eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería del Software

Trabajo de Fin de Grado

Plataforma web para la segmentación y reducción de series temporales captadas en un escenario de Industria 4.0

Autor

Jokin Ortega Zugazaga

2019

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería del Software

Trabajo de Fin de Grado

Plataforma web para la segmentación y reducción de series temporales captadas en un escenario de Industria 4.0

Autor

Jokin Ortega Zugazaga

Directora

Arantza Illarramendi Etxabe

Resumen

Este Trabajo de Fin de Grado ha sido desarrollado en el ámbito de un proyecto real ubicado en el entorno de la Industria 4.0, en el que participa el grupo de investigación BDI de la UPV/EHU. Concretamente, en este Trabajo de Fin de Grado se ha desarrollado un artefacto software que permite segmentar series temporales y posteriormente reducirlas con el fin de lograr un almacenamiento eficiente de las mismas en la nube de datos. Durante el desarrollo del artefacto se han analizado diferentes algoritmos de segmentación y se ha evaluado su rendimiento sobre series temporales sobre las que se han aplicado técnicas de pre-procesamiento. Por último se ha desarrollado una aplicación web que facilita al ingeniero de datos llevar a cabo la tarea de segmentación.

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutora, Arantza Illarramendi, por brindarme la oportunidad de realizar este Trabajo de Fin de Grado y poder conocer los entresijos de un proyecto de trabajo real. Agradecerle también el apoyo y la confianza recibida durante el desarrollo de todo el proyecto.

En segundo lugar, agradecer a Kevin Villalobos, Borja Díez y Víctor Ramirez, miembros del grupo BDI, por la inmejorable acogida y por haberme hecho sentir como uno más desde el comienzo de mi estancia. Agradecerles la ayuda recibida cada vez que he tenido alguna duda o algún problema que solucionar, y que siempre han estado predispuestos a ayudarme supieran ellos la respuesta o no.

Por último, agradecer a mi familia y amigos que han estado ahí para apoyarme y ayudarme a desconectar siempre que lo he necesitado, y que siempre son esenciales en todos los proyectos que llevo a cabo en mi vida.

Índice general

Anexos

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

Vivimos en una época en la que la relevancia de los sistemas de información es mayor que nunca en cualquier ámbito, debido a la alta capacidad de cómputo de los actuales ordenadores y a los avances en materia de gestión del almacenamiento. La capacidad es tal, que se pueden manejar ingentes volúmenes de datos de una manera eficiente, recogiéndolos en "bruto"[1] y, con un pre-procesado y análisis, se pueden convertir en conocimiento útil para todo tipo de ámbitos y aplicaciones.

Este hecho abre una inmensidad de posibilidades de aplicación en todo tipo de sectores, siendo uno de ellos el sector de la industria [Li et al., 2019], en particular, ha contribuido al auge del concepto de la Fábrica inteligente, más conocida como Industria 4.0[2] o cuarta revolución industrial.

La captura y el análisis de Big Data [Madden, 2012] en la Industria 4.0 es un campo de investigación en auge por su potencial para mejorar los procesos industriales y la creación de nuevas líneas de negocio en dicho sector. Los principales retos son el análisis, administración y manipulación inteligente de grandes cantidades de datos que, como es lógico, no son posibles de analizar al momento y, por tanto, se tienen que almacenar para su posterior tratamiento.

El formato normalizado de dichos datos obtenidos de las máquinas sensorizadas son las

[1] El concepto datos en "bruto" (*raw data* en inglés) hace referencia a los datos directamente capturados del medio en el que se recogen, sin haberles sido aplicado ningún tipo de procesado ni haber sido transformados.

[2] Se trata de fábricas informatizadas en las que software, sistemas de análisis de datos, sensores y electrónica están conectados e interactúan con los procesos productivos clásicos de las máquinas

series temporales. Se tratan de series de datos en los que cada punto representa el valor obtenido de un sensor en un momento del tiempo (normalmente un segundo). El análisis de estas series es esencial para la mejora de los procesos industriales y la detección de anomalías, pero a su vez requieren de un espacio de almacenamiento considerable ya que no existe posibilidad de analizarlos a medida que se recogen.

Aunque el espacio de almacenamiento se ha abaratado en gran medida en los últimos años, sigue siendo tal la cantidad de datos recogidos que es conveniente un proceso de reducción de los mismos antes de poder almacenarlos en disco. Existen diversas técnicas de reducción que reducen en mejor o peor medida una serie temporal dependiendo de su morfología. Aún así, existen series temporales que pueden ser muy largas y que presentan diferentes morfologías dentro de una misma. Esto nos lleva a la posibilidad de segmentar dichas series y utilizar el algoritmo de reducción óptimo para cada uno de los segmentos generados, pudiendo llevarnos esto a un mayor aprovechamiento del almacenamiento disponible y, por tanto, a un ahorro en los costes finales.

El Trabajo de Fin de Grado (TFG a partir de ahora) estará centrado en ese ámbito de Preprocesado/Administración de los datos, concretamente en la segmentación de las series temporales previamente mencionada. En él se desarrollará un análisis de distintos algoritmos de segmentación de series temporales y de los beneficios/inconvenientes que pueden llegar a presentar a la hora de optimizar la reducción de las mismas. Para completarlo, se desarrollara una plataforma web en la que un ingeniero de datos podrá segmentar una serie temporal con distintos algoritmos de segmentación, pudiendo posteriormente comparar y analizar, tanto los resultados de las segmentaciones, como los resultados de la aplicación de distintas técnicas de reducción a los segmentos generados, comparando los resultados finales con los de la reducción de la serie original.

El contenido de esta memoria está estructurado en 10 capítulos, que están organizados de esta manera:

- Capítulo 2: Se presenta el contexto del proyecto, con la información necesaria para comprender tanto el entorno en el que el proyecto se ha llevado a cabo como los conceptos técnicos relacionados con el mismo.
- Capítulo 3: Se comenta y se plasma toda la información relacionada con los objetivos y la planificación inicial del proyecto.
- Capítulo 4: Se expone el análisis de distintos algoritmos de segmentación de series temporales, analizando sus ventajas e inconvenientes tanto en el contexto de las

series trabajadas por el grupo de investigación BDI de la UPV/EHU[3](en adelante Grupo BDI) como en el contexto de otro tipo de series temporales de otros ámbitos.

- Capítulo 5: Se analiza la validez de la segmentación de series temporales para mejorar la reducción de las mismas.
- Capítulo 6: Se analizan los requisitos necesarios para el posterior desarrollo de la aplicación web.
- Capítulo 7: Se presenta el contenido relacionado con el diseño y desarrollo de la plataforma web y los elementos relacionados.
- Capítulo 8: Se exponen y detallas las distintas pruebas realizadas durante el desarrollo del proyecto.
- Capítulo 9: Se muestra el Seguimiento y Control llevado a cabo durante el proyecto.
- Capítulo 10: Se presentan las conclusiones generales del proyecto y las líneas de avance futuras relacionadas con el mismo.

[3]Grupo BDI: http://bdi.si.ehu.es/bdi/

2. CAPÍTULO

Contexto

La industria 4.0 requiere de una perfecta integración entre diversos elementos. El software, los sistemas de análisis de datos, los sensores y la electrónica están conectados e interactúan con los procesos productivos clásicos de las máquinas y, por tanto, es necesaria la colaboración de distintos participantes para lograr un perfecto funcionamiento de todo el proceso de implantación, de extracción de la información y de su posterior análisis.

En este ámbito es donde el grupo BDI trabaja, junto con distintas empresas, en el dominio de un proyecto real de recogida y tratamiento de datos de una empresa de fabricación de botellas de plástico. Como este Trabajo de Fin de Grado se ubica en ese contexto, resulta conveniente conocer de antemano qué empresas o entidades son las que trabajan y qué función ocupa cada una de ellas en esta colaboración.

En este capítulo se presentarán los elementos necesarios para entender y conocer el contexto que engloba al proyecto, ya sea el contexto del dominio como el contexto tecnológico o técnico.

2.1. Contexto del dominio

Se presentan seguidamente las características principales de las empresas involucradas y el trabajo ya realizado por el grupo BDI.

2.1.1. Empresas involucradas

Urola Solutions

Urola Solutions[1] es una empresa que se encarga de de construir maquinaria industrial utilizada en el proceso de fabricación de botellas de plástico. Estas máquinas van provistas de sensores en distintas zonas, que adquieren valores relevantes sobre distintos indicadores tales como la temperatura o velocidad del motor.

En este caso, a las máquinas que construyen para la fabricación de botellas se les llama extrusoras puesto que, la formación de la botella, se realiza a través de un proceso llamado extrusión de polímeros.

Bowler Plastics Ltd. y Fábrica de Jabón La Corona

Bowler Plastics Ltd.[2] y La Fábrica de Jabón La Corona[3] son las empresas que utilizan las máquinas desarrolladas por Urola para producir botellas de plástico.

Savvy Data Systems

Savvy Data Systems[4] es una empresa alojada Donostia/San Sebastián que proporciona monitorización avanzada y servicios Big Data para la industria. Es la encargada de desarrollar el software que recoge y codifica los datos que generan los sensores de la maquinaria de Urola instalada en las empresas que producen las botellas, para posteriormente transferirlos al grupo BDI, donde se procesan y se trabaja con ellos.

Estos datos, en forma de series temporales, serán los utilizados en este TFG, con lo que dotarán al trabajo de una visión mayor de realidad.

2.1.2. Sistema web I4TSPS

En el contexto de las empresas mencionadas en el apartado anterior, el grupo de investigación BDI ha desarrollado una plataforma web llamada I4TSPS[5], orientada a los inge-

[1] Urola Solutions: https://www.urolasolutions.com/es/
[2] Bowler Plastics Ltd.: http://www.bowler.co.za/
[3] Fábrica de Jabón La Corona: https://www.lacorona.com.mx/
[4] Savvy Data Systems: http://www.savvydatasystems.com
[5] I4TSPS: https://fdai-b5221.firebaseapp.com/#/demo

nieros de datos, y que ofrece distintas herramientas relacionadas con el preprocesado de las series temporales.

Las funcionalidades principales soportadas por dicha plataforma son las siguientes:

- Limpieza de los datos.
  - Tratamiento de valores ausentes (*missing values*) y de valores atípicos (*outliers*).
  - Eliminación de ruido.
- Reducción del volumen de los datos. Proporciona un modelo que es capaz de recomendar una técnica adecuada de reducción según las características de los datos procesados (ver Fig. 2.1).

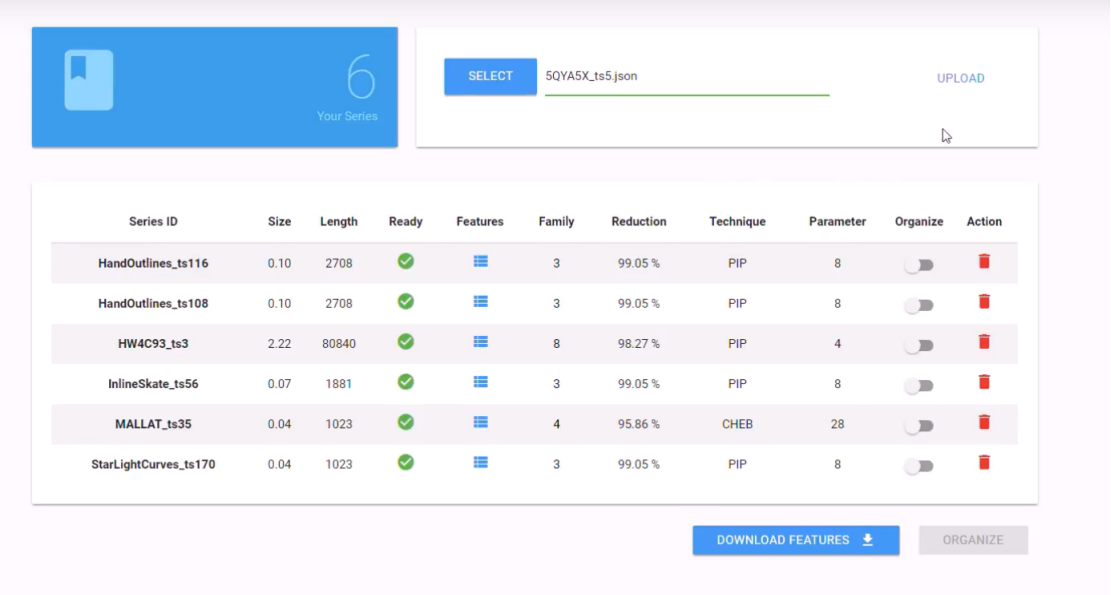


Figura 2.1: Interfaz de uno de los apartados de la plataforma I4TSPS donde se muestra una lista de series reducidas presentando datos como el porcentaje de reducción o el algoritmo utilizado para reducir, junto con otros datos relacionados con la serie temporal.

