to be visualised and exported. The code of the web application has been developed in *Angular* and uses other common languages in web applications such as HTML and CSS. The application is internationalised in Spanish and English.

The application allows the loading of data from .csv files and includes a video tutorial as a user guide. Finally, the project was deployed in Netlify with url: https://mixingmodels.netlify.app

Keywords

Web application, single-page application, diet composition, linear equations with constraints.

Índice general

Índice de figuras

Índice de tablas

Introducción

Este proyecto comienza con la necesidad de crear una aplicación web que permita estimar diferencias y composición de la dieta de vertebrados herbívoros. Este problema está definido en el artículo: *"Using n-alkanes to estimate diet composition of herbivores: a novel mathematical approach"* [10] en el cual se plantea la resolución de un sistema de ecuaciones con restricciones, con el objetivo de calcular la composición de la dieta de diferentes especies de herbívoros. Este problema pretende responder a la pregunta de ¿Cuáles son las proporciones máximas y mínimas de cada componente de la dieta?.

Los modelos de mezcla son utilizados para estimar contribuciones de diferentes fuentes a una mezcla. Estos modelos suelen requerir datos que actúan como marcadores que caractericen rasgos (comúnmente químicos) de las fuentes y las mezclas. Los modelos de mezcla están muy presentes en el ámbito de la ecología, utilizando isótopos estables como trazadores para evaluar la contribución de los componentes de la dieta (fuentes) a la dieta de los consumidores (mezcla). Aunque este problema tiene otras muchas aplicaciones: movimiento de animales, origen de los contaminantes, transferencia de nutrientes entre ecosistemas, huellas de la erosión de los sedimentos y evaluar las relaciones entre depredadores y presas desde perfiles de ácidos grasos [9, 16, 3, 14, 7].

En este trabajo de fin de grado, se ha diseñado una aplicación web que permite resolver sistemas de ecuaciones con restricciones, para lo cual se ha hecho uso de algunas librerías *open source* que han facilitado la resolución del problema (librería GLPK [4] y algunos paquetes Node).

1.1. Estructura de la memoria

La estructura de este documento viene formada por siete capítulos:

- **1. Introducción:** Una descripción breve del marco donde se encuentra. Y la estructura de la memoria y los anexos.
- **2. Objetivos del proyecto:** hemos definido los requisitos de este proyecto, distinguiendo entre generales, funcionales y técnicos.
- **3. Conceptos teóricos:** Exposición conocimientos para facilitar la compresión del proyecto.
- **4. Técnicas y herramientas:** Listado de metodologías y herramientas utilizadas en el proyecto
- **5. Aspectos relevantes del desarrollo del proyecto:** Se ha resumido el proceso de desarrollo del proyecto, destacando aspectos claves del mismo.
- **6. Trabajos relacionados:** Se ha realizado una búsqueda de trabajos similares al software presentado.
- **7. Conclusiones y Líneas de trabajo futuras:** Conclusión personal del proyecto y posibles vías de trabajo futuras.

1.2. Estructura de los anexos

- **Plan de Proyecto Software:** Planificación temporal y estudio de viabilidad económica y legal.
- **Especificación de Requisitos:** Objetivos y requisitos iniciales del proyecto.
- **Especificación de diseño:** Recoge lo referente al diseño de la aplicación, incluyendo diseño de datos, procedimental y arquitectónico.
- **Documentación técnica de programación:** Explica la estructura de los directorios del repositorio GitHub, junto con manuales orientados al programador (instalación, compilación y ejecución del proyecto).
- **Documentación de usuario:** Es una guía de uso de la aplicación paso por paso con capturas de la interfaz.

Objetivos del proyecto

En este apartado se definen los requisitos del proyecto distinguiendo entre objetivos generales, requisitos de software y objetivos técnicos.

2.1. Objetivos generales

El objetivo principal de este proyecto es implementar una aplicación web que resuelva problemas de programación lineal dentro del contexto de *Mixing models* (modelos de mezcla). Esta aplicación deberá ser accesible desde cualquier sistema operativo y móvil. La aplicación web se implementará como una Single-page application. La vista de la interfaz deberá ser simple e intuitiva.

2.2. Objetivos de funcionalidad

- **Usabilidad y diseño:** La aplicación web debe ser fácil de usar para el usuario, y fácil de entender para los usuarios que estén familiarizados con el problema.
- **Accesibilidad:** La web debe ser accesible por la mayor parte de navegadores en los principales sistemas operativos.
- **Importación y Exportación:** Se debe permitir la descarga de las soluciones y la importación de los datos del problema.
- **Internacionalización:** La aplicación debe incorporar todos los textos tanto en Español como en Inglés. Asimismo, deberá permitir la

incorporación de idiomas adicionales sin que esto conlleve a realizar cambios en el código.

2.3. Objetivos técnicos

- Utilización de la metodología Scrum para el seguimiento y el control del proyecto, siendo necesario trabajar con herramientas de gestión de proyectos y de control de versiones durante el desarrollo.
- Selección y aprendizaje de un lenguaje de programación para desarrollar una aplicación web .
- Creación de una aplicación web que realice peticiones web a una API .
- Generación de ficheros .csv a partir de los datos resultantes. Exportaciones de los cálculos intermedios, de la solución junto con los datos del problema y un último documento que tenga la estructura permitida en la carga de datos. La visualización de los informes debe ser similar a visual de la aplicación web.
- Carga de datos por medio de ficheros .csv.
- Utilización de librerías para resolver problemas de programación lineal.
- La aplicación deberá contener un apartado de guía, que tendrá que ser accesible desde la aplicación en cualquier momento.
- Interfaz gráfica simple y limpia.
- Utilización de herramientas de análisis de código.

Conceptos teóricos

En este capitulo se han recogido algunos conocimientos básicos sobre los *Mixing Models* que son necesarios para enmarcar el problema. Además tenemos como objetivo explicar el problema que se resuelve con la aplicación.

3.1. Mixing Models

Los modelos de mezcla (*Mixing Models*) son herramientas estadísticas que utilizan biotrazadores para estimar las contribuciones de diversas fuentes a una mezcla. Estas herramientas son usadas en muchos problemas. A continuación se expondrán algunos ejemplos donde se usan los *Mixing Models*:

La dieta en la ecología

La dieta en ecología (fuentes = presas, mezcla = dieta de un consumidor). En el ejemplo de la Figura 3.1 se usan los *Mixing Models* para calcular las contribuciones de diferentes presas del entorno (p. ej. mamíferos marinos, salmones, ciervos) a la dieta de un depredador (p. ej. Lobo) a partir de la composición de sus heces. En el artículo [2] se resuelve este problema usando MixSir y se obtuvieron resultados robustos. Por último, Jackson et al. (2009) [2] propone añadir parámetros de error adicionales al ámbito de la mezcla .

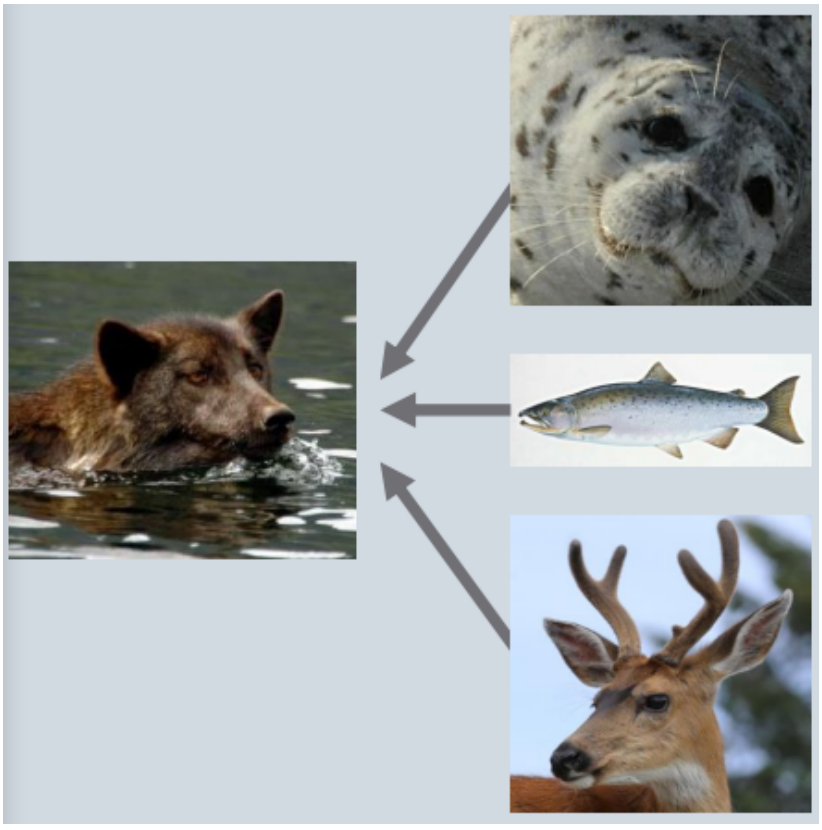


Figura 3.1: Ejemplo dieta en ecología

Movimiento en ecología

Los modelos de mezcla también se utilizan en ecología para estimar movimientos de animales (fuentes = regiones, mezcla = animales que pueden desplazarse entre regiones). Estos modelos son usados por ecologistas para calcular composiciones de comunidades y la biodiversidad a nivel de sitio terrestre de todo el mundo.

Para más información artículo *"The PREDICTS database: a global database of how local terrestrial biodiversity responds to human impacts"* [8].

