



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Jellyfish Forecast
Documentación Técnica**



Presentado por Pablo Santidrian Tudanca
en Universidad de Burgos — 26 de junio
de 2020

Tutor: José Francisco Díez Pastor y Álgvar
Arnaiz González

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	14
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catálogo de requisitos	17
B.4. Especificación de requisitos	18
Apéndice C Especificación de diseño	23
C.1. Introducción	23
C.2. Diseño de datos	23
C.3. Diseño procedimental	26
C.4. Diseño arquitectónico	27
C.5. Diseño de interfaces	27
Apéndice D Documentación técnica de programación	29
D.1. Introducción	29
D.2. Estructura de directorios	29

D.3. Manual del programador	30
D.4. Compilación, instalación y ejecución del proyecto	31
D.5. Pruebas del sistema	35
Apéndice E Documentación de usuario	39
E.1. Introducción	39
E.2. Requisitos de usuarios	39
E.3. Manual del usuario	39
Bibliografía	43

Índice de figuras

A.1. <i>Burndown chart</i> del Sprint 1	3
A.2. <i>Burndown chart</i> del Sprint 2	4
A.3. <i>Burndown chart</i> del Sprint 3	5
A.4. <i>Burndown chart</i> del Sprint 4	6
A.5. <i>Burndown chart</i> del Sprint 5	7
A.6. <i>Burndown chart</i> del Sprint 6	8
A.7. <i>Burndown chart</i> del Sprint 7	9
A.8. <i>Burndown chart</i> del Sprint 8	10
A.9. <i>Burndown chart</i> del Sprint 9	11
A.10. <i>Burndown chart</i> del Sprint 10	12
A.11. <i>Burndown chart</i> del Sprint 11	13
 B.1. Diagrama de casos de uso	 19
 C.1. Variables del dataset original	 24
C.2. Variables del dataset original	25
C.3. Variables del dataset final	25
C.4. Diagrama de flujo	26
C.5. Representación del patrón Modelo-Vista-Controlador	27
C.6. Bocetos iniciales	28
 D.1. Descargar repositorio	 31
D.2. Nueva app en <i>Heroku</i>	33
D.3. Nombrar nueva app en <i>Heroku</i>	34
D.4. Modos de despliegue en <i>Heroku</i>	34
D.5. Comparativa diferentes modelos	37
D.6. Comparativa entre predicción y realidad	37
 E.1. Página de inicio	 40

E.2. Página de consulta	40
E.3. Error en la consulta	41
E.4. Página de consulta con gráficos	41
E.5. Página de contacto	42

Índice de tablas

A.1. Equivalencias <i>Story Points</i> y tiempo estimado	2
A.2. Costes de personal	15
A.3. Licencias de bibliotecas y herramientas utilizadas	16
B.1. Caso de uso 1: Visualizar predicción	19
B.2. Caso de uso 2: Exportar resultados	20
B.3. Caso de uso 3: Visualización del mapa	20
B.4. Caso de uso 4: Consulta predicción	21
B.5. Caso de uso 4.1: Consulta modelo predictivo	21
B.6. Caso de uso 4.2: Selección de la fecha a consultar	22

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En el siguiente apéndice se realizará una estimación de tiempo, trabajo y dinero que será necesario invertir para la realización de este proyecto.

En una primer apartado se tratará sobre la evolución temporal que ha tenido el proyecto, indicando los principales cambios y objetivos de cada *Sprint*, así como una estimación del tiempo que se estimaba, sería necesario invertir antes de la realización del mismo.

En la segunda parte se centra en la viabilidad del proyecto tanto en el ámbito económico (estimación de costes y beneficios) como en el legal (licencias software, etc.).

A.2. Planificación temporal

En este apartado se procederá a explicar con detalle cuál ha sido el resultado de la planificación del proyecto. Esta planificación se ha realizado utilizando una metodología ágil basada en *sprints* de una duración de una o dos semanas en función de las necesidades y el tiempo disponible debido a otras cargas de trabajo diferentes a este proyecto.

En estos *sprints* se van marcando ciertos objetivos que serán revisados junto a los tutores en las reuniones al final de los mismos. Los objetivos del siguiente *sprint* serán marcados durante dichas reuniones.

Para el control de tiempos se ha utilizado la herramienta ZenHub siendo la valoración de los *Story Points* la siguiente:

Story Points	Estimación temporal
1	1 hora
2	1,5 horas
3	2 horas
4	2,5 horas
5	3 horas
6	3,5 horas
7	4 horas
8	6 horas
9	9 horas

Tabla A.1: Equivalencias *Story Points* y tiempo estimado

Sprint 1 (29/01/2020 - 05/02/2020)

En esta primera reunión se marcó el comienzo del proyecto. Ya se había hablado anteriormente con uno de los tutores (José Francisco) del interés sobre el proyecto propuesto del que también formaba parte de los tutores Álar Arnaiz.

Al ser la primera reunión se habló de las herramientas que se iban a utilizar así como acordar los primeros objetivos de este *sprint*:

- Crear el repositorio.
- Añadir la plantilla de L^AT_EX a la documentación.
- Crear cuenta en la plataforma *Copernicus*.
- Investigar el funcionamiento básico de las librerías a utilizar.
- Leer una serie de papers que me proporcionaron sobre las medusas.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Se estimó unas 10 horas de trabajo de las que finalmente se invirtieron 8 horas quedando sin terminar una *issue*.

Figura A.1: *Burndown chart* del Sprint 1

Sprint 2 (13/02/2020 - 28/02/2020)

En la segunda reunión se comentó la existencia de una API para la descarga de los datos meteorológicos como alternativa a la descarga de una gran cantidad de datos a través del FTP.

Por otro lado, se me proporcionaron apuntes de la asignatura de minería de datos para su lectura y aprendizaje.

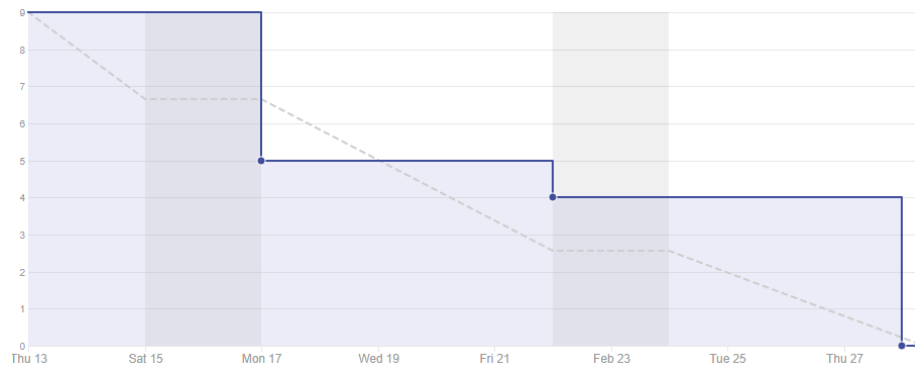
Por último, los tutores me recomendaron iniciar la documentación del plan de proyecto de los *Sprints* que se fuesen sucediendo para no acumular trabajo y se pudiera olvidar detalles del mismos.

Los objetivos fueron los siguientes:

- Realizar script para la descarga de los datos.
- Comenzar a documentar el plan del proyecto.
- Lectura de apuntes y papers.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Se estimaron unas 8 horas de trabajo de las que finalmente se invirtieron 9 horas.

Figura A.2: *Burndown chart* del Sprint 2

Sprint 3 (28/02/2020 - 17/03/2020)

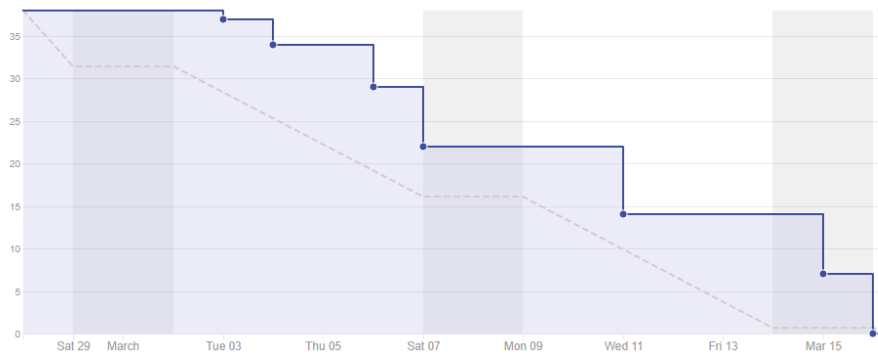
En esta tercera reunión hablamos sobre la descarga de los datos, de las dos opciones posibles nos quedamos con la descarga por FTP por ser más fiable. Además este *Sprint* se centrará en su mayor parte en documentación como las herramientas a utilizar o cuestiones teóricas sobre las medusas aparte de comenzar a desarrollar la base de la aplicación web.