Con el fin de incrementar las funcionalidades de dicha plataforma, se plantea el interés de integrar la funcionalidad que se realizará en la plataforma web del TFG, facilitando así la tarea de segmentación de datos y de comparativas de reducción de los datos segmentado respecto a los datos originales.

2.2. Contexto tecnológico

Seguidamente se presentan los distintos orígenes de datos utilizados en el proyecto y las distintas técnicas de reducción de las implementadas por el grupo BDI que serán utilizadas.

2.2.1. Origen de las series temporales

Para poder desarrollar las pruebas y los análisis pertinentes necesitamos datos con los que llevarlas a cabo. Trabajaremos con dos tipos de bancos de datos: el primero con series proporcionadas por el grupo BDI y el segundo con datos obtenidos del *UCR Time Series Classification Archive* (*UCR Archive* a partir de ahora). [Dau et al., 2018]

Tal y como se ha comentado previamente, este TFG está ubicado en el contexto de un proyecto relacionado con la Industria 4.0 en el que el grupo BDI recibe datos reales de empresas productoras de botellas de plástico. Por ende, los datos proporcionados por el grupo BDI serán los principales dentro de la investigación ya que son los que nos permiten evaluar los resultados teniendo en cuenta un entorno de producción real. Serán los datos que marcarán los resultados respecto a la elección de un mejor algoritmo de reducción o a la ordenación de los mismos con respecto a la utilidad en el ámbito de trabajo del grupo BDI.

Por otra parte, las series temporales obtenidas del *UCR Archive* serán utilizadas para poder interpretar resultados de los algoritmos con datos que no sean los manejados por el grupo BDI. Esto nos permitirá realizar comparaciones y también poder obtener otros escenarios en los que un algoritmo que partía como menos óptimo acabe siendo el mejor para ese escenario en concreto.

2.2.2. Algoritmos de reducción

Tanto en la realización de ciertas pruebas durante el proyecto como en el desarrollo de la web, será necesario utilizar las implementaciones que el grupo BDI tiene sobre ciertos algoritmos de reducción de series temporales. Por ello, es importante conocer cuales serán los algoritmos seleccionados y utilizados para realizar las pruebas pertinentes y posteriormente añadirlos a las funcionalidades de la plataforma web. Por cada uno de ellos se

introducirá de manera resumida su funcionamiento y se comentará para qué tipo de series o para qué casos resulta más adecuado.

Los principales algoritmos seleccionados son:

- **Discrete Fourier transform (DFT):** La idea general de este algoritmo es la de transformar una serie temporal (datos secuenciales finitos en el dominio del tiempo) convirtiéndola en una representación de la serie en el dominio de la frecuencia, quedándose con las frecuencias más relevantes de dicha representación. Estas frecuencias se plasman con unas funciones, cuyos coeficientes son los utilizados para llevar a cabo la reducción de la serie [Agrawal et al., 1993].
  Algoritmo muy polivalente y capaz de obtener buenos resultados en distintos tipos de series temporales, ideal también para funciones similares a gráficos de frecuencias.
- **Perceptually Important Points (PIP):** La idea de este algoritmo es la de determinar los n puntos más importantes de la serie temporal [Fu-Lai Chung et al., 2004]. Primero se determinan el primer y último punto del PIP (punto 1 y punto n), que serán el primero y el último punto de la serie original respectivamente. El tercer punto será el correspondiente al que tenga la mayor distancia respecto a los dos puntos iniciales. A partir de ahí los siguientes puntos restantes, hasta conseguir el total de n puntos, serán los puntos con mayor distancia a los puntos adyacentes. Para determinar dicha distancia se calcula la distancia entre el posible punto x y la línea que conecta los dos puntos adyacentes. En la figura 2.2 se muestra el proceso del PIP con 7 puntos.
  Este algoritmo es ideal en series que no tengan excesivos puntos y sobre todo en segmentos que nos podemos encontrar en los que hay muchas líneas rectas, que permitiría al PIP reducirlas con poco error.
- **Run-Length Encoding (RLE):** Este algoritmo se basa en aprovechar la repetición de un mismo valor durante varios puntos seguidos. Se caracteriza por ser un algoritmo *lossless* (sin pérdida) ya que los valores obtenidos con la reconstrucción son idénticos al de la serie original. Esto se debe a que a la hora de realizar la reducción, solo tiene que almacenar cada valor de la serie que vaya apareciendo junto con el número de veces seguidas que aparece hasta que cambie de valor [Robinson and Cherry, 1967].
  Algoritmo ideal para series en las que los valores se repitan de manera continuada durante mucho tiempo. Por ejemplo, resulta ideal para reducir series que represen-

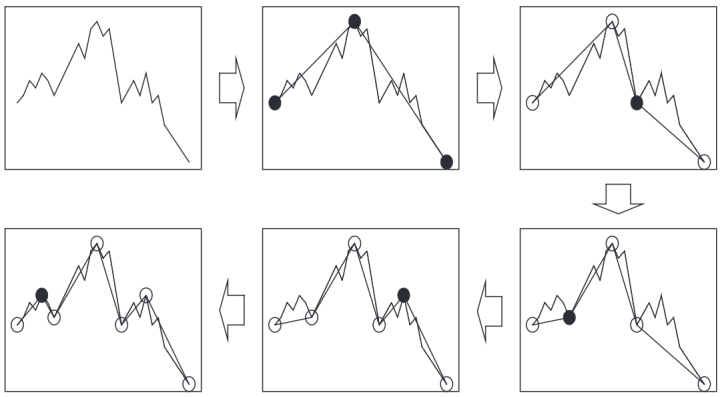


Figura 2.2: Ejemplo de identificación con el PIP de los 7 puntos importantes de la serie.

tan el estado de una máquina (on y off) ya que la máquina se mantiene un largo tiempo en cada uno de esos estados y el valor que devuelve en la serie será todo el rato 0 o 1.

- **Chebyshev Polynomials (CHEB):** La idea de este algortimo reside en representar la serie original mediante los polinomios de Chebyshev [Cai and Ng, 2004]. Funciona de forma similar a la DFT, con la diferencia de que en este caso la serie se representa con los polinomios de Chebyshev, unos polinomios ortogonales relacionados con la fórmula de Moivre[6]. La reducción de la serie se realiza a través de los coeficientes de Chebyshev, asociados a los polinomios mencionados.
  Este algoritmo es ideal para series temporales que sean similares a la representación gráfica de una función matemática.

[6]Fórmula de Moivre: Teorema para poder elevar números complejos en forma polar a una potencia de n.

3. CAPÍTULO

Objetivos y Planificación

El primer paso a la hora de afrontar un proyecto de tamaño importante es el de definir y plantear los objetivos del mismo, junto con la realización de una planificación que permita llevarlos a cabo en unos plazos y pasos razonables.

En este apartado se detallan dichos elementos junto con otros necesarios para ayudar a la realización y finalización del proyecto.

3.1. Alcance

Este proyecto tiene como fin analizar el potencial que pueda tener la segmentación de series temporales a la hora de optimizar y mejorar la posterior reducción de las mismas. Para ello, se desarrollará una plataforma web en la que los ingenieros de datos puedan trabajar con la segmentación de series temporales y sean capaces tanto de analizar ventajas o desventajas de los distintos algoritmos de segmentación como de comparar la reducción de la serie segmentada con la serie original.

Dicha web tendrá la posibilidad de cargar una serie temporal, segmentarla con distintos algoritmos de segmentación implementados y comprobar de manera rápida qué segmentación es la más conveniente para dicha serie. Se podrá también reducir cada segmento generado con cualquiera de los algoritmos implementados por el grupo BDI en su servicio web de reducción y decidir cuál de ellos es el óptimo para cada segmento. Como funcionalidad final se podrá comparar el resultado de haber reducido los segmentos de manera

separada con la reducción de la serie original, para comprobar si en esa serie merece la pena la segmentación para obtener una mejor versión reducida.

3.1.1. Objetivos del proyecto

El objetivo principal de este proyecto es el de la creación de una plataforma web[1] que permita en primer lugar, segmentar series temporales industriales y posteriormente, analizar dichos segmentos con el fin de seleccionar la técnica de reducción del tamaño más adecuada para cada segmento y así lograr optimizar el espacio necesario para almacenar las series completas. Para ello se realizará un análisis previo tanto de los principales algoritmos de segmentación que se han definido en la literatura especializada, como del potencial de reducción de diferentes técnicas aplicadas a los segmentos obtenidos.

Como objetivo secundario, destacaría la realización de un servicio web de segmentación de series temporales que sea capaz de integrarse en el servicio web ya en producción de grupo BDI. La creación del servicio como tal podría considerarse ligada al objetivo principal del proyecto pero la idea está en poder integrarlo en ese producto real mencionado.

Otro de los objetivos secundarios, también importante, sería el del aprendizaje relacionado con las series temporales, su reducción y su segmentación. Debido a la importancia de estos elementos en el producto final, es de gran importancia el conocer los detalles y los elementos que componen y rodean a los conceptos que se van a utilizar.

Como último objetivo secundario estaría el hecho de poder conseguir llevar a cabo y gestionar un proyecto de esta magnitud para poder demostrar las capacidades obtenidas durante los cuatro años de estudios del grado y tener la convicción de estar capacitado en cualquier tipo de proyecto que en el futuro se plantee.

3.1.2. Exclusiones

Del alcance del proyecto se excluirá la integración del sitio web en la plataforma I4TSPS desarrollada por el grupo BDI, quedando como una aplicación independiente pero que sí se servirá de las API REST de reducción de series desarrolladas por el grupo BDI como de la API de segmentación que será desarrollada en este mismo proyecto.

[1] Time series segmentation: https://timeseriessegmentation.000webhostapp.com

3.1.3. Estructura de Desglose de Trabajo (EDT)

Una vez definido el alcance del proyecto, es primordial definir una buena Estructura de Desglose de Trabajo (EDT, Fig. 3.1) para identificar y organizar las diferentes fases del proyecto junto con los principales paquetes de trabajo. Esto nos permitirá una mejor organización del trabajo gracias a la división del mismo en pequeñas tareas más fáciles de manejar y planificar.

En este caso el proyecto está dividido en distintas fases, que son las que encontramos en el primer nivel del diagrama. Dichas fases están divididas en paquetes de trabajo que englobarán distintas tareas, siendo estas definidas en en este mismo apartado.

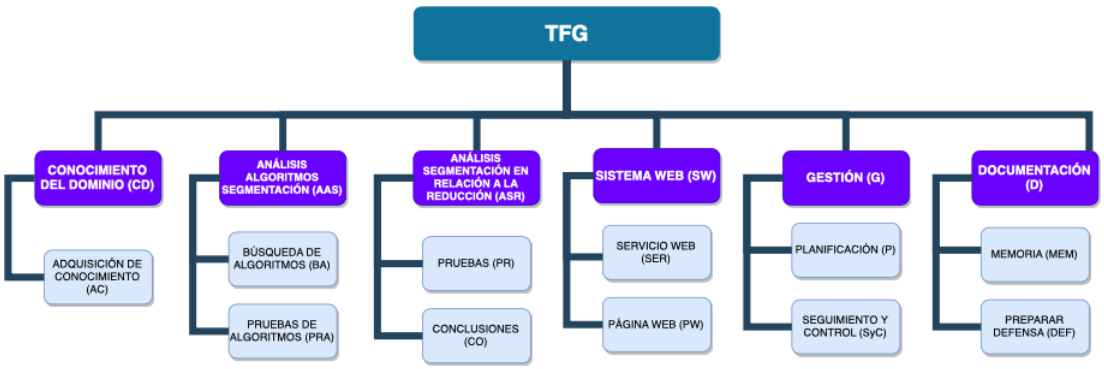


Figura 3.1: Diagrama EDT con las fases y los paquetes de trabajo correspondientes al proyecto.