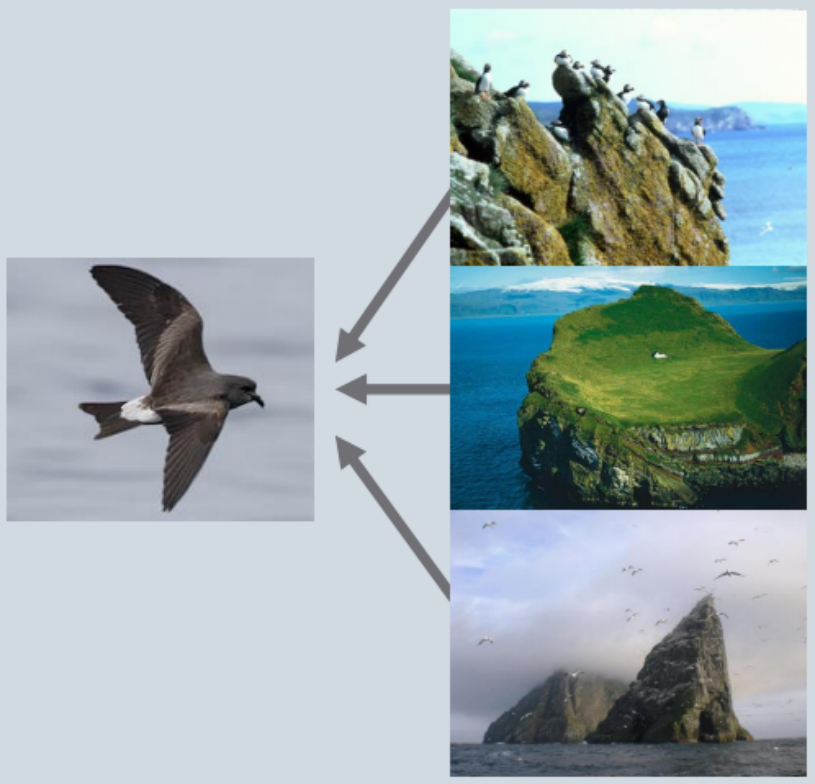


Figura 3.2: Ejemplo Movimiento en ecología

Sedimentos en sistemas fluviales

Sedimentos en sistemas fluviales (fuentes = terrenos aguas arriba, mezcla = sedimentos aguas abajo). El objetivo de los sistemas de agricultura regenerativa [13] es aumentar la calidad del suelo y la biodiversidad de las tierras de labranza, favoreciendo la creación de productos agrícolas nutritivos de forma rentable.

artículo: https://pubmed.ncbi.nlm.nih.gov/30812003/

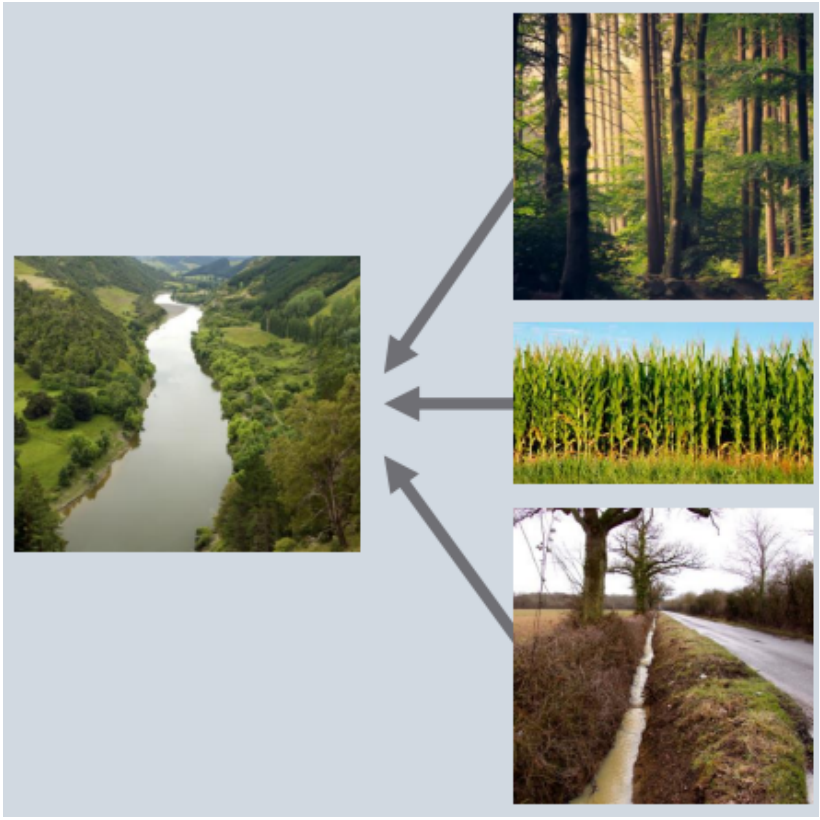


Figura 3.3: Ejemplo Sedimentos en sistemas fluviales

3.2. Cómo funcionan los *Mixing Models*

Comenzamos explicando un problema sencillo. El problema se plantea dadas dos fuentes posibles de alimentos (Source 1, Source 2) y un consumidor (ver fig. 3.4). Nuestro objetivo es identificar la dieta del consumidor a partir de la composición de sus heces. Para ello, se hace uso de biotrazadores o marcadores, que se encuentran presentes en las fuentes y no se alteran en el proceso de digestión. De esta forma, los biotrazadores nos permiten realizar estimaciones de la cantidad de cada una de las fuentes que puede haber ingerido un consumidor.

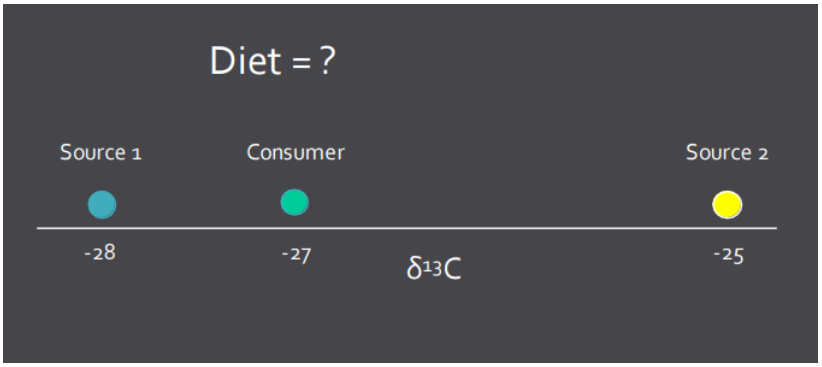


Figura 3.4: Linear *Mixing Models* - Problema

Para simplificar la explicación supondremos que la masa se conserva durante el proceso de digestión, por ello partimos de la ecuación $p_1 + p_2 = 1$, donde p_i denota la masa de fuente (*source*) i por cada unidad de masa de heces. Como suponemos que la masa se conerva, la suma de las proporciones de las fuentes es igual a 1. Tenemos también una segunda ecuación que refleja la conservación de la masa del biotrazador $\delta_{13}C$ a través del proceso de digestión: $Consumer = p_1 * S_1 + p_2 * S_2$, donde $Consumer$ es la masa de biotrazador en las heces, y S_i denota la concentración de biotrazador en la fuente i.

En el caso de la figura 3.5 solo teníamos un eje / biotrazador / marcador, por lo que solo considerábamos una ecuación. En el figura 3.6 tenemos dos ejes para los biotrazadores: $\delta_{13}C$ y $\delta_{15}N$, además de una fuente más (S_1, S_2, S_3). En consecuencia, podemos usar las siguientes ecuaciones:

$$Consumer_C = p_1 * S_{1C} + p_2 * S_{2C} + p_3 * S_{3C}$$

$$Consumer_N = p_1 * S_{1N} + p_2 * S_{2N} + p_3 * S_{3N}$$

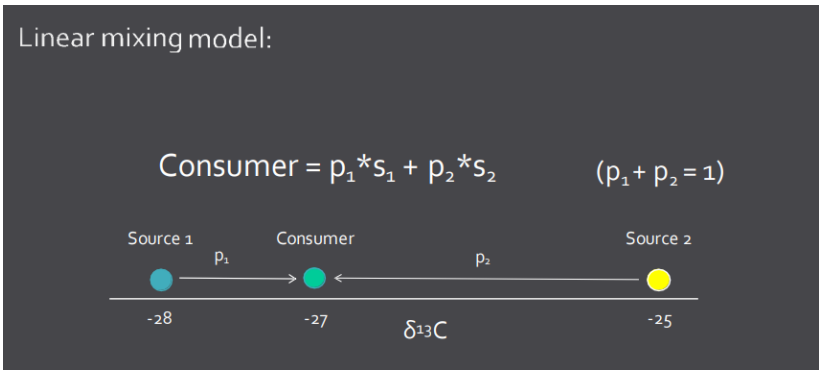


Figura 3.5: Linear *Mixing Models* - Ecuaciones con un marcador

$$p_1 + p_2 = 1$$

Siendo S_{iN} la concentración del segundo biotrazador ($\delta_{15}N$) en la fuente i, y ese valor se multiplicara por la proporción de la fuente i que se ha ingerido.

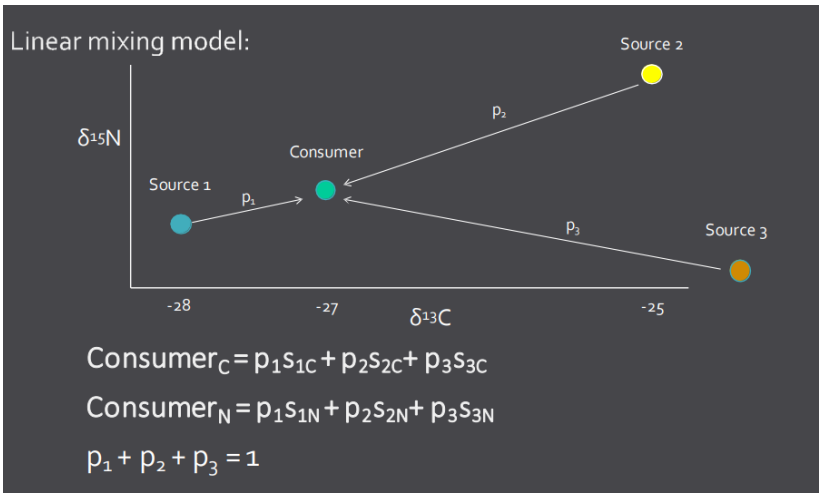


Figura 3.6: Linear *Mixing Models* - Ecuaciones con dos marcadores

En el caso de mi aplicación, se diseñó sin imponer la condición de conservación de la masa. Esto ha ofrecido un mayor realismo a los cálculos de la aplicación, ya que en general la masa de las heces no coincide necesariamente con la suma de las masas que se ingieren (p. ej. los alimentos ingeridos suelen tener más concentración de agua que las heces).