Se marcaron los siguientes objetivos:

- Comienzo desarrollo web.
- Elaboración de parte de la documentación teórica.
- Descarga de los datos en un equipo de cómputo habilitado en la Universidad.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Se estimó unas 19 horas y finalmente se realizaron 24. La causa del desvío de horas principalmente fueron: el comienzo del desarrollo web por el desconocimiento previo y el comentar el código creado para la descarga de los datos necesarios pues se corrigieron errores y se mejoró la salida por pantalla con una barra de descarga más visual.

Figura A.3: *Burndown chart* del Sprint 3

Sprint 4 (17/03/2020 - 30/03/2020)

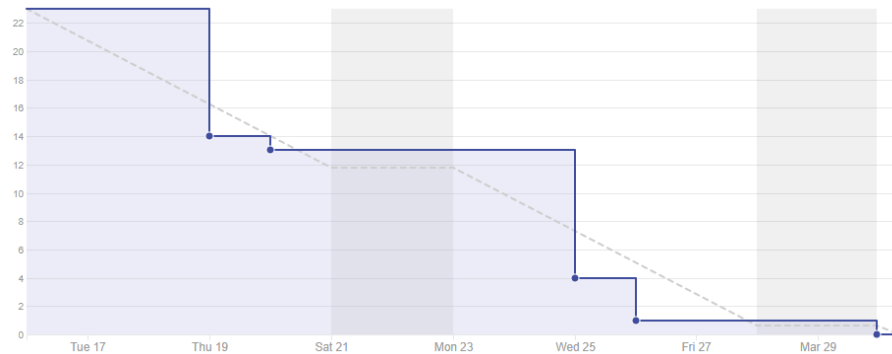
La cuarta reunión se hizo mediante *Skype* con José Francisco debido a la cuarentena por el coronavirus. Se habló sobre la necesidad de utilizar la VPN de la universidad por este mismo motivo para poder tener acceso a la maquina remota.

Por otra parte, se mostró el avance de la web acordando que el siguiente paso debería ser la introducción de los mapas para lo que se comentaron varias bibliotecas de las que se podía hacer uso.

Los objetivos que se marcaron fueron:

- Continuación desarrollo web.
- Introducción de mapas en la aplicación web.
- Conexión a la VPN de la universidad para conseguir dejar los datos descargándose aunque la sesión esté cerrada.
- Continuación de la documentación.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.4: *Burndown chart* del Sprint 4

Se estimaron 15 hora y media y finalmente se realizaron 16.

Sprint 5 (30/03/2020 - 14/4/2020)

Esta reunión se realizó también de manera remota por *Skype* con ambos tutores. Se mostró la implementación de los mapas en la aplicación web así como varias mejoras en la interfaz de la misma. También avances realizados en la memoria y ciertas mejoras propuestas por los tutores.

Sobre la web, a pesar de haber empezado su desarrollo, se recomendó el realizar unos bocetos de la misma para definir la estructura a seguir.

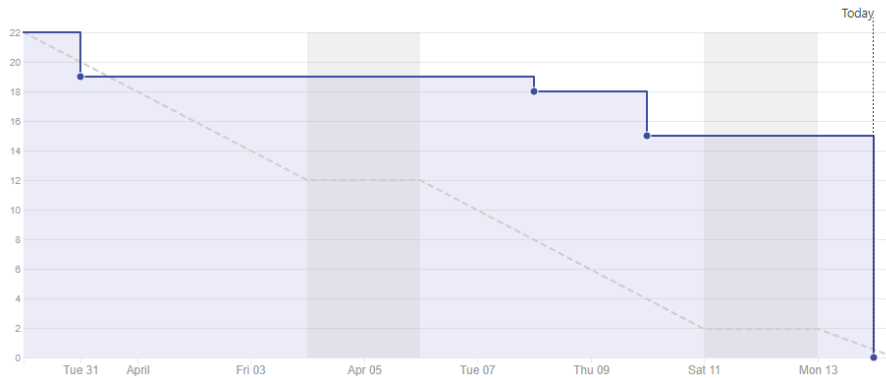
Por otro lado, una vez descargados los datos oceánicos, el siguiente paso será generar una estructura de datos para poder entrenar al modelo.

Por último, se acordó que el siguiente paso en la realización de la memoria debía ser la finalización de los objetivos del proyecto y definir los requisitos.

Los objetivos que se marcaron fueron los siguientes:

- Generar estructura de datos.
- Definir objetivos del proyecto.
- Definir requisitos.
- Definir aspectos relevantes.
- Realizar bocetos aplicación web.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.5: *Burndown chart* del Sprint 5

Se estimaron 12,5 horas y finalmente se realizaron 18,5.

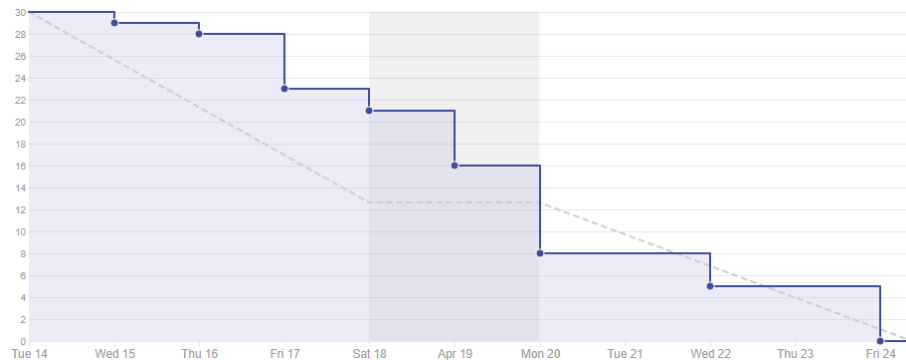
Sprint 6 (15/4/2020 - 24/4/2020)

En esta reunión se mostraron los avances realizados en la generación de la estructura de datos que posteriormente se utilizará en la creación del modelo de predicción. Se comentaron diferentes problemas encontrados como la existencia de valores nulos en los datos de origen así como la mejor forma de localizar los cuadrantes contiguos a las playas. También se realizaron cambios en la memoria y anexos.

Para este sprint se marcaron como objetivo:

- Añadir más cuadrantes a la estructura de datos mencionada así como asegurar la selección de los cuadrantes contiguos a las playa.
- Redacción de casos de uso.
- Investigar el despliegue de la página web.
- Añadir diferentes partes documentación.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.6: *Burndown chart* del Sprint 6

En un principio se estimaron 16 horas y media y finalmente se realizaron 21 horas.

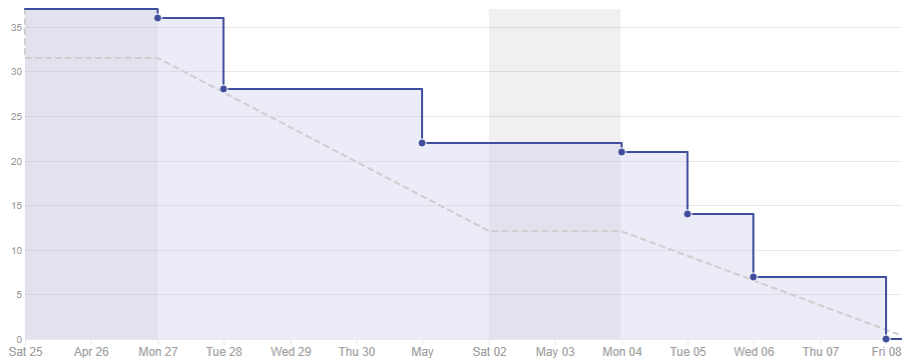
Sprint 7 (25/4/2020 - 8/5/2020)

La temática de la reunión de este sprint fue similar a la anterior. Se habían solucionado los problemas con los valores nulos y añadido más cuadrantes a la estructura de datos, esto dio lugar a algunos problemas por la orientación de las playas. Por este motivo se revisarán las playas para ver si es posible reducir el número de las mismas y así hacer una predicción más precisa. En relación a esto, se empezará con la realización del modelo predictivo aprendiendo la utilización de la biblioteca *Scikit-Learn*. Por último, se mostró el auto despliegue de la aplicación web con *heroku* desde un repositorio a parte del original.

Para este sprint se marcaron como principales objetivos:

- Mejoras en la interfaz de la aplicación web.
- Revisar los datos obtenidos de las playas para reducir el dataset inicial si fuese necesario.
- Aprender la utilización de *Scikit-learn*.
- Corrección y ampliación de anexos y memoria.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.7: *Burndown chart* del Sprint 7

En un principio se estimaron 20 horas y finalmente se realizaron 27.