A continuación se detallan y se explican los diferentes paquetes de trabajo y tareas contenidas en las fases presentadas en la EDT:

Conocimiento del Dominio (CD):

El paquete de trabajo ***Adquisición de Conocimiento (AC)*** agrupa todas las tareas necesarias para adquirir los conocimientos necesarios del dominio en el que está ubicado el proyecto. Las tareas son:

- **AC1:** Conocer el contexto en el que el grupo BDI trabaja como, por ejemplo, las empresas involucradas en el proyecto o los objetivos del mismo.
- **AC2:** Conocer los formatos de las series temporales con las que se trabaja en el grupo BDI.

- **AC3:** Conocer la plataforma web I4TSPS desarrollada por el grupo BDI e identificar en el flujo de trabajo de la misma en qué paso iría la segmentación de la serie temporal.

Análisis Algoritmos de Segmentación (AAS):

El paquete de trabajo Búsqueda de Algoritmos (BA) engloba las tareas relacionadas con la búsqueda de los algoritmos de segmentación principales definidos en la literatura especializada de este ámbito.

- **BA1:** Recogida de los principales algoritmos de segmentación identificando y archivando los documentos y ficheros relacionados con los mismos.
- **BA2:** Analizar la factibilidad del uso de cada algoritmo en nuestro problema (ofrece implementación, es solo teórico, se distancia en exceso de nuestros objetivos...)
- **BA3:** Realizar la selección definitiva de los algoritmos en base a los criterios planteados en la tarea BA2.

El paquete de trabajo PRuebas de Algoritmos (PRA) agrupa las tareas relacionadas con las pruebas a realizar para poder identificar el/los algoritmo/s más adecuado/s para el contexto en el que nos encontramos. Las tareas son las siguientes:

- **PRA1:** Definir las pruebas a realizar con los algoritmos seleccionados.
- **PRA2:** Implementar los algoritmos que no estén implementados para poder realizar las pruebas posteriores.
- **PRA3:** Realizar las pruebas con los distintos algoritmos y documentarlas.
- **PRA4:** Analizar los resultados obtenidos y sacar conclusiones de los mismos, determinando los puntos negativos y positivos de cada algoritmo.

Análisis de la Segmentación en relación a la Reducción (ASR):

El paquete de trabajo Pruebas (PR) contiene todas las tareas que permitan llevar a cabo las pruebas necesarias para demostrar si la segmentación de las series temporales es útil y beneficioso para la reducción de las mismas. Las tareas son las siguientes:

- **PR1:** Definir las pruebas a realizar y los tipos de series con las que probar para la consecución de los objetivos de esta fase del proyecto.

- **PR2:** Realizar los scripts o programas que se crean convenientes para la automatización o la realización más rápida de las pruebas.
- **PR3:** Realizar las pruebas con los distintos tipos de series.

El paquete de trabajo Conclusiones (CO) agrupa las tareas relacionadas con el análisis de los resultados obtenidos en las pruebas y la generación de las conclusiones pertinentes que permitan identificar las ventajas o no que pueda tener la segmentación de series temporales. Dichas tareas son:

- **CO1:** Analizar los resultados obtenidos en las pruebas.
- **CO2:** Plasmar las conclusiones pertinentes obtenidas de los resultados analizados.

Sistema Web (SW):

En el paquete SERvicio web (SER) se agrupan todas las tareas relacionadas con el diseño y la implementación del servicio web que permite segmentar series temporales. Las tareas son:

- **SER1:** Diseño del servicio web, estableciendo el formato de los datos que recibirá y el formato en que devolvera los segmentos correspondientes.
- **SER2:** Elaboración del servicio web, aplicando los algoritmos recogidos o implementados en el paquete de trabajo PRA.
- **SER3:** Realización de pruebas en local del funcionamiento correcto del servicio, corregir si es necesario.
- **SER4:** Desplegar el servicio al exterior en un servidor personal para comprobar el correcto funcionamiento.
- **SER5:** Integrar el servicio web con el servicio ya implementado por el grupo BDI y comprobar su funcionamiento.

En el paquete Página Web (PW) se reúnen las tareas relacionadas con el desarrollo de la página web que utilizará todos los elementos previamente trabajados e implementados para ofrecer la funcionalidad requerida como objetivo del proyecto. Las tareas que engloba son:

- **PW1:** Análisis de requisitos identificando las necesidades y las funcionalidades que tendrá la web.

- **PW2:** Diseño general de de la web, determinando los elementos visuales y diseñando unos bocetos preliminares de los distintos apartados.
- **PW3:** Elaboración de los casos de uso principales y más importantes de la web.
- **PW4:** Preparación del entorno de desarrollo de la web, instalando y configurando los programas y herramientas necesarios.
- **PW5:** Elaborar la página web siguiendo los requisitos planteados.
- **PW6:** Revisión de lo elaborado hasta el momento y realizar algún cambio en los requisitos si es necesario.

Gestión (G):

El paquete de trabajo Planificación (P) engloba todas las tareas relacionadas con la planificación inicial del proyecto, así como las tareas que fuesen necesarias para modificar o actualizar dicha planificación. Estas tareas son:

- **P1:** Identificación de requisitos, recogida de información básica y resolución de dudas necesarias para proceder con la planificación.
- **P2:** Realización de la planificación inicial.
- **P3:** Modificación de la planificación inicial si así fuera necesario.

El paquete de trabajo Seguimiento y Control (SyC) agrupa todas las tareas necesarias para mantener el desarrollo adecuado durante todo el proyecto, comparando el plan previsto con lo que vaya ocurriendo para poder detectar posibles cambios y modificaciones. Las tareas del mismo son:

- **SyC1:** Reuniones y comentarios con la tutora a lo largo del proyecto.
- **SyC2:** Elaboración de un documento en el que anotar las dedicaciones a lo largo del proyecto.
- **SyC3:** Recopilación de información relevante o dudas sobre el proyecto.
- **SyC4:** Contraste de la información de seguimiento con la planificación, identificación de desviaciones significativas y de riesgos emergentes.

Documentación (D):

El paquete de trabajo Memoria (MEM) agrupa todas las tareas relacionadas con la realización de la memoria del Trabajo de Fin de Grado. Las tareas son:

- **MEM1:** Preparar el entorno para desarrollar la memoria con LaTeX en Overleaf, importando la plantilla proporcionada por la facultad y adecuándola a nuestras necesidades.
- **MEM2:** Desarrollar la memoria del proyecto.

El paquete de trabajo Preparar Defensa (DEF) contiene las tareas relacionadas con la preparación de la defensa del Trabajo de Fin de Grado. Dichas tareas son:

- **DEF1:** Identificar los elementos y conceptos que se presentarán en la defensa.
- **DEF2:** Generar los elementos audiovisuales que se consideren oportunos como apoyo en la defensa.
- **DEF3:** Preparar la defensa.

3.2. Periodos de realización de las tareas

3.2.1. Dependencias entre tareas

Para representar las dependencias entre las diferentes tareas se ha elaborado un diagrama de dependencias en el que se puede visualizar qué tareas y/o paquetes de trabajo requieren de la finalización de otra para su comienzo. El diagrama generado es el representado en la figura 3.2.

3.2.2. Periodo de desarrollo de las tareas e Hitos

Hay que tener en cuenta que este proyecto no se llevará a cabo a tiempo completo en parte de los meses en los que está planificado ya que en paralelo al mismo se estará cursando una asignatura del grado y realizando prácticas en una empresa. Esto hace que el proyecto se vaya a alargar más de lo que lo haría si se desarrollase a tiempo completo.

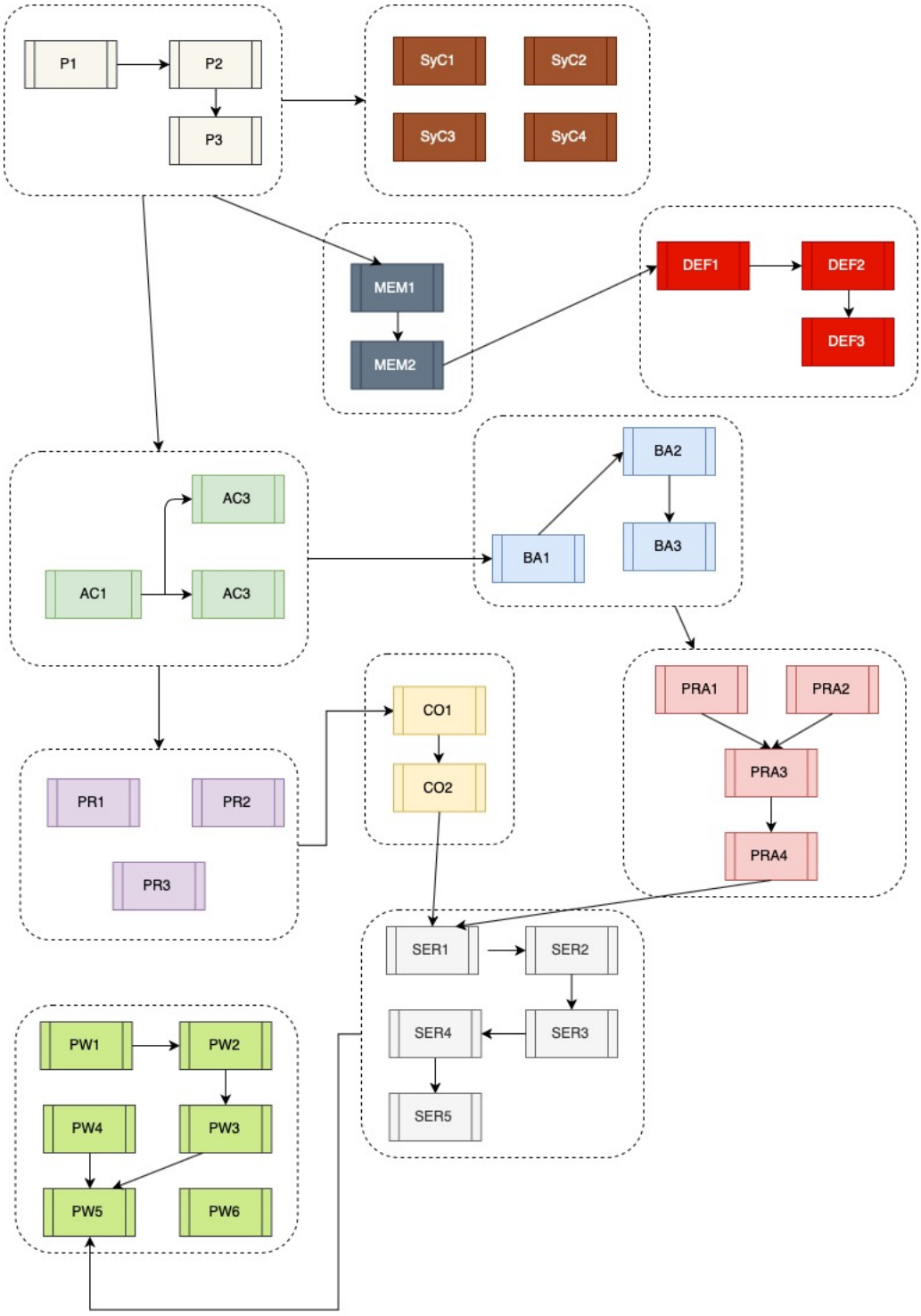


Figura 3.2: Diagrama de dependencias entre tareas y paquetes de trabajo.

Aún así, a partir del mes de Mayo existirá disponibilidad a tiempo completo así que las horas dedicadas diariamente pasarán a ser mayores y deberían permitir terminar el Proyecto en una fecha compatible con el objetivo de defender el Trabajo de Fin de Grado en la convocatoria de Junio.

Todo esto se ha reflejado en un diagrama de Gantt (figuras 3.3 y 3.4), que permite la visualización de los periodos de realización y finalización de las diferentes tareas.

3.2.3. Estimación de dedicación a cada tarea

A continuación, en la tabla 3.1, se presentan las horas estimadas para cada una de las tareas presentadas en el apartado 3.1.3 junto con las totales correspondientes a cada paquete de trabajo y al proyecto completo.