3.3. Problema n-alcanos

En esta sección hemos resumido el problema planteado en el artículo *"Using n-alkanes to estimate diet composition of herbivores: a novel mathematical approach"* [10], que es el punto de partida del proyecto.

Los n-alcanos son hidrocarburos que se encuentran en las cutículas de las plantas, y que pueden ser usados para estimar la composición de la dieta de herbívoros a partir de la composición de sus heces. Conocer la composición de la dieta de los herbívoros es importante para comprender su ecología y alimentación, lo cual es útil para medir sus efectos sobre la vegetación y los ecosistemas.

El modelo usa programación lineal para estimar el mínimo y el máximo de las proporciones de cada planta de la dieta.

Las especies de plantas se caracterizan por diferentes concentraciones de n-alcanos. Los marcadores químicos recuperados en las heces pueden ser utilizados para cuantificar las plantas ingeridas por un animal.

Supongamos que d es el número de n-alcanos en planta y hez. Cada hez de la muestra puede ser considerada un **punto** en el espacio d-Euclideo, que describe la concentración de cada uno de los d n-alcanos. Cada especie de planta puede ser interpretada como un vector en este espacio (Figura 3.7).

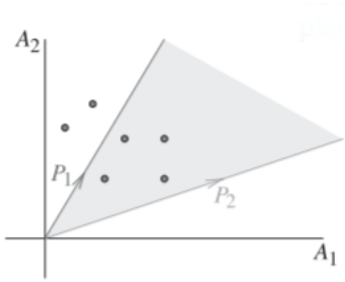


Figura 3.7: Representación de dos alcanos (A_1,A_2), dos especies de plantas (vectores P_1 y P_2), y seis muestras de heces (puntos de la figura). El área sombreada es el cono bidimensional generado por los vectores P_1 y P_2.

Las dietas son combinaciones lineales, es decir, las sumas ponderadas de los vectores que representan las especies vegetales. El peso o coeficiente de cada vector define la cantidad correspondiente de la especie en la dieta. Cada coeficiente dividido por el sumatorio de los coeficientes, da la proporción de cada planta en la dieta. Si $P_1, P_2, ..., P_p$ son los d-vectores que representan p

plantas, cada vector indica la concentración de n-alcanos en esa especie, la combinación lineal con coeficientes $c_1, c_2, ..., c_p$ es $c_1P_1 + c_2P_2 + ... + c_pP_p$. Las combinaciones lineales con coeficientes negativos son *"dietas sin sentido"*. Por lo tanto, debemos centrar nuestra atención en las combinaciones lineales que tienen coeficientes no negativos. Este conjunto de combinaciones lineales no negativas forma el cono generado por esos dos vectores. En la Figura 3.7 el cono generado por los vectores P_1 y P_2 es la región sombreada. Usamos C para denotar el cono generado, es decir:

$$C = c_1P_1 + c_2P_2 + ... + c_pP_p, \text{ con } c_1, c_2, ..., c_p >= 0$$

La estimación de dietas a partir de pruebas fecales $F_1, F_2, ...F_q$ toma una muestra F y busca sí:

(I) F pertenece al cono C o

(II) F es un punto fuera de C.

Esto es determinado resolviendo el sistema de ecuaciones linear

$$Ax = F.$$

Donde A es una matriz $d \times p$ cuyas columnas son los vectores $P_1, P_2, ..., P_p$ que representan las plantas. Si todos los componentes de la solución x son no-negativos, estaremos en el caso $_{(I)}$, y x especifica la composición correspondiente de especies en las heces. Si hay un componente negativo, estamos en la situación $_{(II)}$, indicando que son *"dietas sin sentido"*. Esto puede resultar de errores en la medición de las concentraciones de n-alcanos, o la existencia de plantas no incluidas en la muestra. Una posible solución es encontrar un vector no negativo de coeficientes que se aproximen a la x. Un posible enfoque consiste en:

(III) identificar un punto F_0 en el cono C considerado *"similar a"* F, y

(IV) usando F_0 en lugar de F, proceder resolviendo como en $_{(I)}$ el sistema de ecuaciones.

Normalmente, la proyección de F en C, es decir, el punto en C más cercano a F, es seleccionado para ser F_0. En la figura 3.7 hay dos puntos en el caso $_{(II)}$. Cada uno será remplazado por su proyección en C, por lo

que se encontrará definido por el vector P_1. La dieta estimada del animal se calcula finalmente a partir de las soluciones obtenidas de las muestras fecales $F_1, F_2, ..., F_q$. La dieta son las soluciones del sistema de ecuaciones lineal.

En este ejemplo estamos asumiendo que el número de n-alcanos es igual al número de especies de plantas, es decir, que A es una matriz cuadrada (no-singular). Si hubiera más n-alcanos que especies, la situación no cambiaría (Figura 3.8). Pero si hubiese más plantas que n-alcanos (Figura 3.9) la situación cambia considerablemente. Ahora cualquier combinación lineal no-negativa de vectores P_1 y P_3 define un nuevo vector P contenido en el cono generado por estos vectores. Cualquier punto F encontrado en el cono formado por P y P_2 es una combinación lineal no negativa de P (en consecuencia de P y P_3) y P_2. Al ser infinitos los vectores P definidos a partir P y P_3 tales que F esta dentro del cono generado en P y P_2, el sistema puede tener un número infinito de soluciones no negativas. O lo que es lo mismo, la concentración de n-alcanos en las muestras fecales puede resultar de infinidad de mezclas de plantas diferentes.

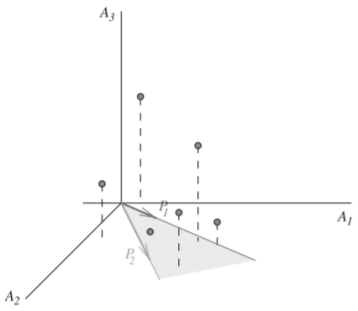


Figura 3.8: Representación de tres n-alcanos (A_1,A_2,A_3), dos especies de plantas (vectores P_1 y P_2), y seis muestras de heces (puntos de la figura), dando lugar a un cono bidimensional generado por los dos vectores de plantas.

Por tanto, siempre que el número de especies supere al número de n-alcanos se suele enfocar el problema agrupando las especies en categorías. Sin embargo, no existe un claro criterio satisfactorio para decidir cómo agrupar las especies (cómo definir una partición d al conjunto de vectores $P_1, P_2, ..., P_p$) ni cómo ponderarlas dentro de cada grupo.

Para resolver este problema se sugiere un enfoque alternativo que usa números arbitrarios de especies de plantas y n-alcanos.

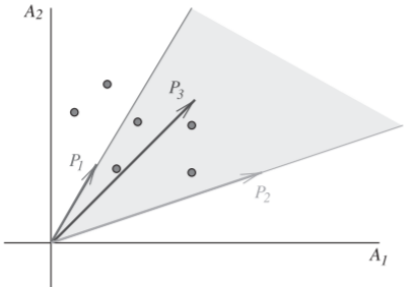


Figura 3.9: Representación de dos n-alcanos (A_1,A_2), tres especies de plantas (vectores P_1, P_2 y P_3), y seis muestras de heces (puntos de la figura). El área sombreada es el cono bidimensional generado por los vectores P_1, P_2 y P_2..

Partiendo de $P_1, P_2, ..., P_p$ vectores de d componentes que representan p plantas, donde d es el número de n-alcanos usados, y C el cono generado por los vectores, denotamos $F_i = 1, 2, ..., q$ la proyección en C del punto en d, y definimos ϕ como el conjunto de muestras fecales *"correctas"*.

Definimos una dieta factible, aquella solución no negativa del sistema lineal $Ax = F$, para cualquier punto de F en ϕ, donde A es $d \times p$ matriz con columnas $P_1, P_2, ..., P_p$. Hay muchas dietas factibles. No obstante, el conjunto de dietas está acotado por **límites superiores e inferiores** para cada uno de los coeficientes, la programación lineal es una herramienta adecuada para abordar algunas de las siguientes cuestiones sobre la dieta:

- ¿Cuáles son las proporciones máximas y mínimas de cada especie en la dieta?
- ¿Cuáles son las proporciones máximas y mínimas de cada grupo en la dieta?
- ¿Cuál es la proporción mínima de una planta, en concreto en aquellas dietas que cumplen un requisito (por ejemplo, los niveles de ingesta de nitrógeno o tanino contenido)?

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas, bibliotecas, lenguajes y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Se comentarán los aspectos más destacados de cada opción y se han añadido referencias bibliográficas que incluyen más información.

4.1. Herramientas de metodología ágil

Git

Git es un software de control de versiones (free and open source) diseñado por Linux Torvalds, centrado en la eficiencia, confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones con una gran cantidad de archivos de código fuente.

La característica diferencial con el resto de SCM[1] es su modelo de ramificación, permitiendo trabajar en ramas locales, para después poder realizar operaciones de fusión y eliminación del código con facilidad.

Para más información: https://git-scm.com/

GitHub

Github es una plataforma destinada a alojar proyectos para el control de versiones Git y desarrollo colaborativo software.

[1] **Gestión de configuración de Software** (Software Configuration Management, SCM) es una especialización de la gestión de la configuración de todas las actividades en el sector de desarrollo de software. SCM trata y controla: La elaboración de código fuente por varios desarrolladores simultáneamente

Hemos utilizado GitHub debido a su facilidad de cara a trabajar con la metodología SCRUM, permitiéndonos dividir el proceso de desarrollo en períodos de trabajo o "sprints", y realizar análisis posteriores que nos mostrarán posibles errores de planificación y aciertos en cuanto al desarrollo del proyecto. Otro factor es su facilidad de interacción con otras herramientas.