Sprint 8 (12/5/2020 - 2/6/2020)

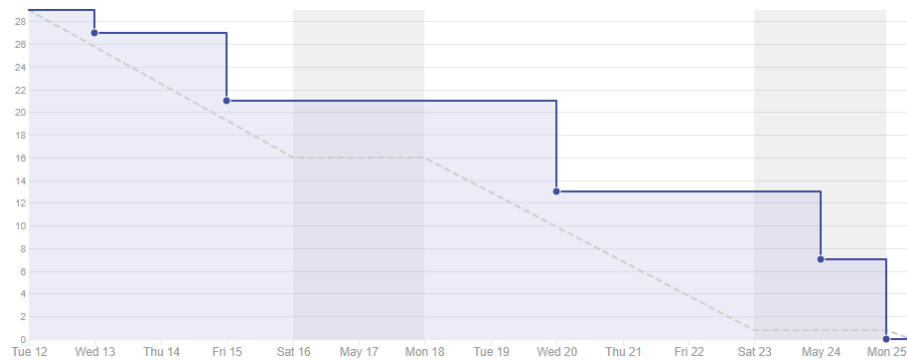
Esta reunión se basó principalmente en la realización del modelo. Se comentaron algunas correcciones en la estructura de los datos (valores nulos, desfase temporal en la organización de los avistamientos).

Por otra parte, hablamos de diferentes algoritmos de minería de datos que se podría utilizar para realizar pruebas y con los que se obtienen mejores resultados así como otras transformaciones que realizar a la estructura de datos para lograr mejores resultado como la normalización de los datos o dar más importancia a unas clases que a otras.

Para este sprint se marcaron como principales objetivos:

- Arreglos en la estructura de datos.
- Investigar los diferentes algoritmos.
- Probar algoritmos de minería de datos.
- Documentar las pruebas y teoría de minería de datos.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.8: *Burndown chart* del Sprint 8

En un principio se estimaron 12 horas y finalmente se realizaron 20.

Sprint 9 (3/6/2020 - 10/6/2020)

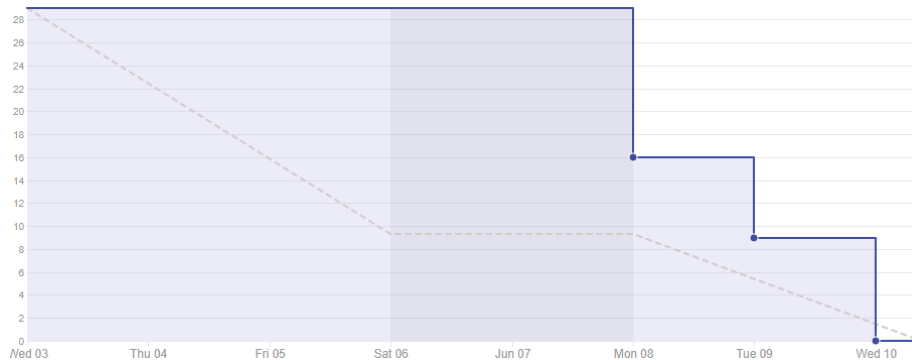
La reunión estuvo basada en los algoritmos de predicción a utilizar. Se mostraron los diferentes algoritmos probados con algunos resultados y se propuso por parte de los tutores la prueba de alguno más.

Aparte de esto, hablamos sobre la posibilidad de utilizar series temporales debido a la naturaleza de los datos. También se obtuvo un nuevo conjunto de datos con un mayor número de lecturas de las que se tenía hasta el momento.

Para este sprint se marcaron como principales objetivos:

- Seguir realizando pruebas con diferentes algoritmos de minería.
- Investigar sobre las series temporales.
- Tratar nuevo Excel.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.9: *Burndown chart* del Sprint 9

En un principio se estimaron 14 horas y finalmente se realizaron 22 hora y media.

Sprint 10 (10/6/2020 - 17/6/2020)

La temática de la reunión fue la misma que las anteriores. Las pruebas realizadas aplicando las series temporales a los modelos no aportaron buenos resultados por lo que se decidió realizar estas con validación cruzada y ver que algoritmo arroja mejores resultados.

En cuanto a la página web, se mostró la introducción de algunas funcionalidades viendo que existían bugs que subsanar.

Para este sprint se marcaron como principales objetivo:

- Pruebas de algoritmos con validación cruzada.
- Arreglar fallos e introducir gráfico en la página web.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.10: *Burndown chart* del Sprint 10

En un principio se estimaron 18 horas y finalmente se realizaron 41.

Sprint 11 (17/6/2020 - 24/6/2020)

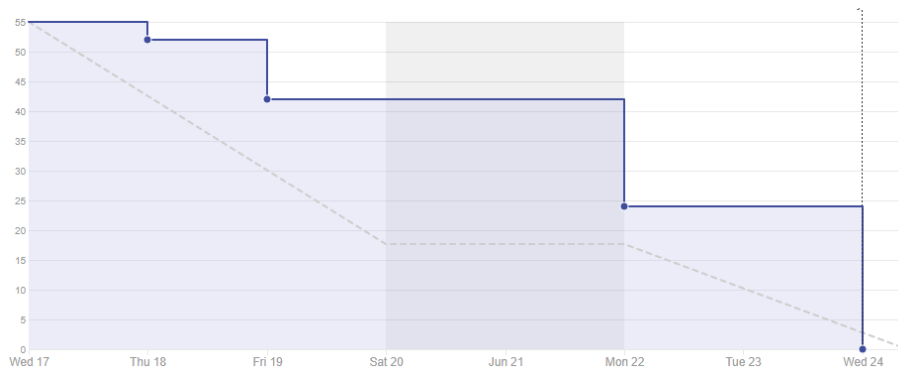
En primer lugar, se trató la realización de las ultimas pruebas realizadas con los modelos. Se obtuvo un resultado aceptable por lo que será el modelo a introducir finalmente en la pagina web.

Por otra parte, se mostraron los avances en la aplicación web que estaba casi terminada por completo. Se propuso por parte de los tutores la introducción de un nuevo gráfico mostrando un historial de los avistamientos en la playa seleccionada. En cuanto a la forma de introducir el modelo, se barajaron varias formas. Finalmente se subirán una serie de los datos de origen con los que pueda trabajar el modelo.

Para este sprint se marcaron como principales objetivo:

- Finalizar la aplicación web.
- Dejar finalizada la documentación a falta de la revisión final por parte de los tutores.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

Figura A.11: *Burndown chart* del Sprint 11

En un principio se estimaron 34 horas y finalmente se realizaron 43.

Sprint 12 (17/6/2020 - 24/6/2020)

Este *Sprint* se centró principalmente en la realización de la documentación. Se finalizó casi por completo y fue revisada por los tutores. A parte se realizaron unas ultimas mejoras en la pagina web a falta de introducir el modelo final.

Para este sprint se marcaron como principales objetivo:

- Finalizar las correcciones de la documentación.
- Introducir el modelo final en la aplicación web.
- Realizar los vídeos de las entregas.
- Mejorar lo *Readme* de ambos repositorios.
- Generar *release*.

Las *issues* para este *Sprint* se pueden ver [aquí](#).

En un principio se estimaron XX horas y finalmente se realizaron XX.

Resumen

A.3. Estudio de viabilidad

Viabilidad económica

Costes

Coste software

Todas las herramientas software utilizadas son de uso gratuito por lo que el coste de este apartado es cero.

Coste hardware

El coste de dispositivos es bajo. Se ha utilizado un ordenador personal valorado en 800 euros. Suponiendo una amortización de 5 años, el coste amortizado seria:

$$\frac{800}{12 * 5} * 5 = 66,67\text{€} \quad (\text{A.1})$$

También se utilizó la máquina de cómputo de la Universidad que tuvo un coste inicial de 4350 euros con una amortización de 6 años.

$$\frac{4350}{12 * 6} * 5 = 322,91\text{€} \quad (\text{A.2})$$

El coste total del hardware será de 389,58 euros.

Coste de personal

Se considera que el proyecto se ha realizado por un desarrollador trabajando 6 horas diarias durante 5 meses:

Concepto	Coste
Salario bruto del trabajador	1200 €
Contingencias comunes (23,6 %)	283,2 €
Desempleo (5,5 %)	66 €
Fogasa (0,2 %)	2,4 €
Formación profesional (0,6 %)	7,2 €
Coste total mensual	1558,80 €

Tabla A.2: Costes de personal

El coste del empleado sería de 1558,80 euros mensuales. Si aplicamos esto por los 5 meses de trabajo el coste total del personal ascendería a 7794€.

Beneficios

El proyecto está realizado para ser disfrutado de manera gratuita y sin publicidad, por lo que a corto plazo no se obtendrían beneficios. Se podría plantear la posibilidad de agregar algunas funcionalidades extras a futuro, siendo estas de pago a través de algún tipo de suscripción mensual.

Viabilidad legal

Para la realización del proyecto se han utilizado multitud de biblioteca de Python de dominio público. A continuación se expondrán las principales herramientas utilizadas.