Tabla 3.1: Dedicaciones estimadas en horas para cada tarea del proyecto

Tareas	Estimación (horas)
Total Trabajo de Fin de Grado	343
Conocimiento del dominio (CD)	27
Adquisición de conocimiento (AC)	27
AC1: Conocer contexto del dominio	5
AC2: Conocer formatos series temporales	20
AC3: Conocer plataforma I4TSPS	2
Análisis algoritmos de segmentación (AAS)	67
Búsqueda de algoritmos (BA)	17
BA1: Recogida de los principales algoritmos de segmentación	10
BA2: Analizar la factibilidad del uso de cada algoritmo	5
BA3: Realizar la selección definitiva	1
Pruebas de algoritmos (PRA)	50
PRA1: Definir las pruebas a realizar	5
PRA2: Implementar los algoritmos que no estén implementados	30
PRA3: Realizar las pruebas con los distintos algoritmos	10
PRA4: Analizar los resultados obtenidos y sacar conclusiones	5
Análisis de segmentación en relación a la reducción (ASR)	61
Pruebas (PR)	55

Continúa en la siguiente página

Continuación de la Tabla 3.1

Tareas	Estimación (horas)
PR1: Definir las pruebas a realizar	5
PR2: Realizar scripts convenientes para las pruebas	20
PR3: Realizar las pruebas con los distintos tipos de series	30
Conclusiones (CO)	7
CO1: Analizar los resultados obtenidos en las pruebas	2
CO2: Plasmar las conclusiones obtenidas de los resultados	5
Sistema web (SW)	103
Servicio web (SER)	20
SER1: Diseño del servicio web	1
SER2: Elaboración del servicio web	10
SER3: Realización de pruebas en local	2
SER4: Desplegar el servicio en un servidor personal	5
SER5: Integrar el servicio web en el del grupo BDI	2
Página web (PW)	93
PW1: Análisis de requisitos	2
PW2: Diseño general de de la web	5
PW3: Elaboración de los casos de uso principales	5
PW4: Preparación del entorno de desarrollo de la web	1
PW5: Elaborar la página web	70
PW6: Revisión de lo elaborado	10
Gestión (G)	36
Planificación (P)	15
P1: Identificación de requisitos	2
P2: Realización de la planificación inicial	10
P3: Modificación de la planificación inicial	3
Seguimiento y control (SyC)	21
SyC1: Reuniones con la tutora a lo largo del proyecto	10
SyC2: Elaboración de documento de anotación de horas	2
SyC3: Recopilación de información relevante	4
SyC4: Contraste del seguimiento con la planificación	5
Documentación (D)	49
Memoria (MEM)	41

Continúa en la siguiente página

Continuación de la Tabla 3.1

Tareas	Estimación (horas)
MEM1: Preparar el entorno para desarrollar la memoria	1
MEM2: Desarrollar la memoria	40
Preparar defensa (DEF)	8
DEF1: Identificar conceptos que se presentarán	1
DEF2: Generar elementos audiovisuales como apoyo	5
DEF3: Preparar la defensa	2

3.3. Gestión de riesgos

La gestión de riesgos es fundamental en un proyecto ya que permite, si han sido previamente identificados, establecer un plan de actuación acorde a los riesgos y poder así mitigar el impacto que puedan tener sobre el proyecto.

En este caso estos son los riesgos que han sido identificados, detallando en cada caso la probabilidad de que ocurra, el impacto que puede generar y la acción que se haya realizado para prevenirse.

3.3.1. Problemas al compaginar el proyecto con las asignaturas y las prácticas

- **Probabilidad:** Baja.
- **Impacto:** Podría retrasar la finalización del proyecto.
- **Plan de acción:** No se ha realizado ninguna acción especial dado que hay un margen razonable de tiempo para la finalización del proyecto, que no debería verse afectado.

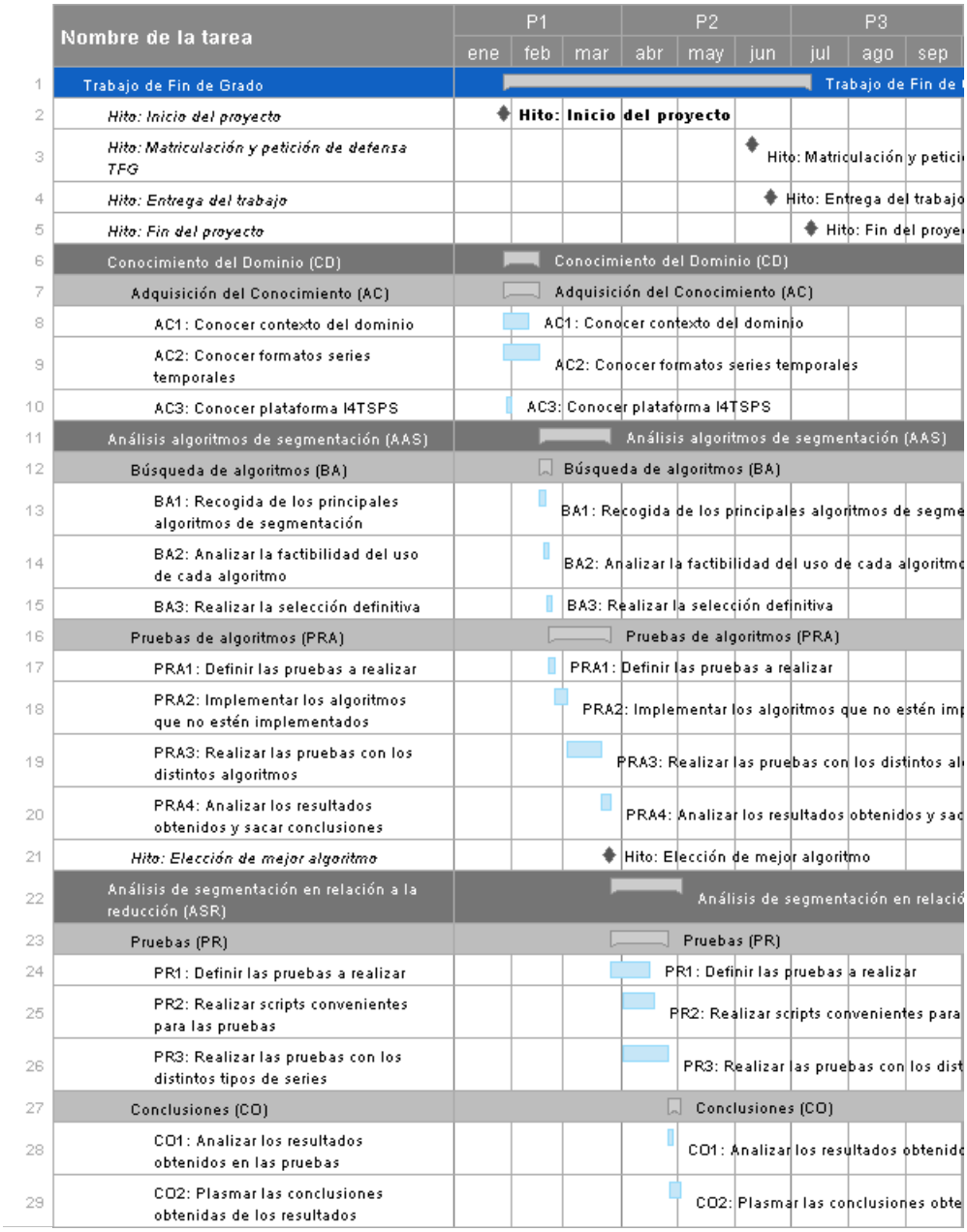


Figura 3.3: Diagrama Gantt con los tiempos de realización de las tareas y los paquetes de trabajo del proyecto.

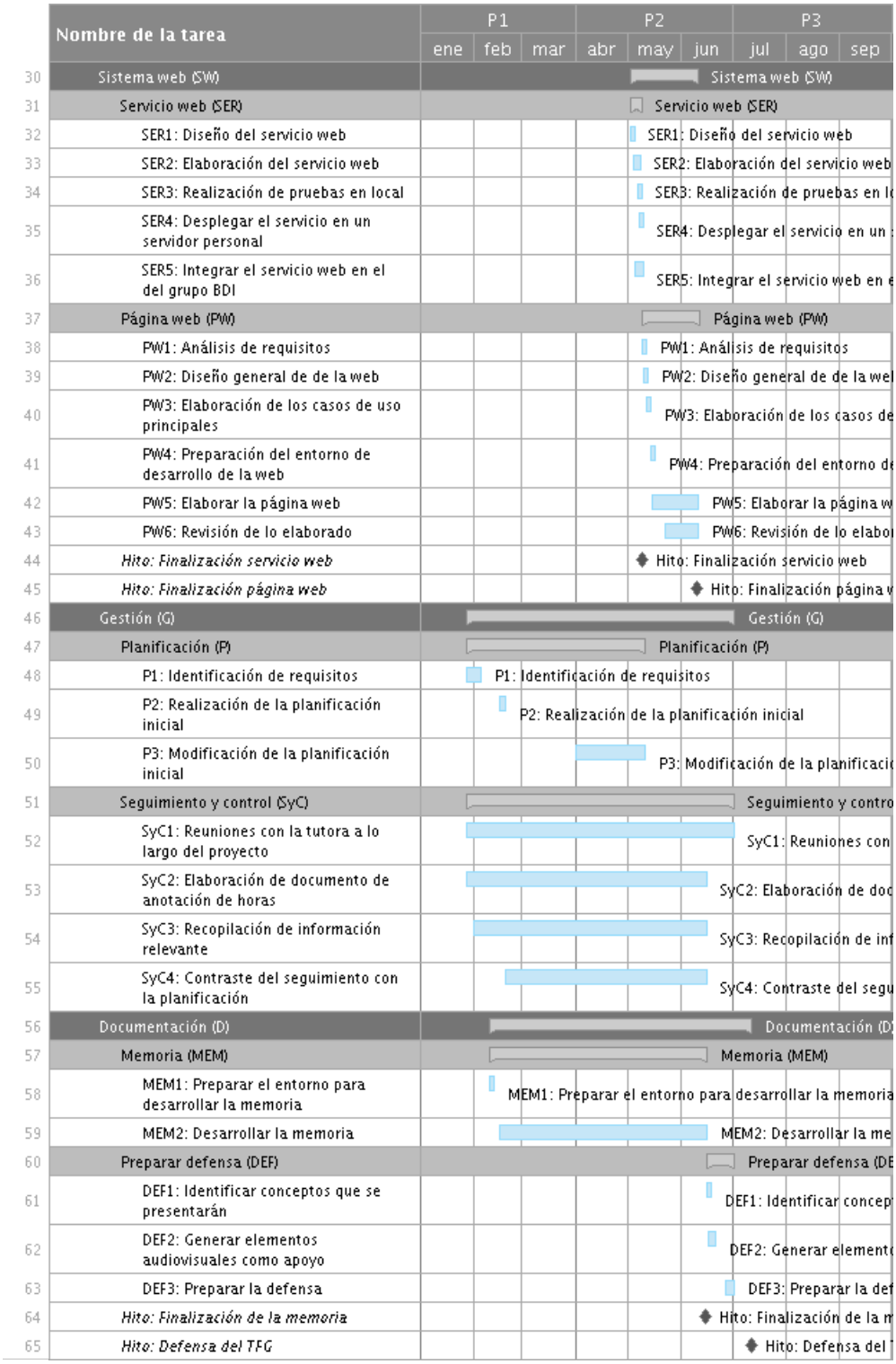


Figura 3.4: Diagrama Gantt con los tiempos de realización de las tareas y los paquetes de trabajo del proyecto.

3.3.2. Dificultad en el aprendizaje de los conceptos relacionados con la compresión las series temporales (formatos, tratamiento, formateado...).

- **Probabilidad:** Bajo.
- **Impacto:** Podría retrasar mínimamente las fases iniciales del proyecto.
- **Plan de acción:** Dar prioridad a la comunicación con los diferentes miembros del grupo BDI para consultar cualquier tipo de duda y aclarar rápidamente el motivo que esté causando la parada en el avance.

3.3.3. Problemas ocasionados en el aprendizaje y correcta utilización de los distintos algoritmos de segmentación a analizar y utilizar

- **Probabilidad:** Medio.
- **Impacto:** Tendría un impacto grande en las fases iniciales del proyecto con lo que retrasaría y perjudicaría las fases avanzadas dependientes.
- **Plan de acción:** Podría plantearse la opción de desechar el algoritmo que pueda causar problemas o que pueda no ser comprendida su utilización en el ámbito del proyecto.