Repositorio del proyecto en GitHub: https://github.com/humbertoms99/DIET_COMPOSITION

ZenHub

Para realizar el seguimiento de los desarrollo del proyecto se ha utilizado ZenHub, que es un extensión de Chrome que se integra de forma nativa en la interfaz de usuario de GitHub, y nos ofrece un seguimiento de los requerimientos de nuestros repositorios. Nos permite realizar planificaciones junto con un seguimiento y posterior análisis. También incluye filtrado por etiquetas, agrupación de problemas, visualizar posibles bloqueadores, entre otras opciones.

Conviene destacar la facilidad con se crean los gráficos de ejecución, seguimiento de la velocidad e informes de versión. Estos se analizarán en los anexos de este proyecto.

Para más información: https://www.zenhub.com/

4.2. Lenguajes de programación

HTML

Es un lenguaje de marcado de hipertexto dedicado a la elaboración de código utilizado para estructurar y desplegar páginas web. HTML se escribe en forma de etiquetas rodeadas por corchetes angulares (<,>,/). Hemos utilizado este lenguaje en el proyecto de front-end para realizar las vistas de nuestra web.

Typescript

Es un lenguaje fuertemente tipado basado en JavaScript. Se diferencia respecto a JavaScript en que añade tipos estáticos y objetos basados en clases.

PHP

Es un lenguaje de código abierto adecuado para desarrollo web y que puede ser incrustado en HTML. Lo que distingue a PHP es que el código se

ejecuta en el servidor, al contrario que en lenguajes como Javascript, que se ejecutan del lado del cliente.

Para más información: https://www.php.net/manual/es/

CSS

Es un lenguaje de diseño gráfico o lenguaje de estilos para definir la presentación de un documento estructurado escrito en lenguajes de marcados (HTML o XML). Es uno de los lenguajes base para Webs; con él podemos alterar la fuente, color, tamaño y espaciado del contenido.

JSON

Es un formato de texto utilizado en el intercambio de datos. Se trata de un subconjunto de notación de objetos de JavaScript. Este tipo de formato es el esperado al hacer peticiones a la API desde nuestro front en Angular, permitiendo enviar estructuras de datos que pueden ser leídas con facilidad.

4.3. Patrones de diseño

MVC

MVC (Modelo-Vista-Controlador) es un patrón de diseño de software utilizado para implementar interfaces de usuario, datos y lógica de control. Su característica principal es la separación de código en tres partes:

1. Modelo: contiene la representación de los datos y lógica de negocio.
2. Vista: diseño de la interfaz gráfica de usuario; contiene la información que se envía al cliente.
3. Controlador: Actúa como intermediario entre Modelo y Vista, gestionando y transformando los datos.

En el caso particular del desarrollo web, el modelo suele corresponder al propio *"model"* de una tabla de base de datos, el código suele estar escrito en Javascript, y la vista en HTML / CSS.

4.4. Frameworks y servicios

Visual Studio Code

Es un editor de código fuente que se ejecuta en Windows, macOS y Linux. Incluye soporte para la depuración control integrado de Git. Algunas características que ofrece el framework:

1. IntelliSense, que es un conjunto de funcionalidades de edición de código como sugerencias de métodos y propiedades de objeto, información de parámetros y sus tipos.

2. Depuración directa desde el editor. Visual Studio Code ofrece depuración de código con puntos de interrupción, pilas de llamadas y una consola interactiva.

3. Comandos git incorporados, permitiendo realizar revisiones en el propio framework, pudiendo descartar cualquier servicio SCM alojado.

4. Una gran cantidad de extensiones que facilitan el rápido desarrollo y visualización del código.

5. Implementación y posibilidad de alojar sitos en React, Angular, Vue, Node, Python entre otros, además de almacenar y consultar datos relacionados.

Para más información: https://code.visualstudio.com

Postman

Postman es una plataforma de API que permite a los desarrolladores testear sus APIs. Nos ofrece la posibilidad de simular peticiones HTTP request a través de una interfaz gráfica adaptada al usuario, por la cual podremos observar el estado de la respuesta y de los datos que se obtienen, posibles mensajes de validaciones.

En la Figura 4.1 se puede observar la interfaz de Postman, en la cual podemos organizar nuestras peticiones en Colecciones; a la derecha encontramos la url a la que se realiza la petición y el tipo operación. En la parte central nos permite introducir parámetros (hemos comprobado que ofrece la opción de introducir textos JSON como parámetros). También nos ofrece la opción de configurar los headers y pasar un token en caso de nuestra API trabaje con un login previo.

Al lanzar la petición, retorna la repuesta en la parte inferior de la interfaz. En la esquina superior derecha se encuentra el status , es decir, si se ha realizado correctamente o si ha llegado a un estado fallido (status 200: OK, status 4XX: Error en la petición al cliente, status 5XX: Error en el servidor).

El *body* de la respuesta puede ser consultado en los siguientes formatos de texto: JSON, XML, HTML, Text o Auto, según nuestra elección.

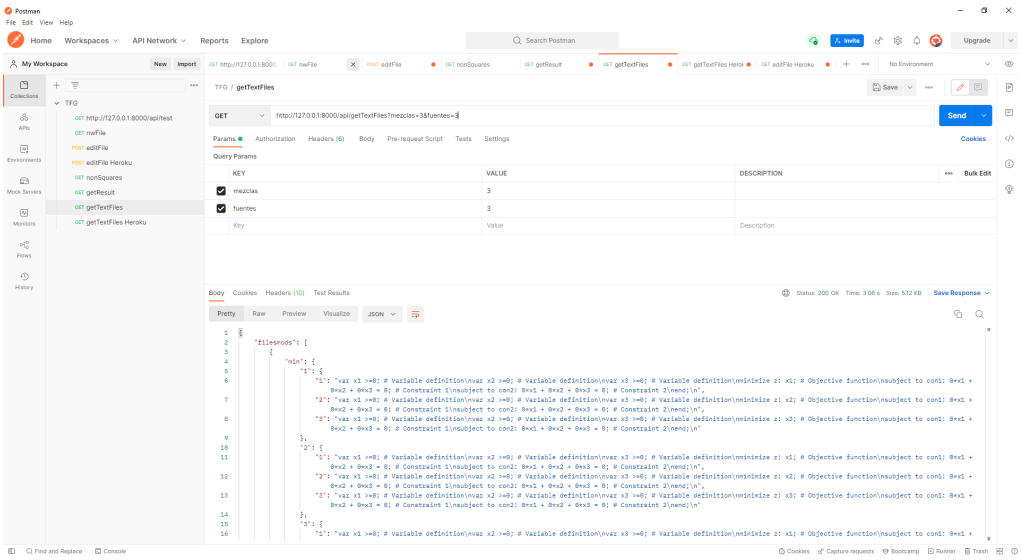


Figura 4.1: Petición Postman

Para más información: https://www.postman.com/

Debian

Debian es un sistema operativo libre desarrollado por la comunidad a través de la colaboración voluntaria. Hemos utilizado este subsistema a modo de terminal Linux para poder instalar la librería GLPK. Hemos seleccionado este sistema debido a compatibilidad y facilidad de instalación desde Windows, que es el sistema operativo donde se han realizado los desarrollos del proyecto.

Para más información: https://rincondelatecnologia.com/subsistema-linux-en-windows-10-que-es-para-que-sirve-y-como-instalarlo/

Angular

Angular es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear aplicaciones web *Single Page Application*. Angular trabaja con el patrón de diseño MVC al ofrecer una separación clara entre las vistas y las funciones que realizan las operaciones con los datos, separando el lado del cliente del lado servidor.

Angular está formado por un conjunto de herramientas para la interfaz de nuestra aplicación web; destaca su capacidad de adaptación permitiendo modificar sus características según el flujo de trabajo lo requiera, además de tener facilidad para incorporar otras librerías

Para más información: https://www.udemy.com/course/angular-fernando-herrera

Angular Translate

Angular translate es un modulo de Angular, que nos ha permitido internacionalizar la web a través de traducciones dinámicas. Su funcionamiento es por medio del pipe *translate*, que busca el texto introducido en la carpeta *i18n/**.json* y muestra su traducción correspondiente. En la figura 4.2 se realiza una traducción de la línea 26 al inglés, añadiendo el pipe **translate** e incorporando el texto en el archivo de traducciones *en.json* (Línea 3).

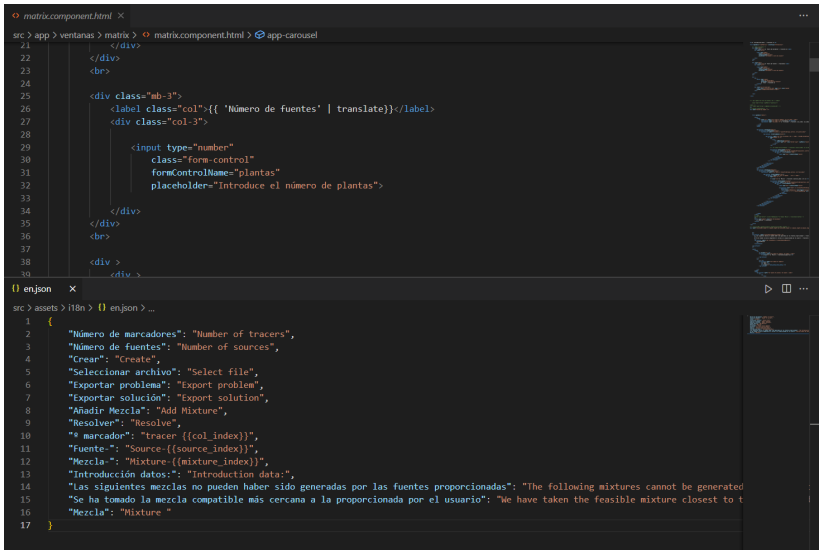


Figura 4.2: Ejemplo Angular Translate

Angular Material

Angular Material es un modulo de Angular, este modulo ofrece una biblioteca de componentes de interfaz de usuario (UI), los desarrolladores añaden estos componentes al modulo de Angular Material que a su vez es utilizado por el proyecto. Estos componentes nos ofrecen interfaces de usuarios elegantes y reutilizables, algunos componentes pueden ser botones, select, checkbox, table, Dialog, entre otros muchos.