Tabla A.3: Licencias de bibliotecas y herramientas utilizadas

Librería	Versión	Descripción	Licencia
VsCode	1.46.1	Editor de código.	MIT
Jupyter Notebook	6.0.3	Aplicación para el desarrollo de código en múltiples lenguajes.	BSD
Scikit-Learn	0.22.1	Biblioteca para aprendizaje automático en Python.	BSD
Flask	1.1.1	Biblioteca para crear aplicaciones web en Python.	BSD
Jinja2	2.11.1	Motor para integrar Python con documentos HTML.	BSD
Folium	0.11.0	Biblioteca para generar mapas en Python	MIT
tmux	3.1b	Multiplexador de terminales.	ISC
Bootstrap	3.3.7.1	Biblioteca para desarrollar aplicaciones web responsive.	MIT
JQuery	3.2.1	Biblioteca para optimizar JavaScript.	MIT
Morris.js	0.5.1	Biblioteca para crear gráficos en Javascript y HTML.	MIT

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se recogerán los diferentes objetivos del proyecto con sus correspondientes requisitos funcionales y no funcionales que marcan el desarrollo de este proyecto.

B.2. Objetivos generales

Este proyecto tiene como objetivo el desarrollo de un modelo que nos permita predecir la presencia de medusas en las costas de Chile en función de las condiciones marítimas.

El modelo resultante se utilizará en un aplicación web con la que poder consultar la predicción de la fecha requerida ayudando a su visualización mediante representaciones gráficas.

B.3. Catálogo de requisitos

Requisitos funcionales

RF-1 Obtención de los datos: Se debe ser capaz de descargar los datos necesarios de manera automática a través de un servidor FTP.

RF-2 Filtrado de los datos: Los datos descargados han de ser tratados, descartando las zonas geográficas distintas al lugar de estudio así como las variables ambientales que no sean de utilidad.

RF-3 Cruce de datos: Los datos filtrados se han de cruzar con los de avistamientos, obteniendo una estructura de los avistamientos con su respectiva fecha, localización y variables oceánicas.

RF-4 Consultar modelo: La aplicación debe ser capaz de realizar una consulta al modelo predictivo a partir de los datos introducidos por el usuario.

RF-5 Mostrar mapa: La aplicación debe ser capaz de mostrar un mapa con el que poder interactuar.

RF-6 Introduccion de datos para su visualización: El usuario debe poder seleccionar una serie de datos con los que realizar las consultas.

RF-6.1 Elección de fechas: Se debe poder seleccionar una fecha de la que obtener información.

RF-6.2 Elección de playa: Se debe poder elegir una playa de la que obtener información.

RF-6.3 Visualización de resultados: El usuario debe ser capaz de visualizar los resultados de una playa en la fecha especificada.

RF-7 Exportación de resultados: Se deben poder descargar los resultados de la consulta.

Requisitos no funcionales

RNF-1 Rendimiento: La aplicación debe tener buenos tiempos de respuesta.

RNF-2 Usabilidad: La aplicación debe ser intuitiva, de manera que al usuario no le suponga un esfuerzo su uso.

RNF-3 Diseño *responsive*: Se debe garantizar una correcta visualización en diferentes dispositivos de distintas dimensiones.

RNF-4 Internacionalización La aplicación debe disponer de varios idiomas.

B.4. Especificación de requisitos

Actores

Solo existe un tipo de actor, aquel usuario que consulta las predicciones.

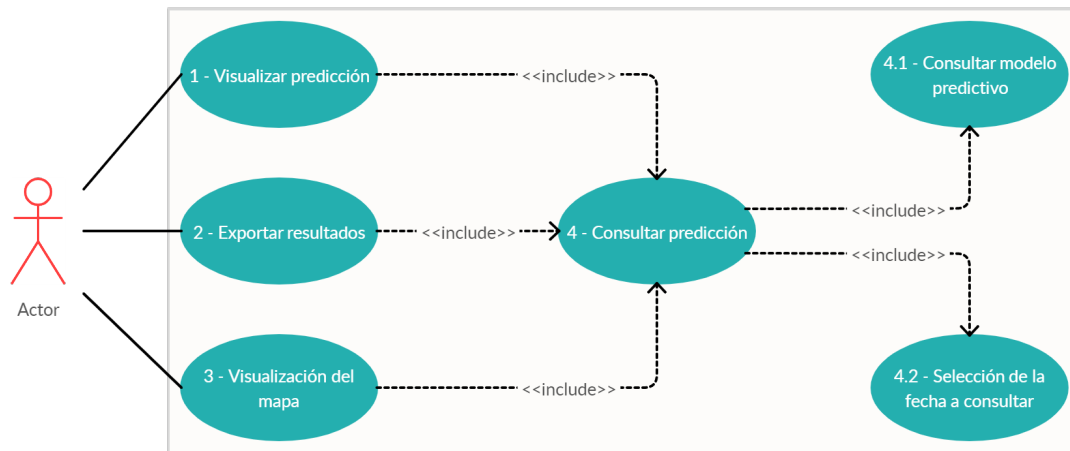


Figura B.1: Diagrama de casos de uso

Diagrama de casos de uso

Casos de uso

CU-1: Visualizar predicción		
Descripción	Permite al usuario visualizar toda la información relativa a la consulta realizada.	
Pre-condiciones	La fecha introducida, es una fecha válida.	
Requisitos	El nombre de la playa introducida, es un nombre válido.	
	RF-5, RF-6	
Secuencia normal	Paso	Acción
	1	El usuario debe acceder a la pestaña «Mapas».
	2	El usuario introduce una fecha.
	3	El usuario podría elegir una playa en particular si lo desea o hacer una búsqueda general.
	4	El usuario pulsa el botón de búsqueda.
	5	Se muestra por pantalla el resultado de la consulta.
Excepciones	1	Fecha introducida no válida.
Frecuencia	Alta	
Importancia	Alta	

Tabla B.1: Caso de uso 1: Visualizar predicción

CU-2: Exportar resultados		
Descripción	Permite al usuario descargar el resultado de la consulta. La fecha introducida, es una fecha válida.	
Pre-condiciones	El nombre de la playa introducida, es un nombre válido.	
Requisitos	RF-6, RF-7	
	Paso	Acción
Secuencia normal	1	El usuario debe acceder a la pestaña «Mapas».
	2	El usuario introduce una fecha.
	3	El usuario podría elegir una playa en particular si lo desea o hacer una búsqueda general.
	4	El usuario pulsa el botón de búsqueda.
	5	Se muestra por pantalla el resultado de la consulta.
	6	El usuario pulsa el botón de descarga.
	7	El archivo se descarga en el dispositivo.
Excepciones	1	Fecha introducida no válida.
Frecuencia	Baja	
Importancia	Alta	

Tabla B.2: Caso de uso 2: Exportar resultados

CU-3: Visualización del mapa		
Descripción	Permite al usuario visualizar un mapa sobre el que se superpondrán los datos	
Pre-condiciones	-	
Requisitos	RF-5	
	Paso	Acción
Secuencia normal	1	El usuario debe acceder a la pestaña «Mapas».
	2	Se muestra la ventana de consultas en la que aparece el mapa.
Excepciones	-	
Frecuencia	Alta	
Importancia	Alta	

Tabla B.3: Caso de uso 3: Visualización del mapa

CU-4: Consultar predicción		
Descripción	Permite al usuario realizar una consulta.	
	La fecha introducida, es una fecha válida.	
Pre-condiciones	El nombre de la playa introducida, es un nombre válido.	
Requisitos	RF-4	
	Paso	Acción
Secuencia normal	1	El usuario debe acceder a la pestaña «Mapas».
	2	El usuario introduce una fecha.
	3	El usuario podría elegir una playa en particular si lo desea o hacer una búsqueda general.
	4	El usuario pulsa el botón de búsqueda.
Excepciones	1	Fecha introducida no válida.
	2	Las coordenadas no existen o no son válidas.
Frecuencia	Alta	
Importancia	Alta	

Tabla B.4: Caso de uso 4: Consulta predicción

CU-4.1: Consultar modelo predictivo		
Descripción	La aplicación web realiza una consulta al modelo predictivo con los parámetros marcados por el usuario.	
Pre-condiciones	-	
Requisitos	RF-4	
	Paso	Acción
Secuencia normal	1	La aplicación web realiza una llamada al modelo predictivo.
	2	El modelo devuelve el resultado de la consulta.
	3	La aplicación web imprime por pantalla dichos resultados.
Excepciones	1	Fecha introducida no válida.
	2	Las coordenadas no existen o no son válidas.
Frecuencia	Alta	
Importancia	Alta	

Tabla B.5: Caso de uso 4.1: Consulta modelo predictivo

CU-4.2: Selección de la fecha la consultar		
Descripción	El usuario elige una fecha en la que realizar a consulta.	
Pre-condiciones	-	
Requisitos	RF-4	
	Paso	Acción
Secuencia normal	1	El usuario debe acceder a la pestaña "Mapas".
	2	El usuario introduce una fecha.
Excepciones	1	Fecha introducida no válida.
Frecuencia	Alta	
Importancia	Alta	

Tabla B.6: Caso de uso 4.2: Selección de la fecha a consultar

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice se tratará la manera en la que se han implementado los datos dentro de la aplicación así como los principales procedimientos utilizados y el diseño estructural de la misma.

C.2. Diseño de datos

Para hablar de los datos utilizados en este proyecto, es necesario referirnos a conjuntos de datos. Para ello, se definirán la estructura de mismos y las transformaciones realizadas antes de trabajar con ellos.