3.3.4. Problemas relacionados con las herramientas o tecnologías usadas en la página web

- **Probabilidad:** Bajo.
- **Impacto:** Podría tener un impacto en los tiempos de finalización del proyecto pero sin causar desviaciones excesivas.
- **Plan de acción:** Dado el margen de tiempo disponible en el proyecto, este riesgo no debería de requerir de ningún tipo de acción.

3.4. Herramientas y tecnologías

Durante el desarrollo del proyecto se utilizarán una serie de herramientas y tecnologías, que permiten llevar a cabo el contenido a elaborar. En este punto se presentarán dichas herramientas y tecnologías, explicando brevemente su función en el proyecto.

3.4.1. Google cloud

Google cloud[2] es una plataforma desarrollada por Google que ofrece un conjunto de productos y servicios relacionadas con el desarrollo tanto de la fase de análisis de algoritmos y segmentación como de la fase de implementación de la aplicación web. Ofrece un servicios del tipo alojamiento de máquinas virtuales (compute engine), alojamiento de aplicaciones web, almacenamiento de datos (storage)...

En este TFG, se utilizarán las funciones de Storage y Compute engine para llevar a cabo pruebas relacionadas con la segmentación de las series temporales, ejecutando dichas pruebas de manera automatizada utilizando una máquina virtual creada en la plataforma.

Será utilizada también, en otro proyecto independiente de Google Cloud, la función de Firestore junto con la ya mencionada Storage para la implementación de la Base de Datos de la plataforma web. En el apartado 7.2.1 se detalla el diseño y la utilización de dichas funcionalidades en el la web desarrollada.

3.4.2. MatLab

MATLAB[3] es sistema compuesto por un entorno de desarrollo y con un lenguaje de programación propio que permite realizar complejas operaciones matemáticas y realizar cálculos y simulaciones complejas. Es un software muy utilizado por ingenieros y matemáticos.

En este proyecto será utilizado para trabajar y realizar pruebas relacionadas con el Matrix Profile (ver apartado 4.1), algoritmo que está desarrollado de manera oficial en dicha plataforma.

[2]Google cloud: https://cloud.google.com
[3]Matlab: https://www.mathworks.com/products/matlab.html

3.4.3. Rstudio (R)

Rstudio[4] es un entorno de desarrollo que permite trabajar con el lenguaje de programación R[5]. R es un lenguaje de programación libre orientado a cálculos estadísticos y a gráficos.

Rstudio será uno de los elementos más utilizados en la parte central de las pruebas sobre la reducción de las series segmentadas. Esto se debe a que el grupo BDI tiene implementados los algoritmos de reducción de series temporales en R y por tanto serán ejecutadas utilizando este entorno.

3.4.4. React.js

React.js[6] es una librería de JavaScript que permite construir interfaces de usuario e incluso parte de la lógica de una plataforma web. Es una tecnología en auge debido al enfoque que tiene respecto a la integración de la interfaz y de la lógica de una página, desmarcándose cláramente de las implementaciones clásicas.

React ha sido la tecnología seleccionada para el desarrollo de la aplicación web y su funcionamiento y características se desarrollan en mayor profundidad en el apartado 7.2.2.

3.4.5. Visual Studio Code

Visual Studio Code[7] es un entorno de desarrollo software gratuito de Microsoft pensado para llevar a cabo desarrollos con diferentes propósitos y tecnologías. Su planteamiento está enfocado en poder utilizar un mismo editor para poder desarrollar en distintos lenguajes, ofreciendo todo el entorno necesario para ellos ya sea de forma nativa o a través de la instalación de librerías.

En este caso será utilizado para desarrollar la web del proyecto ya que ofrece integración con React de manera nativa, permitiendo un desarrollo cómodo e intuitivo.

[4]Rstudio: https://www.rstudio.com
[5]R: https://www.r-project.org
[6]React.js: https://es.reactjs.org
[7]Visual Studio Code: https://code.visualstudio.com

3.4.6. Git(GitHub)

Git[8] es un software enfocado al control de versiones, que permite registrar los cambios realizados en los archivos de un proyecto e identificar la autoría de dichos cambios. En este caso se utilizará junto con GitHub[9], una plataforma que ofrece un servicio de hosting para poder almacenar, visualizar y gestionar los proyectos creados con Git.

En este caso se utilizarán estas herramientas para almacenar los cambios que se vayan realizando en el proyecto y así mantener un control de las versiones de la web.

3.4.7. Overleaf(LaTeX)

LaTeX[10] es un sistema de composición de textos escritos que está diseñado para la producción de documentos técnicos y científicos. Se considera el estándar de facto en la comunicación y publicación de documetación científica. En este caso, dada su óptima estructuración y la facilidad que ofrece en la generación del documento se ha seleccionado para crear esta memoria.

Para la utilización de dicho sistema, se ha seleccionado la plataforma Overleaf[11]. Overleaf es una plataforma online de escritura y generación de textos en LaTeX, siendo a su vez orientada a la colaboración. Se ha seleccionado debido a la sencillez de su utilización y a la interfaz intuitiva que ofrece, pudiendo visualizar en una misma ventana tanto el código que se está escribiendo como el documento final que se va generando. Se tiene en cuenta también el hecho de que sea una plataforma web, ya que permite editar el trabajo desde cualquier dispositivo.

3.4.8. Smartsheet

Smartsheet[12] es una aplicación que ofrece opciones de colaboración y gestión de trabajos dentro de una empresa, permitiendo gestionar tareas, calendarios, obejtivos, documentos...

[8] Git: https://git-scm.com
[9] GitHub: https://github.com
[10] LaTeX: https://github.com
[11] Overleaf: https://es.overleaf.com/
[12] Smartsheet: https://es.smartsheet.com

En este caso no se ha utilizado esa herramienta principal como tal, si no que se ha utilizado una herramienta concreta que ofrece para la creación de diagramas Gantt, que ha sido utilizado para el apartado 3.2.2 de Periodos de realización de las tareas.

3.4.9. Draw.io

Draw.io[13] es una plataforma online gratuita que permite generar una gran cantidad de diagramas de manera sencilla. Permite elaborar diagramas de flujo, UML, diagramas de red, diagramas de procesos...

Entre sus funciones está la de poder generar una EDT, que es una de las funciones para la que lo hemos usado en el apartado 3.1.3, siendo otra la elaboración del diagrama de dependencias generado en la planificación en el apartado 3.2.1.

3.4.10. Django

Django[14] es un framework basado en Python que permite el desarrollo de aplicaciones web de una manera sencilla y rápida. En este caso se utilizará junto con el módulo de Django-rest-framework[15] para desarrollar un servicio web (API REST) de segmentación de series temporales, ya que los algoritmos de segmentación estarán desarrollados en Python y por tanto encajarán de manera perfecta con el código de Django.

3.4.11. Python

Python[16] es un lenguaje de programación multiparadigma con licencia de código abierto, es decir, que permite programación orientada a objetos, programación imperativa y programación funcional. Es un lenguaje creado con la premisa de ser simple de interpretar, muy legible y flexible. Como se ha dicho, este lenguaje será utilizado con Django para desarrollar el servicio web y también será utilizado, gracias a su potencia y polivalencia, durante las diversas pruebas que se realizarán en el proyecto.

[13] Draw.io: https://www.draw.io
[14] Django: https://www.djangoproject.com
[15] Módulo desarrollado para Django que permite generar servicios API REST de manera simple y efectiva.
[16] Python: https://www.python.org

3.4.12. CanvasJS

CanvasJS[17] es una librería de JavaScript que permite integrar en una página web distinta variedad de gráficos. Funciona con JavaScript y HTML5 y se ofrecen integraciones con distintas tecnologías, tanto frontend (React, Angular...) como backend (PHP, ASP.NET...).

En este caso será utilizado en la aplicación web para mostrar las representaciones gráficas de las series temporales con las que un usuario trabaje.

[17] CanvasJS: https://canvasjs.com/

4. CAPÍTULO

Análisis y elección de los algoritmos

Dentro del procesamiento de series temporales, la segmentación es un apartado muy interesante y desafiante para los ingenieros de datos, debido a las expectativas que genera, acompañado con la dificultad que conlleva su aplicación.

En este contexto existen varios algoritmos relacionados con la segmentación de series temporales que han sido objeto de análisis en distintas publicaciones científicas y que podrían ser más o menos útiles en el ámbito de trabajo del grupo BDI.

Aunque podamos encontrar distintos algoritmos de segmentación diferentes y cada uno de ellos con una solución distinta, todos ellos podrían clasificarse en 3 grupos distintos según su manera de afrontar el proceso de segmentado. Estos grupos son:

- **Sliding Windows:** Se selecciona un segmento y se va incrementando su tamaño hasta que excede algún tipo de límite de error indicado. Esto se va repitiendo hasta completar toda la serie. (ver figura 4.2)
- **Top-Down:** La serie temporal completa se va particionando hasta que se cumple un criterio de parada establecido. (ver figura 4.1)
- **Bottom-up:** Comenzando desde la segmentación más fina establecida, se van uniendo segmentos hasta que se cumple un cierto criterio de parada. (ver figura 4.3)

En el momento del inicio del proyecto era sabido que el grupo BDI conocía de la existencia del algoritmo del Matrix Profile y de la capacidad que tenía de, entre otras cosas, sementar series de datos. Partiendo de esa base, el primer paso a realizar consistía en una

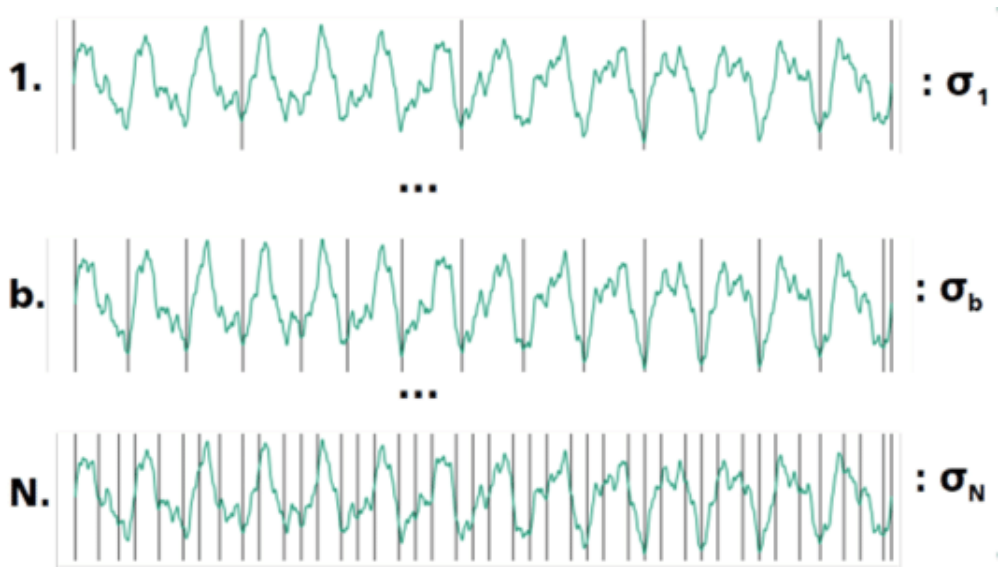


Figura 4.1: Ejemplo de utilización del Top-down en la que por cada paso la serie se va granulando en mayor medida.

búsqueda de las distintas aproximaciones a la segmentación publicadas, seleccionando las más prometedores para un posterior análisis de sus funcionalidades y aplicabilidad.

Se llevó a cabo una primera búsqueda en la que se seleccionaron distintas publicaciones que de un primer vistazo podrían estar relacionadas con la segmentación de series. Dicha lista fue analizada y se fueron seleccionando los casos en los que la idea planteada tenía algún tipo de implementación presentada como aporte a las explicaciones teóricas.