Para trabajar con los componentes ofrecidos se ha seguido su documentación: https://material.angular.io/components/categories

Bootstrap

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Incluye diseños plantilla para componentes habituales de las web, al igual que Angular Material. Una de sus características es su diseño adaptativo (*Responsive Design"*), que consiste en adaptar el diseño de la aplicación a los distintos tamaños de pantallas, permitiendo así acondicionar nuestras interfaces a los posibles cambios en el tamaño de pantalla.

Además ofrece la posibilidad de incluir esta librería por via CDN, que consiste en añadir un link en el archivo *index.html* y carga automáticamente los estilos de bootstrap en la aplicación.

Node.js

Node.js es un entorno de ejecución para JavaScript construido con V8, basado en el motor de JavaScript de Chrome, con E/S de datos en una arquitectura orientada a eventos, y que utiliza un único hilo de ejecución asíncrono.

Para más información https://nodejs.org/es/

npm

Npm es un sistema de gestión de paquetes por defecto para Node.js. Coloca los módulos en su lugar para que Node pueda encontrarlos y gestionar los conflictos eficientemente.

Npm es muy configurable algunas de sus funcionalidades más comunes son la publicación, búsqueda, instalación y desarrollo de programas node.

Con vista a la instalación de librerias, npm tiene dos modos:

- modo global: npm instala los paquetes en *prefix/lib/node_modules* y los bins en *prefix/bin*.
- modo local: npm instala los paquetes en el directorio del proyecto actual. Los paquetes en la ruta *./node_modules*, y los bins en ./bin.

Heroku

Heroku es una plataforma como (PaaS) de computación en la Nube que soporta distintos lenguajes de programación. Heroku es propiedad de Salesforce.com. y ofrece servicios gratuitos para el despliegue proyectos.

Se ha utilizado para realizar el despliegue del back-end escrito en php en un servidor.

Netlify

Netlify es una empresa informática en la nube con sede en San Francisco que ofrece alojamiento a servicios para aplicaciones web y sitios estáticos.

En nuestro caso hemos hecho uso del paquete Starter gratuito para uso individual, y hemos desplegado el front-end en Angular, que llama a la API.
.

4.5. Librerias

glpk-5.0

GLPK (GNU Linear Programming Kit) es un paquete destinado a resolver problemas de programación lineal, programación de enteros mixtos y otros problemas relacionados. Es un conjunto de rutinas escritas en ANSI C y organizadas en forma de biblioteca invocable.

Se ha usado esta librería en el back-end para resolver problemas de programación linear mediante comandos que se ejecutan en una terminal Linux y tienen como resultado la generación de archivos .mod. Posteriormente veremos el análisis de estos archivos y datos requeridos para realizar el cálculo.

Para más información: https://www.gnu.org/software/glpk/

libtsnnls-2.3.4

Libtsnnls es un paquete de node que resuelve el problema de Fast Combinatorial Non-negative Least Squares. Se ha realizado la instalación con su comando de instalación npm *"npm i ml-fcnnls"* en el proyecto front y se ha usado para calcular las proyecciones de los puntos en el cono formado por las fuentes.

Para más información: https://www.npmjs.com/package/ml-fcnnls/v/1.0.0

4.6. Otras Herramientas

Latex: TexStudio

TexStudio es un editor de Latex de código abierto y Multiplataforma con una interfaz similar a Texmaker, que proporciona un soporte moderno de escritura, ofrece corrección ortográfica, plegado de código y resaltado de

sintaxis. Ha aportado una revisión de la sintaxis de los documentos escritos en Latex.

Para más información: https://www.texstudio.org/

Mendeley

Mendeley es un gestor de referencias que se utiliza para administrar y compartir trabajos de investigación y la generación de bibliografías para artículos académicos. Se ha utilizado su aplicación de escritorio *Mendeley Reference Manager*.

Para más información: https://www.mendeley.com/?interaction_required=true

Draw.io

Draw.io es un software de dibujo gráfico multiplataforma gratuito y de código abierto desarrollado en HTML5 y JavaScript. Su interfaz permite crear diagramas de flujo, wireframes, diagramas UML, organigramas y diagramas de red. Se ha trabajo desde su versión online, y se ha elegido como herramienta debido a su facilidad de uso y la cantidad de opciones de diseño que ofrece.

Overleaf

Overleaf es un editor de textos Latex online que se utiliza para publicar documentos científicos, Se ha usado para realizar la documentación de este proyecto. Algunas ventajas respecto a otros editores es su guardado en nube automático y permite auto compilación del código lo que nos proporciona una mayor agilidad a la hora de documentar.

Para más información: https://es.overleaf.com/learn

4.7. Uso de herramientas

En la tabla 4.1 podemos ver un resumen visual de las herramientas que hemos usado en cada apartado del proyecto.

- **Planificación y metodología**.
- **APP Angular**: aplicación de tipo single-page application, que se divide en componentes con archivos TS / HTML / CSS / TEST por componente.

- **API**: Proyecto que contiene estructura modelo-vista-controlador (en este caso solo se ha trabajado con controladores, al haber considerado que no era necesario guardar datos en bases de datos y por tanto no ser necesario el modelo).
- **Memoria**: documentos memoria y anexos escritos en LaTeX.

Herramientas	Planificación	App Angular	API	Memoria
Git	X			
GitHub	X			
ZenHub	X			
HTML		X		
Typescript		X		
PHP			X	
CSS		X		
JSON		X	X	
Visual Studio Code		X	X	X
Postman		X	X	
Debian			X	
Angular		X		
Angular Translate		X		
Angular Material		X		
Bootstrap		X		
Node.js		X	X	
npm		X	X	
Heroku			X	
Netlify		X		
GLPK		X	X	
libtsnnls		X		
TeXStudio				X
Mendeley				X
Draw.io				X
Overleaf				X

Tabla 4.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, la exposición del ciclo de vida, y detalles de mayor relevancia en las fases de análisis, diseño e implementación.

5.1. Inicio del proyecto

En una primera reunión se realizó una primera discusión sobre el artículo [10], para entender los objetivos reales del proyecto.

A partir de este momento se buscaron opciones para llevar a cabo el desarrollo de una web que realizara operaciones de programación lineal para calcular las composiciones de la dieta de grupos de animales (más adelante se entendió que realmente esta implementación permitía resolver otros muchos problemas dentro del marco de los *Mixing Models*). Las dos grandes alternativas a la hora de desarrollar la web fueron React vs Angular.

5.2. Selección React vs Angular

Con el objetivo de ordenar y aclarar diferencias entre React y Angular se ha realizado la tabla 5.1.

Característica	React	Angular
¿Que es?	Javascript framework	Javascript biblioteca para el desarrollo de interfaces gráficas
Tipo de Webs	Web y móvil, Single- y Multiple-page applications	Web y móvil, Single- y Multiple-page applications
Aprendizaje	Largo y difícil	Rápido y simple si se conoce Javascript previamente
Comunidad	Grande	Grande
Rendimiento	Optimizado con detección de cambios	Optimizado con virtual DOM
Lenguajes	JavaScript, TypepScript	JS ES6+, JSX script
Estructura de la APP	Fijo y complejo, componentes basados en MVC	Flexible, componentes basados en vistas
Directivas	Incomprensibles sin conocimiento previo en Angular	Fáciles de entender con conocimiento en Javascript
Vinculación de datos	Bidireccional, datos mutables	Unidireccional, datos inmutables
Herramientas	Aptana, Sublime Text, Visual Studio, Angular CLI, Angular Universal, Jasmine, Protractor, Karma	Sublime Text, Visual Studio, Atom, Create React App (CLI), Next.js framework, Enzyme, Hest, React-unit
Añadir librerias en Javascript en el código	Posible	No posible

Tabla 5.1: Comparativa entre React y Angular. Fuente: Elaboración propia

Inicialmente, el lenguaje más atractivo era React por su facilidad de aprendizaje y uso. Un punto que nos preocupaba tanto a mí como a los tutores en cuanto al diseño de la web, era que fuera Single-page-application, opción que permiten los dos lenguajes. Por parte de los lenguajes de programación, se partía de cero en los dos casos, al yo no tener conocimientos previos en Javascript ni en Typescript. En cuanto a la estructura de la APP, el MVC ha sido estudiado en algunas asignaturas durante la carrera, habiéndole dado más importancia que modelos basados únicamente en vistas. Respecto a las herramientas, se conocían con anterioridad Sublime Text y Visual Studio, que son comunes en los dos lenguajes y no tienen una gran importancia en cuanto al uso exclusivo de un lenguaje, al ser los dos editores de textos.

En este caso, hubo un punto más a tener en cuenta, y es que durante este año he estado realizando prácticas en una empresa, y el lenguaje con el que se ha trabajado ha sido Angular, de modo que para reducir tiempos de formación y trabajar en un lenguaje un poco más conocido se decidió seleccionar Angular como el lenguaje principal para el desarrollo del proyecto al no tener previamente una mayor diferenciación a parte de su simplicidad de aprendizaje por parte de React.

5.3. Metodología

La metodología utilizada ha sido SCRUM, metodología aprendida en la asignatura de *Gestión de Proyectos*. SCRUM es un marco de trabajo para desarrollo ágil, el cual son un conjunto de buenas prácticas para trabajar de forma colaborativa y obtener el mejor resultado posible.

Algunas de las medidas características de la metodología que se han aplicado en el proceso de desarrollo son:

- La planificación del desarrollo se ha realizado en periodos de una misma duración con objetivos claros, llamados *"sprints"*. Estos se encuentran en el repositorio de GitHub `https://github.com/humbertoms99/DIET_COMPOSITION`.
- El desarrollo de la aplicación ha sido incremental.
- Se han mantenido reuniones al final de cada sprint para hablar de las implementaciones y próximos desarrollos.
- Se ha estimado el valor de los objetivos de los *sprints* mediante Story points.
- Se han añadido, movido y completado tareas mediante la herramienta Zenhub.
- Se ha establecido un tiempo de 2 semanas por cada sprint.