Estructura del conjunto de datos

Los datos oceánicos fueron obtenidos de un servidor FTP de la organización *Copernicus* como se explica en el apartado 5.1 de la memoria. Estos tienen un formato poco habitual (.nc), un formato de archivo para almacenar datos científicos multidimensionales. En el apartado 4.4 de la memoria se da una explicación más detallada.

En este caso, nuestro dataset consta de 4 dimensiones o variables principales: Latitud, Longitud, Profundidad y Fecha. En la intersección de estas cuatro dimensiones, nos encontraremos una serie de variables que podemos observar en la figura [C.2](#). Estas variables corresponden a:

- **mlofst**: Profundidad de la capa mixta del océano.
- **zos**: Altura de la superficie del mar.
- **bottomT**: Temperatura potencial del fondo marino.
- **sithick**: Espesor de hielo marino.
- **siconc**: Concentración de hielo.
- **usos**: Velocidad del hielo marino hacia el este.
- **vsi**: Velocidad del hielo marino hacia el norte.
- **thetao**: Temperatura.
- **so**: Salinidad.
- **uo**: Velocidad de la corriente hacia el este.
- **vo**: Velocidad de la corriente hacia el norte.

xarray.Dataset

► Dimensions: (depth: 50, latitude: 2041, longitude: 4320, time: 1)

▼ Coordinates:

longitude	(longitude)	float32	-180.0 -179.91667 ... 179.91...		
latitude	(latitude)	float32	-80.0 -79.916664 ... 89.9166...		
depth	(depth)	float32	0.494025 1.541375 ... 5727....		
time	(time)	datetime64[ns]	2018-10-14T12:00:00		

▼ Data variables:

mlofst	(time, latitude, longitude)	float32	...		
zos	(time, latitude, longitude)	float32	...		
bottomT	(time, latitude, longitude)	float32	...		
sithick	(time, latitude, longitude)	float32	...		
siconc	(time, latitude, longitude)	float32	...		
usi	(time, latitude, longitude)	float32	...		
vsi	(time, latitude, longitude)	float32	...		
thetao	(time, depth, latitude, longitude)	float32	...		
so	(time, depth, latitude, longitude)	float32	...		
uo	(time, depth, latitude, longitude)	float32	...		
vo	(time, depth, latitude, longitude)	float32	...		

► Attributes: (24)

Figura C.1: Variables del dataset original

Por otra parte, los datos de avistamientos de medusas fueron proporcionados por los tutores. Estos se encontraban en formato *.xlsx*, y contenían los avistamientos de medusas por fecha y localización. En la figura C.2 se puede ver un pequeño ejemplo.

ID	Region	Lugar	Lat.Grac	Lat.Min	Lat.Seg	Long.Gra	Long.Mi	Long.Seg	Lat.dec	Long.de	Date	Year	Origen	Abundanc	Tipo.Abun	Moment	Observaci
3	Arica	Cajeta Quane	18	29	36	70	19	34	18.453	70.326	10/05/2014	2014	Cap Puerto	15	numero	Varadas	Colonias
3	Atacama	Los Machos	27	7	19.19	70	51	27.38	27.122	70.858	10/05/2014	2014	Cap Puerto	23	numero	Varadas	Colonias
5	Antofagasta	Mejillones	23	6	0	70	27	0	23.100	70.450	15/05/2014	2014	Cap Puerto	2	numero	Varadas	Colonias
5	Atacama	Bahia Salada	27	39	55.48	70	56	41.24	27.665	70.945	15/05/2014	2014	Cap Puerto	2	numero	Varadas	Colonias
6	Los Lagos	Bahia Ancond	41	52	26.48	73	49	50.61	41.914	73.831	04/06/2014	2014	Cap Puerto	1	numero	Varadas	Colonias
6	Antofagasta	Tocopilla	22	5	0	70	12	0	22.083	70.200	04/06/2014	2014	Cap Puerto	1	numero	Varadas	Colonias
7	Bio Bio	Cofura	37	6	24.64	73	9	48.02	37.107	73.163	07/06/2014	2014	Cap Puerto	1	numero	Varadas	Colonias
7	Bio Bio	Huentelelen	37	38	15.41	73	40	2.35	37.638	73.667	07/06/2014	2014	Cap Puerto	1	numero	Varadas	Colonias

Figura C.2: Variables del dataset original

Preprocesamiento del conjunto de datos

El dataset inicial contiene información innecesaria para nuestro proyecto. Los datos descargados eran de todo el mundo, por lo que se realizó un filtrado de los datos quedando unicamente las coordenadas cercanas a la zona de estudio. Entre las variables del dataset, también existían algunas que no eran de utilidad. Del mismo modo que con las coordenadas, se desecharon estas variables quedando el dataset de la figura C.2.

xarray.Dataset

► Dimensions: (depth: 3, latitude: 601, longitude: 721, time: 1)

▼ Coordinates:

longitude	(longitude)	float32	-120.0 -119.916664 ... -60.0	
latitude	(latitude)	float32	-60.0 -59.916668 ... -10.0	
depth	(depth)	float32	0.494025 5.078224 9.572997	
time	(time)	datetime64[ns]	2014-05-15T12:00:00	

▼ Data variables:

mlofst	(time, latitude, longitude)	float32	...	
zos	(time, latitude, longitude)	float32	...	
bottomT	(time, latitude, longitude)	float32	...	
thetao	(time, depth, latitude, longitude)	float32	...	
so	(time, depth, latitude, longitude)	float32	...	
uo	(time, depth, latitude, longitude)	float32	...	
vo	(time, depth, latitude, longitude)	float32	...	

► Attributes: (24)

Figura C.3: Variables del dataset final

En el conjunto de avistamientos se eliminaron las columnas innecesarias quedando unicamente las coordenadas, la fecha y el numero de avistamientos.

Una vez pre-procesados ambos conjuntos de datos, se unen para trabajar con un único dataframe. La forma en la que se preparó está desarrollada en los aspectos relevantes del proyecto (punto 5.3 de la memoria).

C.3. Diseño procedimental

A continuación se muestra un diagrama de flujo del uso de la aplicación a la hora de consultar una predicción.

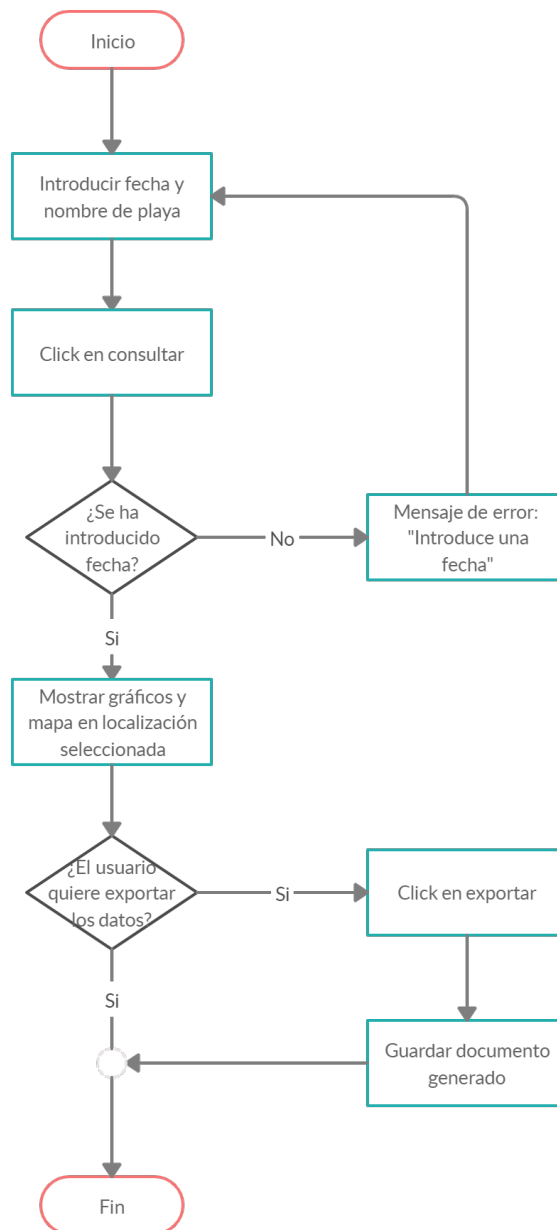


Figura C.4: Diagrama de flujo

C.4. Diseño arquitectónico

La arquitectura de la aplicación se puede considerar como un patrón Modelo-Vista-Controlador (MVC) [1] en lo que nos encontramos tres elementos principales:

- **Modelo:** Contiene una representación de los datos que utiliza el sistema y la lógica de negocio del mismo. Los datos suelen consultarse a una base de datos, en nuestro caso, los datos de consulta se encuentran en una serie de *dataframes*.
- **Vista:** Se trata de la interfaz de usuario. Todo lo relacionado con la representación gráfica es responsabilidad de la vista.
- **Controlador:** Como su propio nombre indica es el encargado de controlar los eventos generados por el usuario. Es el encargado de solicitar los datos al modelo y enviárselos a la vista.