Finalmente, los algoritmos seleccionados para llevar a cabo las posteriores pruebas e implementaciones han sido los algoritmos de Matrix profile, Simple segment y el Greedy Gaussian Segmentation. A continuación se introducen los distintos algoritmos de segmentación mencionados, presentando sus principales características así como un análisis de las pruebas realizadas y las conclusiones obtenidas de las mismas:

4.1. Matrix profile

El Matrix profile[1] es una algoritmo de tratamiento de series temporales desarrollado por la *Keogh research group de la Universidad de California-Riverside*. Es una de las herramientas más notorias en el mundo de la Industria 4.0 para el tratamiento de series

[1]Matrix profile: https://www.cs.ucr.edu/~eamonn/MatrixProfile.html

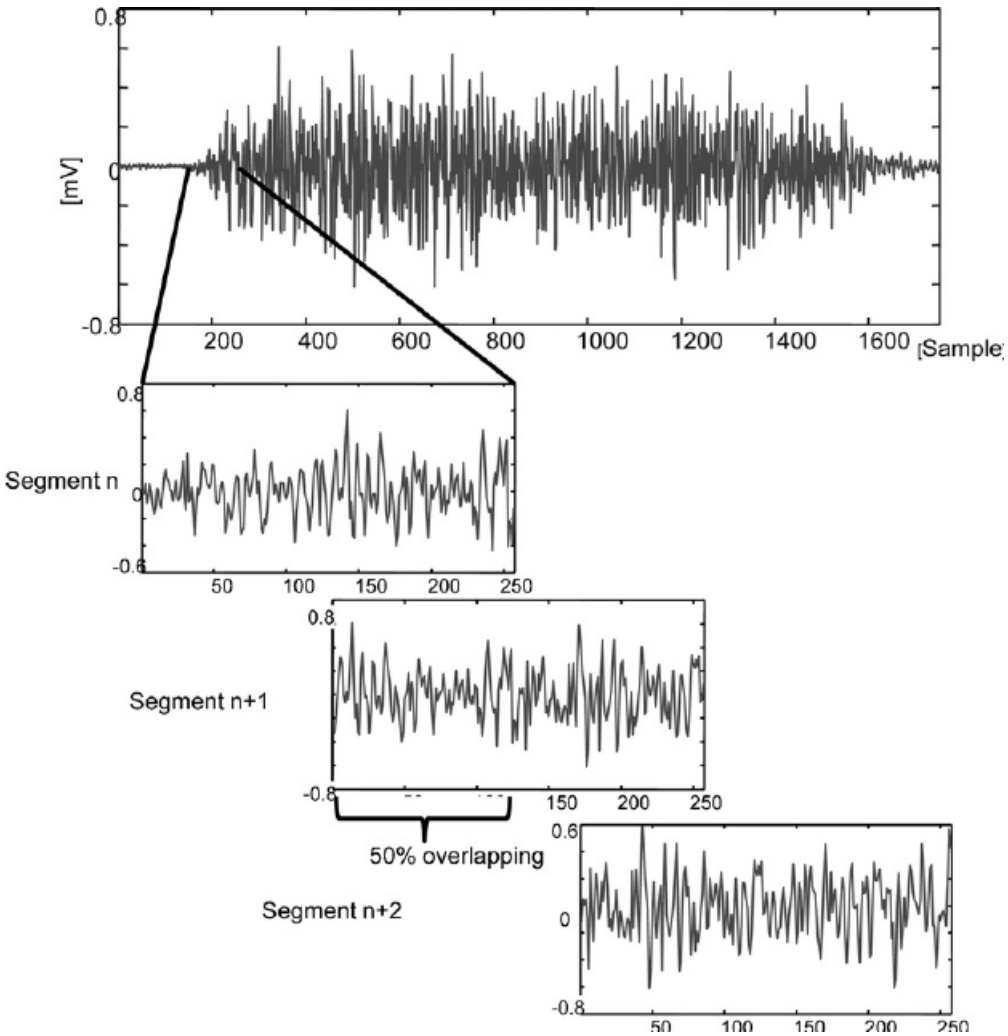


Figura 4.2: Ejemplo de algoritmo basado en el Sliding window, en este caso solapando las ventanas adyacentes.

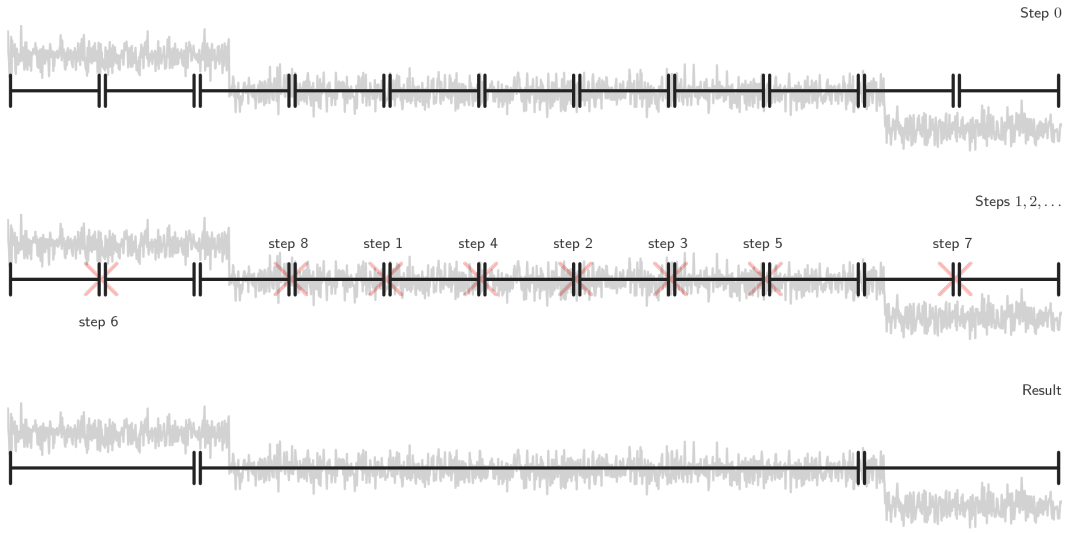


Figura 4.3: Ejemplo de utilización del Bottom-up para la segmentación en el que se muestra la eliminación de puntos de corte por cada paso realizado.

temporales dada su potencia y su polivalencia. Permite obtener una cierta información de las series que posteriormente puede ser utilizada para infinidad de funciones, tales como la segmentación, comparación, detección de diferencias...

Cabe destacar, que al no ser un algoritmo de segmentación propiamente dicho, no se podría clasificar en ninguna de las 3 categorías que se han mencionado al comienzo del capítulo. El Matrix profile no deja de ser un algoritmo que saca una información genérica relacionada con la serie temporal de la que posteriormente se puede sacar la información relativa a la segmentación, pero que no tiene nada que ver con las aproximaciones tradicionales a la segmentación.

El algoritmo permite obtener, por cada subsecuencia de la serie (de tamaño previamente indicado), qué otra subsecuencia es la que más se le parece y la posición que ocupa dentro de la serie. Estos datos son plasmados en una nueva serie temporal, justamente la que es llamada Matrix Profile (ver figura 4.4), en la que cada elemento de la serie corresponde a la distancia al *Vecino más próximo [Cover and Hart, 2018]*.

Una vez obtenido el Matrix Profile correspondiente a la serie temporal, se abren diferentes ramas de acción, en las que podemos trabajar, tales como la detección de diferencias entre series, segmentación o detección de anomalías. En este caso nos centraremos en la que a nosotros nos interesa: la segmentación.

Centrándonos ya en la segmentación, el grupo Keogh presenta una manera de afrontarla,

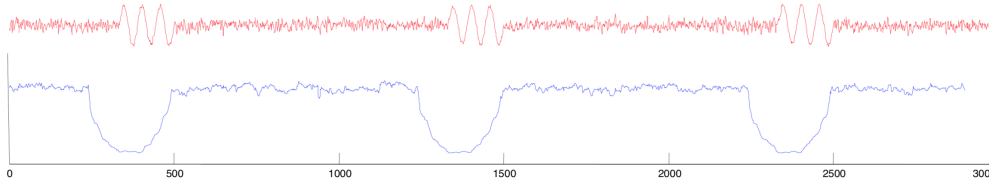


Figura 4.4: Ejemplo de serie temporal (arriba) junto con su representación con el Matrix profile (debajo).

utilizando el Matrix profile, en la publicación [Gharghabi et al., 2017]. Tal y como se comenta en la publicación, se introduce un nuevo concepto llamado CAC (*Corrected Arc Crossings*), que es el que hay que obtener del Matrix Profile para poder conseguir los puntos a segmentar.

El algoritmo que presentan algoritmo se basa en el resultado obtenido en el Matrix Profile para detectar cambios en la serie y poder así detectar puntos de segmentado a través de la obtención del CAC. Explicándolo de manera resumida, este algoritmo va recorriendo todos los puntos del Matrix Profile y va trazando un arco desde cada punto hasta el punto indicado en el Matrix Profile index (segmento más similar al de ese punto), tal y como se puede ver en la figura 4.5.

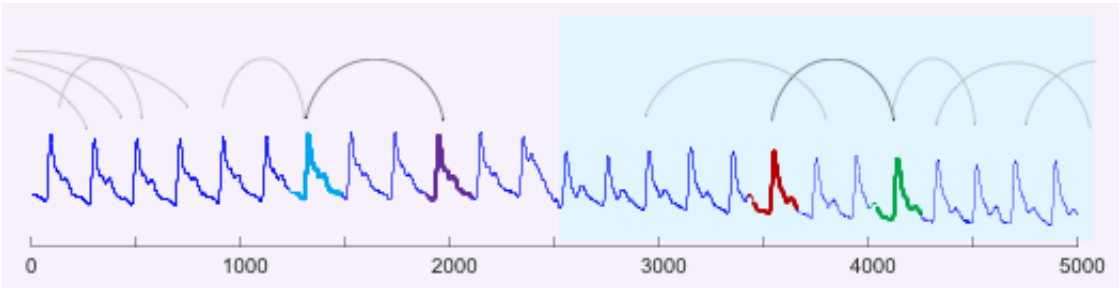


Figura 4.5: Representación de los arcos que se van trazando durante la ejecución del algoritmo CAC.

Después de trazar los arcos, va sumando por cada uno de los puntos de la serie el número de arcos que pasan por encima del mismo, generando una nueva serie donde cada punto es el número de arcos que pasan por encima del punto correspondiente del Matrix Profile, todos ellos reducidos a un número entre 0 y 1. Finalmente, teniendo en cuenta que los puntos cercanos al comienzo y al final de la serie van a tener menos arcos que pasen por

encima, se selecciona una subsecuencia determinada al comienzo y al final de la serie del CAC y se le da un valor de 1 a cada punto para que no afecte al resultado final de la representación. Esta corrección final se puede ver reflejada en la figura 4.6, en la que se muestra tanto el resultado del CAC sin corrección (gráfico superior) como el resultado con la corrección en los extremos (gráfico inferior).

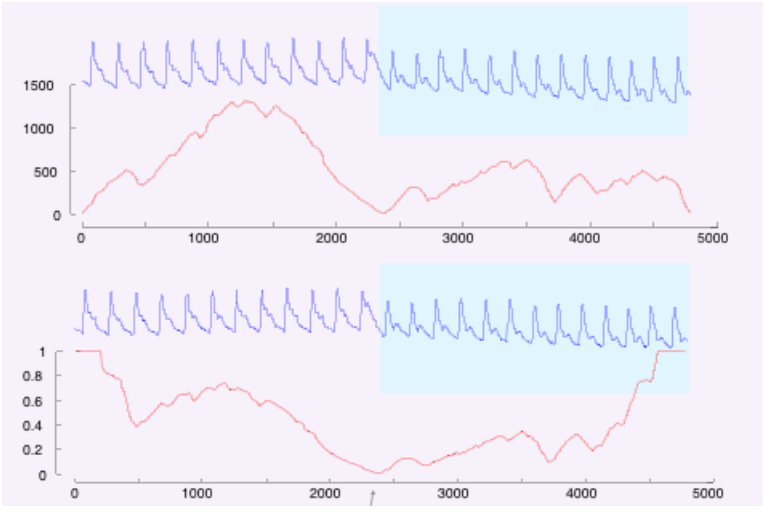


Figura 4.6: Representación de los arcos que se van trazando durante la ejecución del algoritmo CAC.

Una vez obtenida esa serie resultante del CAC, el grupo de Keogh presenta un último algoritmo que detecta los N mínimos locales que se le indiquen, cogiendo el mínimo de la serie y excluyendo la zona cercana al mismo. Así, una vez aplicado este algoritmo final, obtenemos los N puntos donde finalmente la serie tendría que ser segmentada.

Dicho esto, y para facilitar las menciones a estos algoritmos utilizados para segmentar series, a partir de ahora nos referiremos como Matrix Profile a todo este proceso.