5.4. Desarrollo del proyecto

Durante las primeras semanas se realizó un curso de LaTeX [6], con el objetivo de documentar las elecciones entre herramientas.

Posteriormente se pasó a documentar el problema planteado en el articulo [10] sobre el problema de n-alcanos, siendo el objetivo inicial del proyecto.

Uno de los objetivos más importantes para el funcionamiento del proyecto fue el uso de una biblioteca que resolviera problemas de programación lineal, como [4]. El proceso de instalación y configuración de esta biblioteca conllevó algunos problemas: el primer problema fue la necesidad de ejecutar algunos comandos de la instalación desde una terminal Linux. Al estar trabajando en un entorno Windows se decidió continuar en el sistema operativo, por lo que se optó por usar una herramienta que actuara como subsistema Linux en Windows 10, en este caso Debian [12], al ser este un software gratuito y con distribución libre.

Otro problema que surgió durante la configuración de la librería fue el requerimiento de un compilador en C, ya que la librería está escrita en ANSI C, lo cual se resolvió con siguiendo los pasos de una respuesta encontrada en StackOverFlow [15]. Otro punto a destacar sobre el uso de esta librería, es que requiere que el servidor se este ejecutando en una terminal Linux, por lo que se configuró Visual Studio Code para escribir comandos en una terminal Linux.

Una vez entendidos los objetivos del problema a resolver y disponer de una librería que resolviera estos requerimientos, comenzó la fase de implementación, dividiendo esta parte del desarrollo en periodos muchos más marcados con requisitos claros y trabajando simultáneamente en el proyecto front-end en Angular y el back-end en un proyecto laravel. A continuación analizaremos algunos de los puntos que consideramos claves de implementación.

Formularios

Angular ofrece dos técnicas para crear formularios: formularios basados en plantillas (Template Forms) y formularios reactivos (Reactive Forms).

Se optó por realizar los formularios reactivos, al ser más sencillos, ya que las validaciones se trabajan desde la clase componente siendo una lógica más limpia, al pasar las validaciones al componente. Al contrario que los formularios por plantilla, es más fácil definir formularios dinámicos (punto importante al trabajar con una matriz que va cambiar de tamaño).

La diferencia principal es que los *Reactive Forms* definen el formulario en la clase del componente y trabajan en la vista con las directivas *formGroup* o *formControlName*, al contrario que los *Template Forms*, donde las reglas

y estructura del formulario se hacen en la vista utilizando directivas como *ngForm* y *ngModel*

Para mantener la idea de separación entre vista y lógica, se decidió utilizar los *Reactive Forms*, que además son más fáciles de usar y permiten más adaptaciones por parte de las validaciones.

```
miFormulario: FormGroup = this.fb.group({
  alcanos: [0, [Validators.required, Validators.min(1)]],
  plantas: [0, [Validators.required, Validators.min(1)]],
});
```

Figura 5.1: Fórmulario index Matrices

En la figura 5.1 se puede ver el código de clase del componente que contendrá los índices de la matriz de datos a solicitar posteriormente. Estos campos son siempre requeridos y deben tener un valor superior a "1"; por el contrario, el botón de crear matrices permanecerá deshabilitado.

Sources:	0:	tracers	0: value ="1"
			1: value = "2"
	1:	tracers	0: value = "4"
			1: value = "3"
	2:	tracers	0: value = "8"
			1: value = "4"

Figura 5.2: Estructura de datos del formulario que contiene los valores de las fuentes

En la Figura 5.2 podemos observar la estructura del segundo formulario que contendrá los datos de la matriz de datos con los valores de los marcadores para cada fuente. La estructura de este formulario es un "*array de Sources*"que tendrá tantos elementos como fuentes se especifiquen en los *index* del anterior formulario. Cada elemento está formado por un único

formArray de marcadores (tantos como se especificaron en el primer formulario). En este último array se encuentran los correspondientes valores de cada marcador para la fuente.

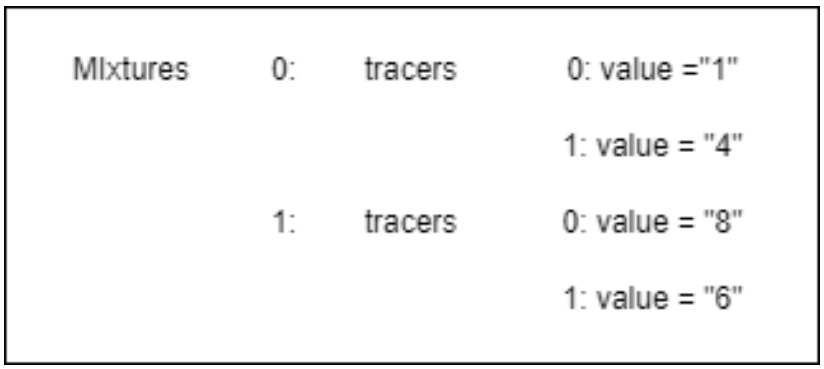


Figura 5.3: Estructura de datos del formulario que contiene los valores de las mezclas

En la Figura 5.3 observamos la estructura de datos del tercer formulario que contiene los datos de las mezclas sobre las cuales queremos calcular el problema. Estos datos son rellenados mediante la interfaz al usar el botón de añadir mezcla. La estructura del formulario es un "*array de Mixtures*"que tendrá tantos elementos como mezclas se crean en la interfaz. Cada elemento esta formado por un único *formArray* de marcadores (tantos como se especificaron en el primer formulario). En este último array se encuentran los correspondientes valores de cada marcador para la fuente.

Por último, se implementó un cuarto formulario que permite el cambio de nombre de las fuentes, con el objetivo de hacer más visible las posibilidades de resolver el problema de los *Mixing Models*.

Estructura documentos csv

En esta sección visualizaremos los archivos .csv exportados por la aplicación web y explicaremos su estructura.

Export Input Data

En la Figura 5.4 podemos ver una vista de los datos que corresponden al archivo *InputData.csv*; este archivo será el esperado en la opción de importar datos.

Las dos primeras filas marcan el número de marcadores y fuentes respectivamente. A continuación, siempre se dejará una línea vacía que marca que

las siguientes líneas son los valores para cada fuente; el número de filas de fuentes corresponderá al número de fuentes fijado en la segunda fila. Para diferenciar las filas referentes a los valores de las mezclas se deja otra fila vacía. Es obligatorio dejar estas filas vacías para que la entrada de los datos sea correcta.

	A	B	C	D
1	Marcadores:	2		
2	Fuentes:	3		
3				
4	Fuente- 1:	1	2	
5	Fuente- 2:	4	3	
6	Fuente- 3:	8	4	
7				
8	Mezcla- 1:	1	4	
9	Mezcla- 2:	9	6	
10				

Figura 5.4: Vista csv InputData

Export Solution

En la figura 5.5 podemos ver la vista de la solución del problema. Se ha intentado que mantuviera una disposición similar a la de la aplicación web. En una primera parte tenemos los datos del problema tal cual se encuentran en el archivo *InputData.csv*. Posteriormente se añade una línea con la proyección de la muestra usada para realizar el calculo.

Por último, podemos apreciar una estructura de tabla en la cual podemos ver las estimaciones máximas y mínimas de las mezclas sobre las fuentes.

Export Paso Intermedio

En el caso de los dos tipos de exports vistos previamente, la descarga era trivial ya que se mostraban sus pertinentes botones de exports una vez realizado el cálculo. En el caso de que queramos ver los cálculos intermedios realizados para cada valor, sera necesario dar click sobre el propio valor de estimación tanto para máximos, que descargará un archivo con nombre

	A	B	C	D	E	F	G
1	Marcadores:	2					
2	Fuentes:	3					
3							
4	Fuente- 1:	1	2				
5	Fuente- 2:	4	3				
6	Fuente- 3:	8	4				
7							
8	Mezcla- 1:	1	4				
9	Mezcla- 2:	9	6				
10							
11	Mezcla -> proyeccion: (1 4) -> (1.7999999999999998 3.5999999999999996)						
12							
13		Mezcla- 0			Mezcla- 1		
14		[Min	,Max]		[Min	,Max]	
15	Fuente- 1:	1.8	1.8		0	1	
16	Fuente- 2:	0	0		0	1.5	
17	Fuente- 3:	0	0		0.375	1	
18							

Figura 5.5: Vista csv Solution

M1_max1.csv, como para mínimos, donde el fichero se llamará *M1_min1.csv*. El nombre es una simplificación de "Mixture 1 - Maximize Source 1".

El contenido de la Figura 5.6 corresponde con los archivos *.mod* usados desde la API para resolver el problema de maximización de un objetivo sobre un conjunto de reglas y ecuaciones. La segunda columna marca el tipo de línea que es, teniendo líneas para la definición de las variables: una única línea que señaliza el tipo de operación (maximización o minimización) y la función objetivo, y unas últimas líneas que contendrán las ecuaciones con los índices de los marcadores tanto para fuentes como para las mezclas

	A	B	C	D
1	var x1 >=0	# Variable definition		
2	var x2 >=0	# Variable definition		
3	var x3 >=0	# Variable definition		
4	minimize z: x1	# Objective function		
5	subject to con1: 1*x1 + 4*x2 + 8*x3 = 1.8	# Constraint 1		
6	subject to con2: 2*x1 + 3*x2 + 4*x3 = 3.6	# Constraint 2		
7	end			
8				

Figura 5.6: Vista csv paso intermedio

5.5. Ciclo de la aplicación

En este apartado vamos a comentar los pasos que realiza la aplicación y nos detendremos en los que consideremos más importantes

Al entrar a la web en la parte superior veremos un *"Top Navigation Bar"* [17], donde la primera ventana ha sido dedicada a la resolución del problema, una segunda con un vídeo guía de uso de la aplicación, el icono con enlace directo al GitHub y finalmente un apartado de las personas participantes en el desarrollo del proyecto. Justo debajo encontraremos el primer formulario con dos inputs para introducir los índices de la matriz de fuentes. junto con dos botones: botón *"Crear"* y *"Seleccionar archivo"*. Este segundo lanza un evento y permite leer archivos con el formato esperado.