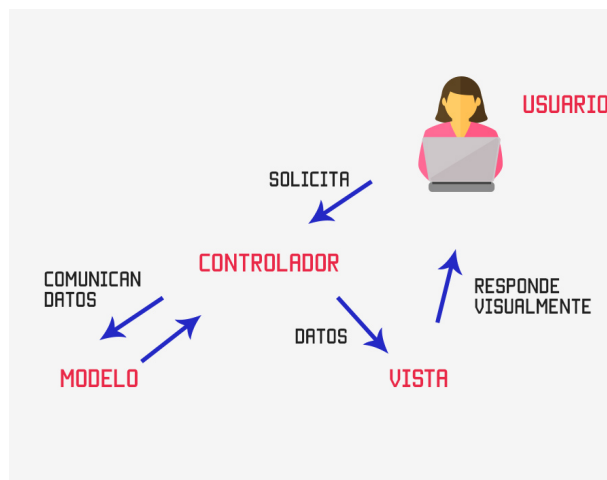


Figura C.5: Representación del patrón Modelo-Vista-Controlador

C.5. Diseño de interfaces

Anteriormente a la realización de la aplicación, se realizaron una serie de bocetos en los que se reflejaron las principales ideas. Para esto se utilizó la aplicación Pencil.

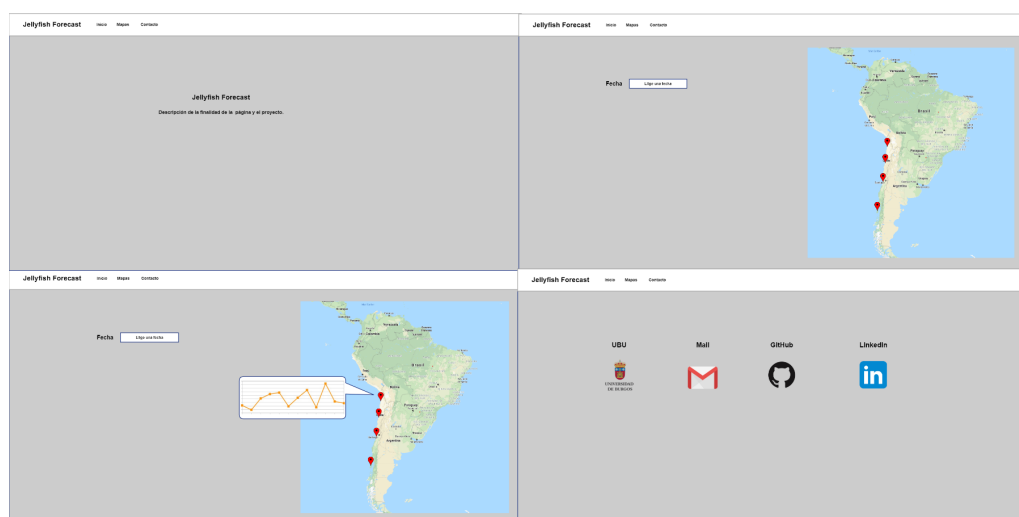


Figura C.6: Bocetos iniciales

Apéndice *D*

Documentación técnica de programación

D.1. Introducción

En este apéndice se explicarán todos los aspectos relevantes del proyecto con el objetivo de facilitar la comprensión del mismo si otro desarrollador quisiera continuar con el trabajo o entenderlo más a fondo.

D.2. Estructura de directorios

El proyecto está alojado en dos repositorios diferentes. El motivo, es el de conseguir un despliegue automático a través de la plataforma *heroku*. Por lo que nos encontramos con:

Web Jellyfish Forecast

Es el repositorio¹ en el que se alojan todos los ficheros necesarios para el funcionamiento de la aplicación web.

Contiene los siguientes directorios:

- **Documentos:** Se guardan, el dataframe en el que estan la relación de playas con sus coordenadas así como un fichero excel con los avistamientos, necesario para realizar el historial de avistamientos.

¹Web Jellyfish Forecast. <https://github.com/psnti/WebJellyfishForecast>

- **static:** Aloja los ficheros css y javascript así como las imágenes necesarios para el buen funcionamiento de la aplicación.
- **templates:** Contiene los ficheros html.

Tambien contiene los siguientes archivos:

- **Procfile:** Archivo necesario para el despliegue en *Heroku* que contiene el comando que debe ejecutar la app en el arranque.
- **app.py:** Archivo python con la funcionalidad de la aplicación.
- **requirements.txt:** Listado de bibliotecas que se deben instalar en la máquina antes de ejecutar la aplicación.

Jellyfish Forecast

Es el repositorio² en el que se alojan el resto de archivos del proyecto.

Contiene los siguientes directorios:

- **Web:** Contiene el boceto de la aplicación realizado antes del desarrollo de la misma así como una referencia al anterior repositorio.
- **docs/latex:** Aloja la documentación del proyecto.
- **src:** Contiene todo el código utilizado para la descarga y tratamiento de los datos así como el entrenamiento del modelo.

D.3. Manual del programador

Este apartado está destinado a la explicación de la instalación de todo lo necesario para poder ejecutar el proyecto o programadores que quieran trabajar en la mejora del mismo.

Para la ejecución del proyecto es indispensable la instalación de Python (el desarrollo se ha realizado con la versión 3.7.6). Posteriormente se instalarán los paquetes recogidos en el fichero requirements.txt como se indica en el apartado D.4.

²Jellyfish Forecast. https://github.com/psnti/Jellyfish_Forecast

En cuanto a la generación del modelo la mayoría del código está elaborado con *jupyter notebook* por lo que es necesaria su instalación. La manera más sencilla es instalando *Anaconda*³.

D.4. Compilación, instalación y ejecución del proyecto

En este apartado se indicará el proceso de obtener proyecto desde el repositorio hasta su despliegue en la plataforma de *Heroku*.

Descargar el proyecto

1. En primer lugar se ha de acceder al repositorio⁴.
2. Descargar el contenido desde **Clone or Download**

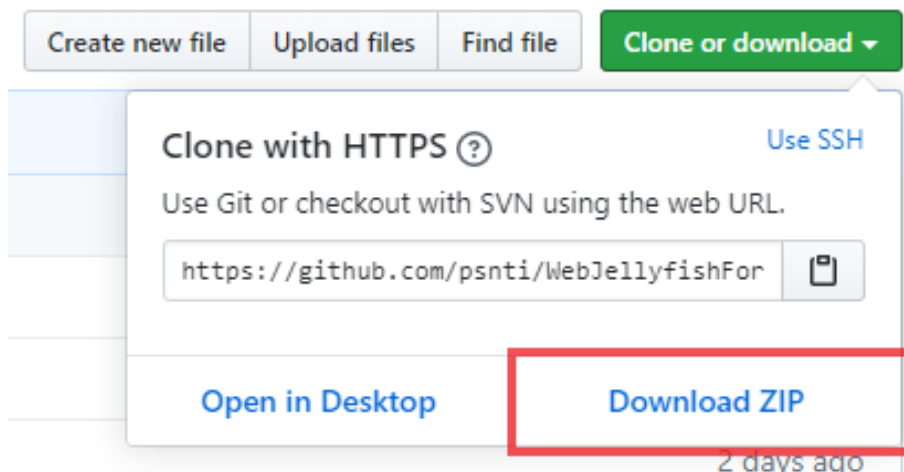


Figura D.1: Descargar repositorio

3. Descomprimir el fichero .zip en la ruta deseada.
4. Antes de instalar las bibliotecas utilizadas en el proyecto crearemos un entorno virtual y accederemos a el:

```
Python -m venv env
env\Scripts\active.bat
```

³Anaconda. <https://www.anaconda.com/products/individual>

⁴Web Jellyfish Forecast. <https://github.com/psnti/WebJellyfishForecast>

5. Para la instalación de la biblioteca necesarias, contamos con el archivo `requirements.txt`. Ejecutando el siguiente comando, se instalarán todas las dependencias del proyecto en nuestro entorno virtual.

```
pip install -r requirements.txt
```

Ejecutar aplicación

1. Una vez instaladas todas las dependencias, podremos ejecutar la aplicación. Desde el entorno virtual ejecutamos:

```
Flask run
```

2. Accederemos desde el navegador a la dirección <http://127.0.0.1:5000>

Modificación de los datos

Si se quisiera modificar el listado de playas que aparecen en la aplicación se deberá editar o sustituir el fichero “listado_playas.pkl” contenido en el directorio “documentos” donde se recogen el nombre la playa con sus respectivas coordenadas. En este mismo directorio se encuentra el documento excel con los avistamientos de las medusas (“Datos_Physalia_20171010.xls”) y del que se extraen los datos para generar el gráfico del historial de cada playa.

Si se modifica el fichero con el listado de las playas, los nombres de las mismas deberán coincidir con los que aparecen en el archivo excel si se quiere que el gráfico del historial tenga contenido.