Este algoritmo viene implementado y ofrecido por el propio grupo Keogh en una versión de MatLab, que ha sido en este caso la utilizada para el desarrollo de las pruebas relacionadas a la segmentación. Los algoritmos para la obtención del CAC y la búsqueda de mínimos en el CAC, en cambio, no estaban implementados de manera oficial, habiendo sido implementados para el desarrollo de este proyecto siguiendo el pseudocódigo presentado en la publicación del algoritmo.

En adición al código de MatLab, el algoritmo del Matrix profile se encuentra también desarrollado en código Python. Esta variante ha sido posteriormente seleccionada para la implementación del algoritmo en el servicio web de segmentación desarrollado en el

proyecto, junto con los otros dos algoritmos necesarios, que en este caso también han sido implementados a partir del pseudocódigo ofrecido.

4.2. Simple segment

El Simple segment es una implementación de variantes de distintos algoritmos de segmentación ya conocidos en el ámbito, como son los ya mencionados Sliding Windows, Bottom-Up y Top-Down. En este caso ofrece una variante de cada uno de ellos combinándolos tanto con la Interpolación lineal[2] como con la Regresión lineal[3].

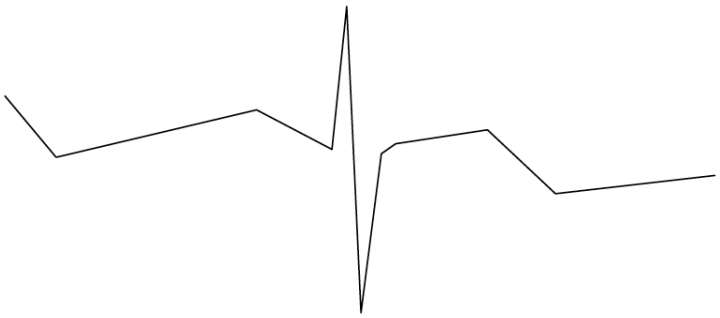


Figura 4.7: Ejemplo de aplicación de la Interpolación lineal.

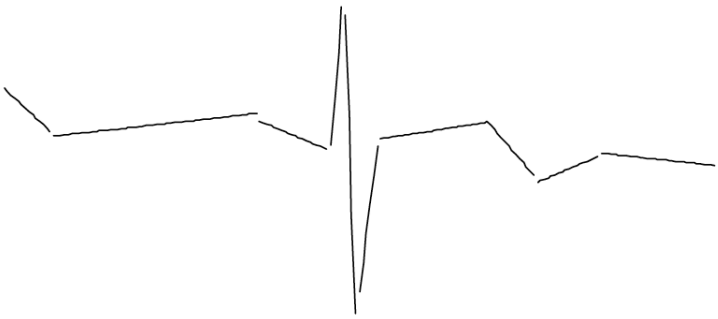


Figura 4.8: Ejemplo de aplicación de la Regresión lineal.

En este caso concreto, después de unas pruebas iniciales, se llegó a la conclusión de que el que tenía mayor potencial para su utilización futura era el de la variante del Sliding

[2]Interpolación lineal: Se traza una línea que conecta el inicio de una subsecuencia con el final de la misma. Ver Fig. 4.7

[3]Regresión lineal: Se traza una línea que corresponda a la recta de regresión de dicha subsecuencia de puntos, es decir, que más se adecue a dichos puntos de acuerdo al criterio del mínimo error cuadrático[]. Ver Fig. 4.8

Window con Interpolación lineal. Por tanto, esa ha sido la opción seleccionada y será el algoritmo al que nos refiramos cuando mencionemos el Simple segment.

Tal y como hemos dicho, como el algoritmo se basa en el Sliding Window, lo que hace es ir generando desde el inicio de la serie una subsecuencia que se va aumentando hasta que llega a un determinado error, donde se para ese aumento y se traza una línea utilizando la interpolación lineal. Esto se va repitiendo cogiendo como siguiente punto el final de la subsecuencia generada anteriormente. Respecto al parámetro del error que hemos mencionado, tal y como hemos dicho actúa como criterio de parada del crecimiento de la ventana, siendo este parámetro introducido por el usuario a la hora de ejecutar el algoritmo. Una vez el proceso termina, el algoritmo devuelve, entre otras cosas, los puntos de inicio y final de las subsecuencias generadas, que nos indican los puntos de corte de la segmentación a realizar.

Respecto a la utilización de la interpolación lineal o de la regresión lineal, en este caso es irrelevante la opción que se utilice a la hora de llevar a cabo las pruebas y las posteriores implementaciones. Esto es debido a que, en este contexto en el que va a ser utilizado este Simple segment, lo interesante son los puntos de corte de la serie temporal procesada y no tanto las rectas que genere entre dichos puntos. Dicho esto, en este caso se ha trabajado con la opción de interpolación lineal por un mero hecho estético, ya que con esta opción la visualización gráfica de los resultados obtenidos y la identificación visual de los puntos de corte generados es mayor.

A diferencia del Matrix profile, este algoritmo se encuentra implementado solamente en Python, versión que facilita tanto su uso en las diferentes pruebas como su posterior implementación en el servicio web de segmentación.

4.3. Greedy Gaussian segmentation

El Greedy Gaussian Segmentation of Multivariate Time Series (GGS a partir de ahora) es un algoritmo desarrollado por David Hallac, Peter Nystrup y Stephen Boyd, presentado en una publicación en Abril de 2018[Hallac et al., 2019] en la que muestra una posible solución enfocada a la segmentación de series temporales de datos.

El algoritmo GGS está basado en un método heurístico que resuelve de manera aproximada un problema de optimización, buscando maximizar el resultado de la función definida en la figura 4.9. El algoritmo no garantiza encontrar la solución óptima respecto al número de puntos de corte o *breakpoints* que hacen óptima la segmentación, pero es capaz de

obtener una solución muy cercana a la misma y garantiza que para el número de *breakpoints* encontrado la solución propuesta es la óptima. Es decir, si la solución óptima es la de segmentar la serie por 10 *breakpoints*, el algoritmo puede que devuelva 9 *breakpoints*, pero que dichos 9 *breakpoints* serán los óptimos respecto a todas las segmentaciones con 9 *breakpoints* posibles.

El algoritmo calcula una solución aproximada a la función a maximizar 4.9 de manera iterativa, añadiendo por cada iteración un nuevo punto de corte y ajustando posteriormente todos los *breakpoints* ya existentes buscando maximizar el objetivo final. El algoritmo itera como mucho hasta llegar al número de *breakpoints* máximo (indicado como parámetro) y en cada iteración realiza los siguientes pasos:

$$\text{maximize} \quad -\tfrac{1}{2}\sum_{i=1}^{K+1}\left((b_i - b_{i-1})\log\det(S^{(i)} + \tfrac{\lambda}{b_i - b_{i-1}}I) - \lambda\,\mathbf{Tr}(S^{(i)} + \tfrac{\lambda}{b_i - b_{i-1}}I)^{-1}\right)$$

Figura 4.9: Función a maximizar en busca de la respuesta optimizada en la ejecución del algoritmo GGS.

1. Por cada segmento existente el algoritmo define un nuevo *breakpoint* que maximice la función objetivo en dicho segmento.
2. De los *breakpoints* fijados se comprueba cual de ellos permite obtener el resultado más óptimo sobre la serie completa.
3. Si con todos los *breakpoints* la optimización ha empeorado se termina de añadir *breakpoints* y se devuelven los *breakpoints* actuales. Si esto ocurre en en la primera iteración el algoritmo no devolverá ningún *breakpoint* para la serie.
4. Si la optimización no ha empeorado con todos los *breakpoints*, se selecciona el *breakpoint* que optimice en mayor medida (seleccionado en el punto 2) y se añade como nuevo *breakpoint* de la serie.
5. Se ajustan todos los *breakpoints* ya existentes para encontrar la solución óptima con el número de *breakpoints* actual en esta iteración.

Como se puede deducir de dicho proceso, este algoritmo estaría englobado dentro de la categoría de algoritmos con enfoque Top-down, ya que parte de la serie completa y va añadiendo distintos *breakpoints* generando cada vez una segmentación con más *breakpoints*.

Este algoritmo se encuentra implementado en Python, al igual que en el caso de Simple segment, por lo que el código ha sido directamente utilizado tanto en el análisis de los

algoritmos como en la implementación correspondiente en el servicio web de segmentación.

4.4. Pruebas

En esta sección se presentan algunos ejemplos de las pruebas realizadas en el análisis de los distintos algoritmos de segmentación seleccionados. Se pretenden reflejar los principales ejemplos de series que han sido analizadas, que caracterizan el funcionamiento y resultados obtenidos con cada uno de los algoritmos.

Tal y como se ha mencionado en el apartado 2.2.1, los datos con los que se han realizado las diferentes pruebas han sido tanto los proporcionados por el grupo BDI como los obtenidos del *UCR Archive*, permitiendo comparar el funcionamiento de los algoritmos en distintos dominios de aplicación.

4.4.1. Pruebas con los distintos tipos de series del grupo BDI

La idea inicial de las pruebas era la de comprobar si alguno de los algoritmos era capaz de segmentar una serie que contenía un numero repetido de patrones claros, diferenciando dichos patrones para que a cada tipo de patrón se le pudiese aplicar una técnica distinta.

Por ejemplo, se puede comprobar la prueba realizada con la serie que se muestra en la figura 4.10, en la que las líneas rojas representan la serie temporal y las líneas discontinuas azules los puntos de corte que se consideran ideales y que se pretendían conseguir con los algoritmos.

En dicha serie los tramos que tienen una forma similar se identifican fácilmente y la idea está en poder conseguir una segmentación que sea capaz de aislarlos. Los resultados obtenidos con los distintos algoritmos han sido los siguientes:

Matrix profile

Para la ejecución del algoritmo se parte con dudas del rango en el que se moverá el parámetro que pueda arrojar resultados correctos. Se ha probado con los valores 10, 100, 1000 y 10000 pero el resultado es totalmente incorrecto para todos ellos, tal y como se

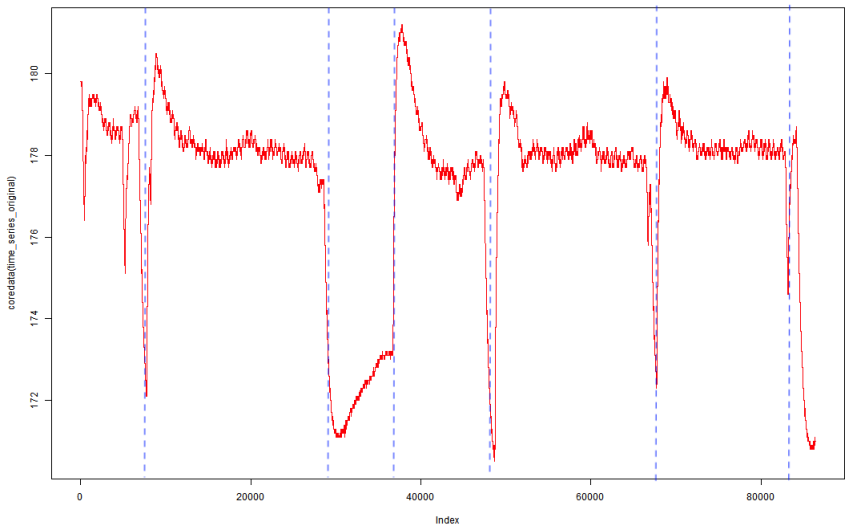
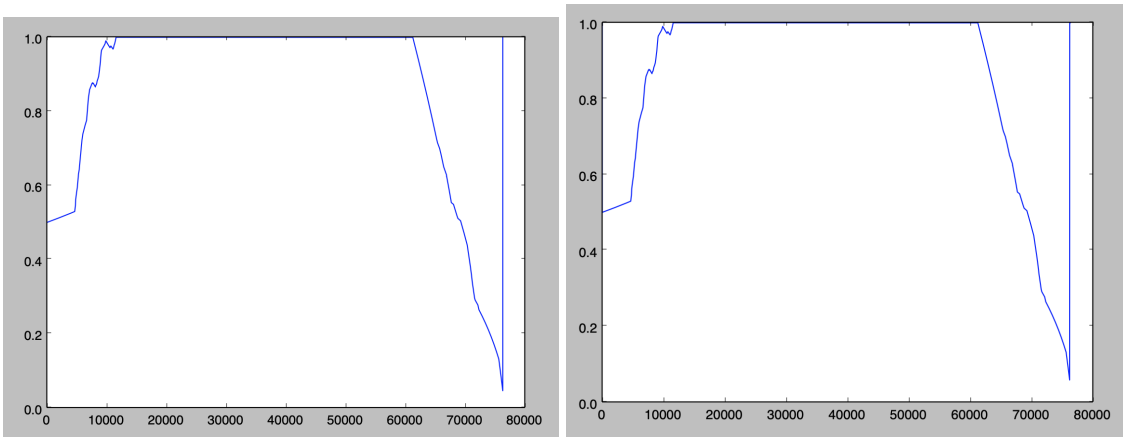


Figura 4.10: Serie temporal utilizada en una de las pruebas de detección de patrones.