Estos dos botones marcan las dos alternativas del flujo en la Figura 5.7. En el caso de la rama izquierda del flujo (*"Introducción manual de los datos"*) es necesario rellenar el primer formulario con los índices, un segundo con los valores de las fuentes y dar valores para las mezclas (pudiendo añadir más mezclas con su botón correspondiente).

Estos dos flujos se unen al dar al botón *"Resolver"*. Al hacer click se calcularán las proyecciones de las mezclas, y en caso de ser diferentes a las mezclas originales se mostrará un mensaje en pantalla con los valores sobre los que se han realizado los cálculos.

Una vez se han calculado las proyecciones de las mezclas, se ejecuta una petición por cada mezcla con los arrays de fuentes y valores para la mezcla al API. Esta petición genera un archivo tipo *"file_max1_1.mod"* en la ruta **storage/app** del proyecto en laravel por cada marcador que tiene cada mezcla. Su contenido es equivalente al mostrado en la figura 5.6. Para resolver el problema de maximización o minimización, hacemos uso de la librería [4] con el comando *"glpsol -m ../storage/app/file_max1_1.mod -o files_sol/file_max1_1.sol"* que nos genera un archivo tipo *".sol"*. Este comando requiere ser ejecutado desde una terminal Linux (En el entorno local se uso debian wsl [12]); este comando genera un archivo tipo *".sol"* en la ruta *public/files_sol/*.

Dicho archivo, denominado *file_max1_1.sol*, devuelve el valor objetivo calculado en la línea 6, como podemos ver en la figura 5.8. El método agrupará los objetivos de todos los archivos generados en arrays de máximos y mínimos por mezcla y se retornan como repuesta JSON (Figura 5.9).

Por último, se muestra en la web los resultados de los cálculos junto con los botones de exportación de los datos de entrada y de la solución.

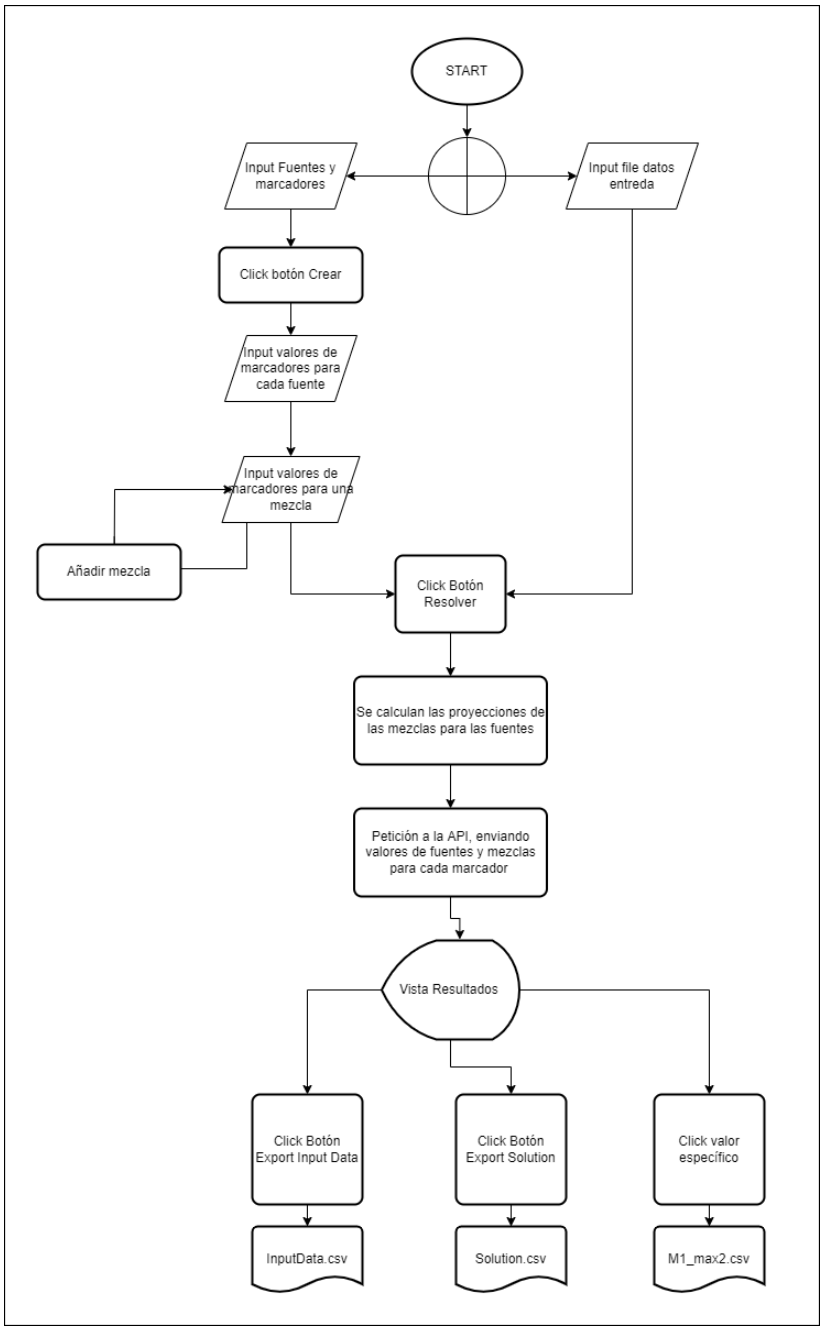
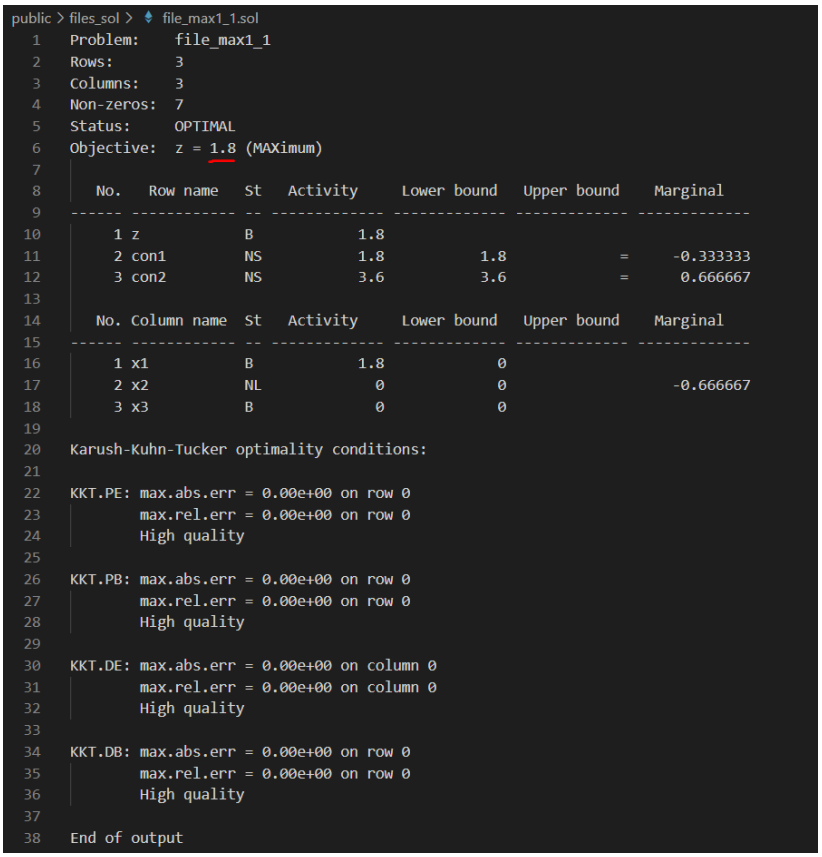
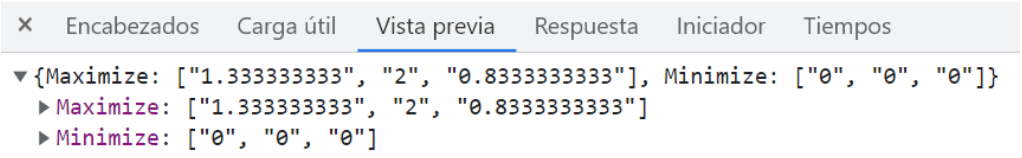


Figura 5.7: Ciclo de vida



```
public > files_sol > file_max1_1.sol
Problem:    file_max1_1
Rows:       3
Columns:    3
Non-zeros:  7
Status:     OPTIMAL
Objective:  z = 1.8 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            1.8
     2 con1         NS           1.8           1.8             =     -0.333333
     3 con2         NS           3.6           3.6             =      0.666667

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B            1.8             0
     2 x2           NL             0             0                  -0.666667
     3 x3           B              0             0

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err = 0.00e+00 on column 0
        max.rel.err = 0.00e+00 on column 0
        High quality

KKT.DB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```

Figura 5.8: Archivo file_max1_1.sol