En cuanto al modelo predictivo, se encuentra en la carpeta “static”.

Despliegue de la aplicación

El despliegue se realizó en la plataforma *Heroku* pues es gratuita y los recursos que ofrece son suficientes para los requisitos de nuestro proyecto.

Hay dos ficheros importantes para el despliegue:

- **requirements.txt**: como se ha mencionado anteriormente, recoge todas las dependencias del proyecto y es necesario para que se instalen en la maquina antes de ejecutarlo. Si se añaden bibliotecas se deberá reflejar en este archivo. Dentro del entorno virtual ejecutaremos:


```
pip freeze > requirements.txt
```

De esta manera se actualizarán todas la bibliotecas instaladas en el entorno virtual.

- En segundo lugar, el archivo **Procfile**. Contiene el comando que debe ejecutar la app en el arranque y debe ir sin extensión.

Una vez tengamos estos archivos deberemos seguir los siguientes pasos:

1. Acceder a la pagina de *Heroku*⁵ y crear una cuenta.
2. Entraremos en nuestra cuenta, y en el panel de usuario crearemos una nueva *app*.

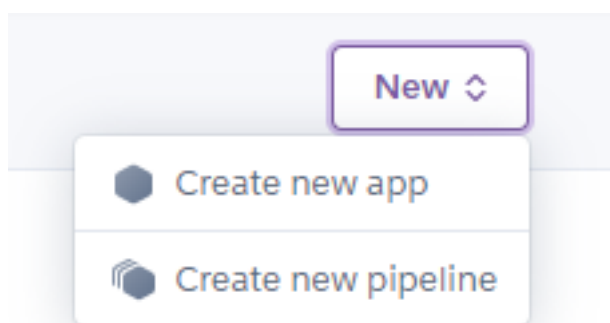



Figura D.2: Nueva app en *Heroku*

3. Añadimos un nombre para nuestra aplicación y ya tendremos nuestra app creada.
4. A la hora de realizar el despliegue tendremos dos opciones: asociar la aplicación a un repositorio existente de *GitHub*, o crear un nuevo repositorio en *Heroku*. En el caso de este proyecto se eligió la opción de asociarlo a un repositorio en *GitHub*.

Si se prefiriese el repositorio en *Heroku* las instrucciones para su creación aparecen en la pagina de la aplicación.


⁵Heroku <https://www.heroku.com/>


App name

jellyfishforecast is available




Choose a region

 Europe

 Add to pipeline...

Create app

Figura D.3: Nombrar nueva app en *Heroku*

 Heroku Git
Use Heroku CLI
  GitHub
Connect to GitHub
  Container Registry
Use Heroku CLI

Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a jellyfishforecast
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a jellyfishforecast
```

Figura D.4: Modos de despliegue en *Heroku*

Generación del modelo

La generación del modelo y las pruebas están recogidas en el segundo repositorio⁶. La forma de descargarlo e instalar todas las bibliotecas necesarias es igual a lo explicado en el anterior repositorio.

⁶Jellyfish Forecast. https://github.com/psnti/Jellyfish_Forecast

En la carpeta “src/Descarga datos” podremos encontrar dos scripts diferentes para poder descargar los datos oceánicos de origen desde *Copernicus*. Uno lo hace a través de una API. Esta opción se descartó, pues en el momento de descargar los datos era bastante nueva y originaba errores, además de tardar demasiado en generar los paquetes de datos. El otro, descarga los datos a través de un FTP. Esta opción descarga un volumen mucho mayor pues se descargan los datos de cada día y de todo el mundo, por lo que tras descargar cada archivo, el script realiza un *crop* dejando únicamente el área de Chile y las variables que nos interesan para el estudio como se explica en el apartado [C.2](#)

Si nos movemos a la carpeta “src/Avistamientos” nos encontramos con los *notebooks* utilizados para la generación del modelo predictivo.

- En la carpeta “#**ExcelsAvistamientosIniciales**” encontraremos los conjuntos de datos de los avistamientos. La versión más reciente es “Datos_Physalia_20171010.xls”.
- El archivo GenerarEstructura.ipynb realiza todo el trabajo de tratar los conjuntos de datos, unirlos y generar una estructura que posteriormente se utilizará para entrenar al modelo. Estas estructuras se guardan en dos formatos distintos. En la carpeta “Excels” están los archivos con formato .xlsx para consulta, pues son más fáciles de visualizar. Por otro lado, en la carpeta “pkls” se encuentran los archivos con formato .pkl que son con los que se trabaja pues su lectura y guardado son más eficientes de leer y guardar.
- La carpeta “PruebasGeneracionEstructuraDatos” contiene varios *notebooks* con pruebas de las diferentes transformaciones que se han realizado a la estructura de datos.
- Por ultimo, “GeneracionModelo” contiene algunas pruebas que se realizaron a la hora del aprendizaje con *Scikit-learn* y tres *notebooks* con las pruebas realizadas con diferentes algoritmos.

D.5. Pruebas del sistema

Pruebas de funcionamiento

A parte de las pruebas por parte de los usuarios, se han realizado pruebas automáticas utilizando la herramienta *Selenium*.

Se ha utilizado la biblioteca para Python con el driver correspondiente para Google Chrome. Elegí esta opción en vez de utilizar la extensión disponible debido a tener una breve experiencia con esta biblioteca.

La versión de Chrome con la que se ha utilizado ha sido la 83.0.4103.106 con el driver correspondiente para esta versión. Si se prueba en el futuro, es probable que la versión del navegador sea diferente. Habría que descargar la versión correspondiente del driver⁷, e introducirla en la carpeta “test/chromedriver” del repositorio *WebJellyfishForecast*.

Para ejecutar las pruebas, accederemos el entorno virtual como se ha explicado en el punto anterior y ejecutaremos el fichero *test.py*.

Python test/test.py

Esta ejecución genera un fichero “test.log” con las salida de los test.

También se han realizado pruebas del correcto funcionamiento de la aplicación en diferentes navegadores web. Google Chrome, Firefox y Opera soportan toda la funcionalidad de la aplicación, mientras que en Microsoft Edge no se cargan los mapas.

Pruebas del modelo

Además de las pruebas de la aplicación web, se realizan pruebas comparativas entre diferentes algoritmos de aprendizaje sobre nuestro conjunto de instancias.

En la figura D.5 podemos observar como el modelo que mejores resultados nos aporta, es el de arboles de decisión. Se ha utilizado como parámetro de puntuación el coeficiente de determinación o R^2 . Este es una medida de la calidad del ajuste del modelo.

Este modelo nos ha otorgado un coeficiente de determinación de 0,25. Esto nos indica que el modelo no realiza una predicción suficientemente precisa de los avistamientos por lo que habría que realizar un mayor tratamiento de los datos previamente a el entrenamiento del mismo.

Se ha realizado una prueba con la playa que contaba con el mayor número de días con avistamientos. El resultado de la predicción respecto a los datos reales se puede observar en la figura D.6

⁷Web para descarga del ChromeDriver. <https://chromedriver.chromium.org/downloads>

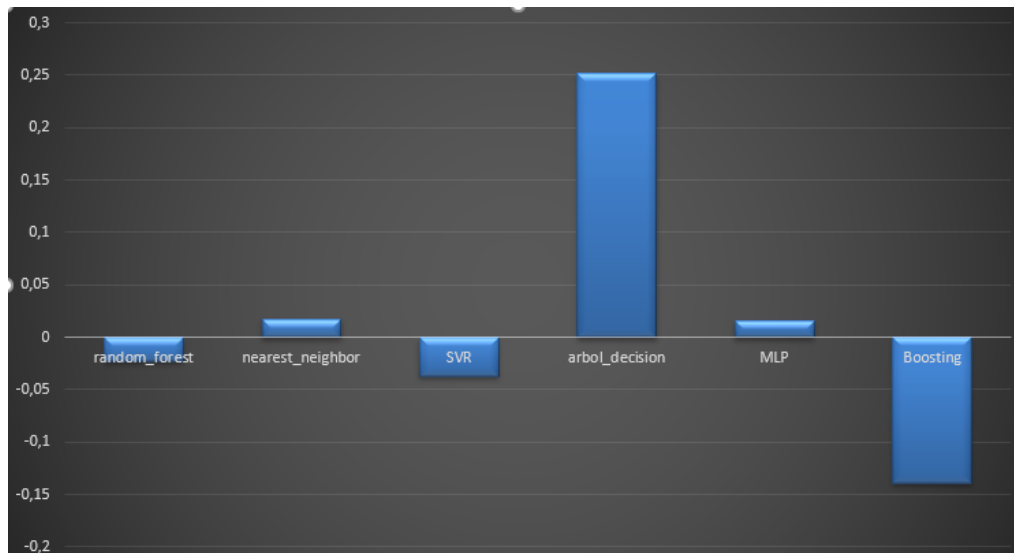


Figura D.5: Comparativa diferentes modelos

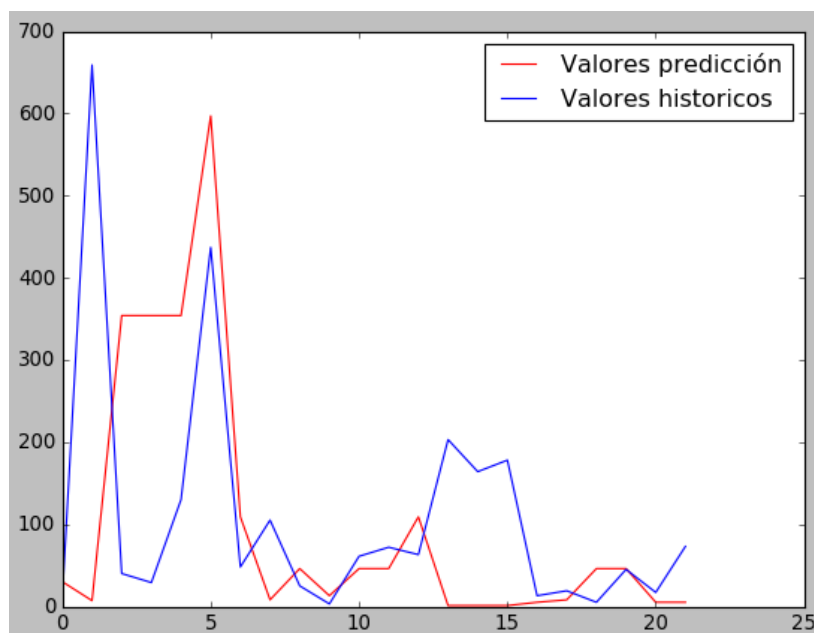


Figura D.6: Comparativa entre predicción y realidad

Apéndice E

Documentación de usuario

E.1. Introducción

Este apéndice recoge todo lo que un usuario debe saber para poder ejecutar la aplicación y los requisitos mínimos que se necesitan.

E.2. Requisitos de usuarios

La aplicación se encuentra desplegada por lo que la única instalación necesaria sería la de un navegador web. En el apartado de pruebas de la aplicación se recogen algunos navegadores que han sido probados y funcionan correctamente [D.5](#).

E.3. Manual del usuario

Una vez dentro de la web ya sea ejecutándolo en un servidor local, o desde página desplegada, las acciones se realizan de la misma manera.

En primer lugar nos encontramos con una página de inicio meramente informativa.



Figura E.1: Página de inicio

Desde ahí, podremos acceder a la pestaña de «Mapas» donde se encuentra la funcionalidad de la aplicación.

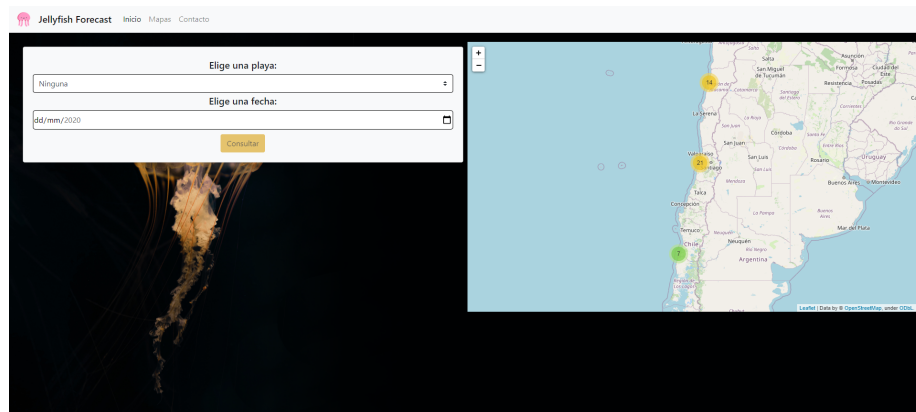
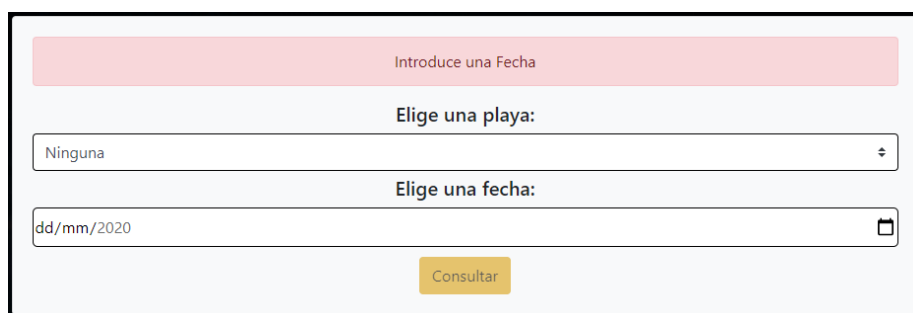


Figura E.2: Página de consulta

Para realizar una predicción, deberemos introducir una fecha y una playa. En caso de seleccionar únicamente una playa sin fecha, aparecerá un mensaje de error indicándonoslo.



Formulario de consulta de datos de una playa. El formulario tiene un fondo gris claro y un borde negro. En la parte superior, hay un campo de texto con el placeholder "Introduce una Fecha" y un fondo rosa pálido. Debajo, hay un campo de selección con el label "Elige una playa:" y el valor "Ninguna". A la derecha de este campo hay un icono de flecha hacia abajo. Debajo de eso, hay un campo de fecha con el label "Elige una fecha:" y el valor "dd/mm/2020". A la derecha de este campo hay un icono de calendario. En la parte inferior, hay un botón amarillo con el texto "Consultar".

Figura E.3: Error en la consulta

Una vez seleccionadas ambas opciones correctamente, al hacer click en «Consultar» nos aparecerán dos gráficos y el mapa realizará un zoom a la playa seleccionada. Estos gráficos se tratan, de la predicción realizada y el historial de avistamientos de la playa seleccionada.

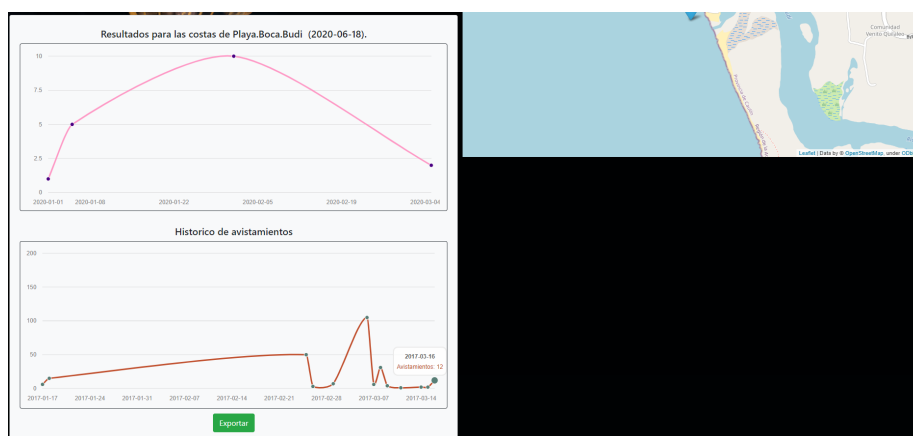


Figura E.4: Página de consulta con gráficos

Una vez realizada la predicción, existe la opción de exportar dicha predicción a un fichero de tipo .xlsx

imagen fichero exportado

Por último, nos encontramos la pestaña de contacto donde se sitúan enlaces a la página de la Universidad, el repositorio y enlaces de contacto.

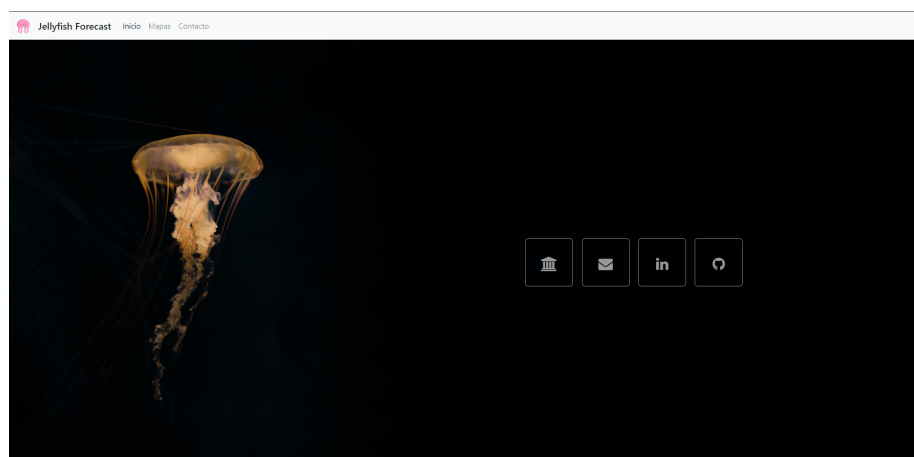


Figura E.5: Página de contacto

Bibliografía

- [1] MVC (model, view, controller) explicado. Library Catalog: codigofacilito.com.