```
× Encabezados  Carga útil  Vista previa  Respuesta  Iniciador  Tiempos

▼ {Maximize: ["1.333333333", "2", "0.8333333333"], Minimize: ["0", "0", "0"]}
  ▶ Maximize: ["1.333333333", "2", "0.8333333333"]
  ▶ Minimize: ["0", "0", "0"]
```

Figura 5.9: Respuesta JSON formado los arrays máximos y mínimos de una mezcla.

5.6. Problema subida API a Heroku

Inicialmente se desarrolló el proyecto en dos códigos: una API que utilizaba la librería GLPK escrita en C y una parte front que se comunicaba con esta API mediante peticiones. Estos dos proyectos se desarrollaron por completo, funcionando ambos correctamente en un entorno local.

Uno de los requisitos del proyecto es que fuera una herramienta accesible, y que se desplegara como aplicación web. Por este motivo se seleccionaron Heroku para subir el proyecto back-end y Netlify para el proyecto front-end. Los códigos se consiguieron subir sin problemas. Pero en el caso de la API, necesitaba contar con la librería GLPK instalada en el servidor. La librería GLPK usada en el entorno local no se consiguió instalara en un servidor web gratuito.

Por este motivo, se tomó la decisión de cambiar el código front-end ya subido a producción, haciendo que el proyecto front-end en Angular, no necesitara el uso de la API. En este momento se realizó una búsqueda de librerías que pudieran ejecutarse en Typescript y resolvieran los mismos problemas que el paquete GLPK. Las opciones finales fueron: `https://www.npmjs.com/package/glpk-ts` y `https://www.npmjs.com/package/glpk.js`. Finalmente, y ya que la lógica del código está desarrollada en Typescript se instaló y aprendió a utilizar **glpk-ts**. Este paquete es una interfaz de la misma librería GLPK [4] que se usa desde módulos.

Este cambio ha significado no realizar peticiones a la API, lo cual ha derivado en un desarrollo de estas funcionalidades en el proyecto front-end.

5.7. Internacionalización

A través de la internacionalización se consigue realizar un proceso de expansión del software. Se ha diseñado la aplicación permitiendo intercambiar entre diferentes idiomas. Se proporcionan dos idiomas: Español e Inglés. Esto se permite desde una select en el menú superior que está siempre visible (figura 5.10), cambiando este valor se lanza un evento que cambia una variable en el localStorage. Esta variable marca el fichero de traducciones a tomar por la aplicación. Los ficheros de traducciones se encuentran en la ruta **src/assets/i18n** y tienen extensión .json.

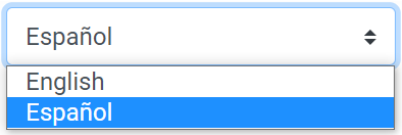


Figura 5.10: Select de Idioma

Para realizar traducciones desde angular se hace uso del módulo Angular Translate, que mediante el pipe **translate** marca que ese texto tiene una traducción. En la ejecución de la aplicación se carga el fichero .json correspondiente a la variable **lang** en el localStorage. La estructura de estos archivos la podemos ver en la figura 5.11: consiste en una cadena de texto (línea de código escrita en el fichero *.html*), el símbolo **":"** y una cadena de la traducción al idioma correspondiente.

Figura 5.11: Archivo en.json

Trabajos relacionados

6.1. Trabajos relacionados

Este proyecto es el primer trabajo de fin de grado que implementa la resolución de problemas lineales para *Mixing models* desde una web. Como objetivo de esta primera implementación se ha realizado el desarrollo una aplicación web híbrida.

6.2. Proyectos similares

MixSIR

MixSIR fue el primer proyecto desarrollado por Jonathan W. Moore y Brice X. Semmens sobre Bayesian Mixing Models, que estimaba las proporciones de las fuentes en una mezcla, como herramienta de calculo para ecología.

En el artículo [1], Andrew LLoyd Jackson, Richard Inger, Stuart Bearhop y Andrew C Parnell. Declaran un comportamiento erróneo de MixSIR. Mostrando ejemplos en los cuales MixSIR no identifica las proporciones dietéticas correctas más del 50 % de las veces.

SIAR

SIAR es un paquete que utiliza cálculos estadísticos bayesianos principalmente orientado en la ecología sobre isótopos estables: calculando las contribuciones de las fuentes a una mezcla. Fue desarrollado por Parnell, A.C., Inger R., Bearhop, S. y Jackson, A.L. 2010 [5].

SIAR trabaja con unos términos de error residual por cada eje isotópico. Si los datos de las mezclas son muy variables respecto a las fuentes, se utiliza este término de error residual

MixSIAR

MixSIAR es un paquete de R desarrollado por Brice Semmens, Brian Stock, Eric Ward, Agrew Parnell, Donald Phillips y Andrew Jackson. Es un proyecto colaborativo que incorpora los avances en Bayesian Mixing Models, combinando MixSIR y SIAR.

En el articulo [11] se describe su principal ventaja respecto a los desarrollos anteriores, permitiendo variar las proporciones de la mezcla y calcular mediante criterios de información el apoyo relativo.

Es importante comentar que ninguno de estos software se pueden utilizar vía web, ni tienen un diseño tan sencillo e intuitivo como el presentado en este proyecto.

Conclusiones y Líneas de trabajo futuras

En este capítulo incluiré mis conclusiones. Además, se citarán algunas posibles mejoras al proyecto o posibles líneas de desarrollo futuras.

7.1. Conclusiones

El tiempo que he dedicado al trabajo de fin de grado me ha permitido obtener muchas habilidades nuevas, especialmente sobre el desarrollo de aplicaciones web, aprendiendo los fundamentos de los principales lenguajes que se utilizan en estas aplicaciones. Se ha conseguido desarrollar una aplicación Single-page Application en Angular; por otra parte, se han utilizado librerías que resuelven problemas lineales utilizando una librería escrita en Typescript. Se aprendió a realizar peticiones desde una API a una web usando textos en JSON y apoyándome en Postman para comprobar el funcionamiento de las peticiones. También querría destacar el uso de estilos, principalmente por medio los Componentes que ofrece Bootstrap. Y también se ha aprendido sobre LaTeX, que es un lenguaje de creación de documentos que se ha utilizado para realizar la documentación de este proyecto. Por último, he aplicado las metodologías ágiles para la planificación de este proceso.

7.2. Posibles mejoras

Algunas mejoras que consideramos interesantes ante posibles desarrollos futuros son:

- **Mejorar la interfaz gráfica:** En la aplicación se han utilizado principalmente Componentes Bootstrap prediseñados, y se han reajustado algunos colores y posiciones, pero creemos que los estilos pueden aportar más calidad a la aplicación. Algunas nuevas mejoras serian hacer la aplicación más adaptativa (Responsive), añadir Spinner en los tiempos de carga, o aplicar un estilo más similar en toda la aplicación.
- **App Android:** Creación de una App para móviles, añadiendo mejoras de diseño de interfaz y experiencia de usuario para los usuarios móviles, y diseño de un logo de la aplicación
- **Permita resolver otros problemas:** Añadir la resolución de otros problemas matemáticos que estén relacionados con problemas de Mixing Models.
- **Análisis de datos:** Añadir análisis de los resultados obtenidos, con sus correspondientes botones de exportación.

Bibliografía

[1] Stuart Bearhop Andrew Lloyd Jackson, Richard Inger and Andrew C Parnell. Erroneous behaviour of mixsir, a recently published bayesian isotope mixing model: A discussion of moore & semmens (2008). *ResearchGate*, 12(3):E1–E5, 2009.

[2] Jonathan W.Moore Brice X. Semmens and Eric J. Ward. Improving bayesian isotope mixing models: a response to jackson et al. (2009). *Ecology Letters*, 12:E6–E8, 2009.

[3] Jana E. compton Elise F. Granek and Donald L. Phillips. Mangrove-exported nutrient incorporation by sessile coral reef invertebrates. *SpringerLin*, 12:462–472, 2009.

[4] Inc. Free Software Foundation. Glpk (gnu linear programming kit). https://www.gnu.org/software/glpk/, 2012. [Intenet; descargado 10-febrero-2022].

[5] Andrew L. Jackson. Siar stable isotope analysis in r - package. https://andrewljackson.github.io/siar/, 2010. [Internet; visto 26-junio-2022].

[6] Michelle Krummel. Latex tutorials (featuring texmaker). https://www.youtube.com/watch?v=0ivLZh9xK1Q&list=PL1D4EAB31D3EBC449&ab_channel=MichelleKrummel, 2021. [Youtube; visto 11-enero-2022].

[7] Philipp Neubauer and Olaf P. Jensen. Bayesian estimation of predator diet composition from fatty acids and stable isotopes. *PeerJ*, 2015.

[8] Lawrence N. Hudson; Tim Newbold;. The predicts database: a global database of how local terrestrial biodiversity responds to human impacts. *ERIC*, 4(24):4701–4735, 2014.

[9] Anthony W. J. Bicknell; Mairi E. Knigth; David T. Bilton; Maria Campbell; James B. Reid; Jason Newton and Stephen C. Votier. Intercolony movement of pre-breeding seabirds over oceanic scales: implications of cryptic age-classes for conservation and metapopulation dynamics. *Wiley Online Library*, 20(2):160–168, 2013.

[10] M. L. Campagnolo P. Barcia, M. N. Bugalho and J. O. Cerdeira. Using n-alkanes to estimate diet composition of herbivores: a novel mathematical approach. *Animal(2007)*, 1:141–149, 2007.

[11] Brian C Stock; Andrew L Jackson; Eric J Ward; Andrew C Parnell; Donald L Phillips; and Brice X Semmens. Analyzing mixing systems using a new generation of bayesian tracer mixing models. *PeerJ*, 2018.

[12] The Debian Project. Debian. https://apps.microsoft.com/store/detail/debian/9MSVKQC78PK6?hl=es-es&gl=ES, 2022. [Microsoft store; descargado 17-febrero-2022].

[13] Robert Rodale. Breaking new ground: The search for a sustainable agriculture. *ERIC*, 17(1):15–20, 1983.

[14] William H. Blake; Katherine J. Ficken; Philip Taylor; Mark A. Russel and Desmond E. Walling. Tracing crop-specific sediment sources in agricultural catchments. *ScienceDirect*, 138-140:322–329, 12.

[15] Stackoverflow. No acceptable c compiler found in $path when installing python. https://stackoverflow.com/questions/19816275/no-acceptable-c-compiler-found-in-path-when-installing-python, 2022. [Internet; descargado 20-febrero-2022].

[16] Gideon Bartov; Amrika Deonarine; Thomas M. Johnson; Laura Ruhl; Avner Vengosh and Heileen Hsu-Kim. Environmental impacts of the tennessee valley authority kingston coal ash spill. 1. source apportionment using mercury stable isotopes. *ACS Publications*, 47(4):2092–2099, 2012.

[17] w3schools. Top navigation bar. https://www.w3schools.com/howto/howto_js_topnav.asp, 2022. [Internet; visto 12-septiembre-2022].