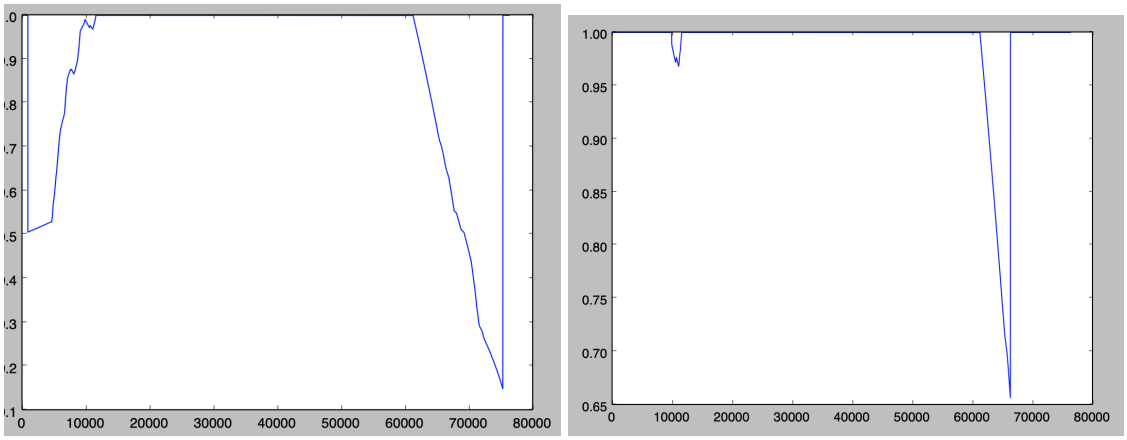
puede ver en la figuras 4.11 y 4.12, en las que debemos identificar los posibles puntos de corte fijándonos en los mínimos que vemos en las gráficas presentadas en dichas figuras.



(a) Parámetro 10 **(b)** Parámetro 100

Figura 4.11: Resultados de la aplicación del Matrix profile a una serie con patrones repetidos.

Vemos incluso como, a medida que vamos modificando el algoritmo, en muchos casos los resultados son prácticamente idénticos entre ellos, por lo que no parece que se vaya a poder llegar a la solución deseada. Esto nos lleva a pensar si el Matrix profile podría llegar a ser capaz de segmentar esta serie por los puntos indicados ya que, si no lo hace, no sería útil en la gran mayoría de casos.



(a) Parámetro 1000 **(b)** Parámetro 10000

Figura 4.12: Resultados de la aplicación del Matrix profile a una serie con patrones repetidos.

Se han realizado ciertas pruebas con otras combinaciones del parámetro de entrada pero el resultado final en ningún caso estaba cerca del óptimo buscado, quedando todas las pruebas en un rango de puntos obtenidos parecido al de las pruebas mostradas.

Simple segment

Para el parámetro del Simple segment ocurría algo parecido al del Matrix profile, y es que no se sabía qué rango del valor sería el adecuado para este caso. Se probó con el parámetro 1000, obteniendo los resultados reflejados en la figura 4.13 (cada recta en color azul representa un segmento). Como se puede observar, los resultados no estám muy alejados de los objetivo, pero quizás el error es demasiado pequeño y desgrana en demasía la serie.

Por esto, se probó a aumentar el parámetro a 10000 y se obtuvieron los resultados de la figura 4.14. En este caso vemos que los resultados obtenidos son mucho más cercamos a los planteados inicialmente, con lo que se puede deducir que el parámetro óptimo en este caso estará más cerca del 10000.

Se probaron ciertos valores distintos del parámetro y se encontró que con el valor 8000 los resultados eran los mostrados en la figura 4.15. Se puede ver como, a diferencia de con el parámetro 10000, se ha conseguido precisar en el corte y detección de los patrones

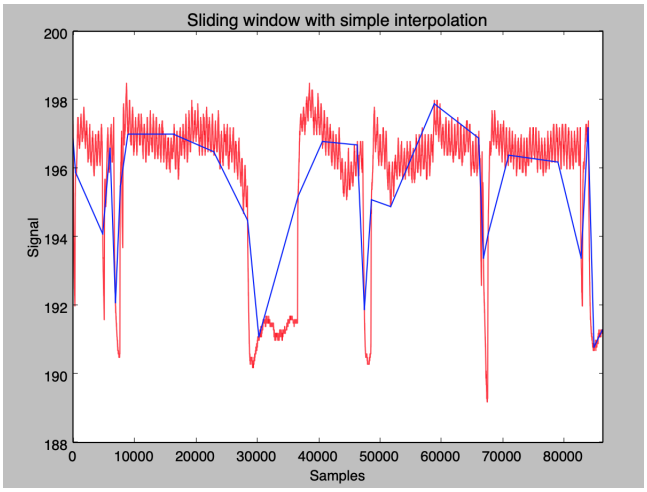


Figura 4.13: Serie temporal con patrones repetidos segmentada utilizando el algoritmo Simple segment con parámetro 1000.

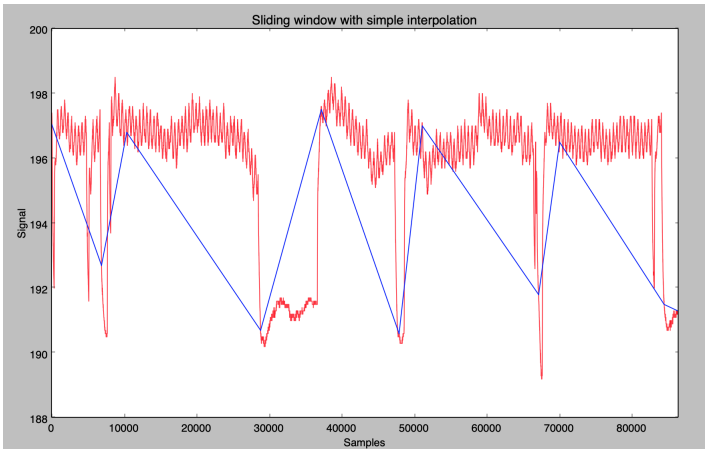


Figura 4.14: Serie temporal con patrones repetidos segmentada utilizando el algoritmo Simple segment con parámetro 10000.

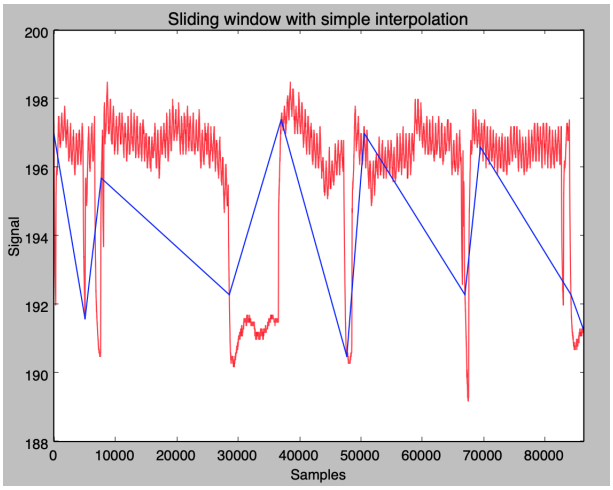


Figura 4.15: Serie temporal con patrones repetidos segmentada utilizando el algoritmo Simple segment con parámetro 8000.

repetidos de la serie, siendo casi idéntica a la planteada previo al segmentado.

GGS

Para la ejecución de este algoritmo le indicamos de parámetro el 6, para que nos encuentre como máximo ese número de cortes. El resultado del algoritmo con ese parámetro es el que se ve reflejado en la figura 4.16.

En este caso el resultado es extremadamente cercano a la segmentación previa planteada como la objetivo. Se pueden ver grandes coincidencias en los puntos de corte excepto en uno de ellos. Teniendo en cuenta que el algoritmo ha devuelto el resultado más óptimo posible utilizando 6 puntos de corte es posible que la solución que propone sea mejor que la que se había planteado previa a las pruebas, lo que habría que comprobar aplicando una posterior reducción de los segmentos.

Otro de los casos a analizar era la de detectar un cambio de patrón único en la serie y que el algoritmo fuese capaz de segmentar la serie en dos, detectando el punto de corte en el lugar en el que ocurre dicho cambio. En este caso se presenta la prueba realizada con la serie temporal que se presenta en la figura 4.17.

La idea era la de comprobar si los algoritmos tenían la capacidad de detectar y segmentar la serie desde dicho punto:

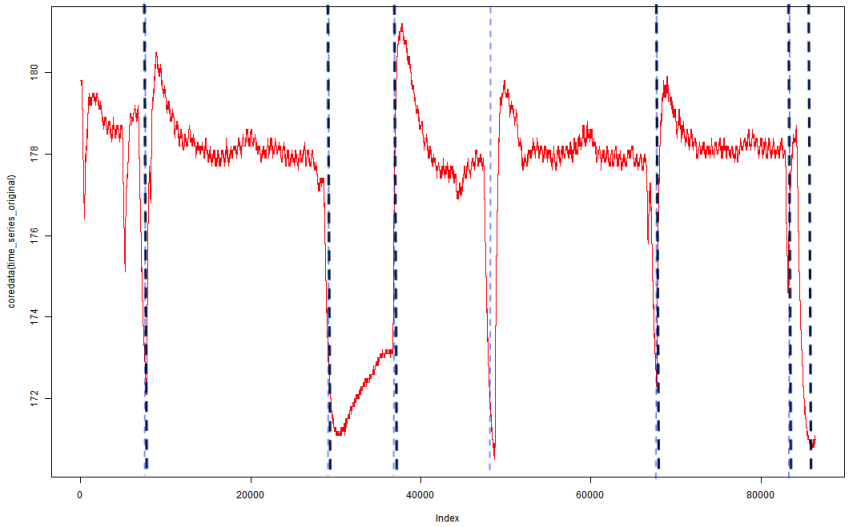


Figura 4.16: Serie temporal con patrones repetidos segmentada utilizando el algoritmo GGS.

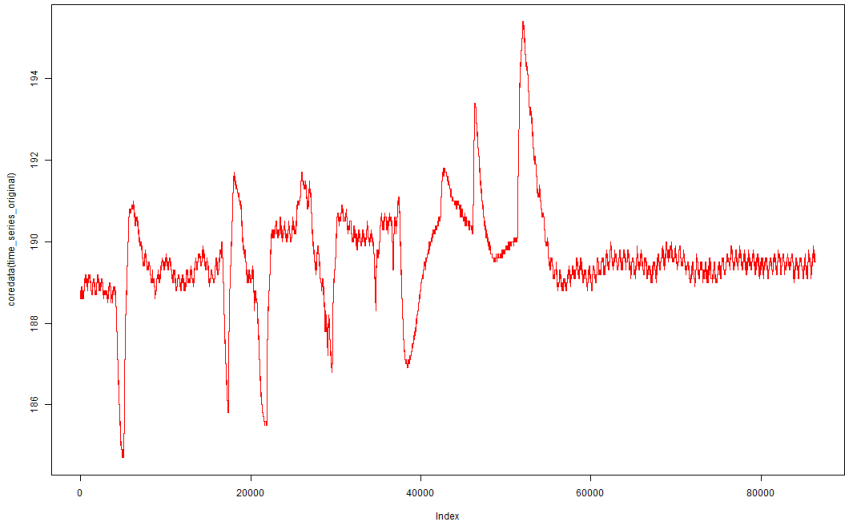


Figura 4.17: Serie temporal utilizada como prueba de la capacidad de los algoritmos de segmentación de detectar un único cambio claro de patrón.

Matrix profile

La ejecución en este caso ha sido realizada utilizando el parámetro 100000, obteniendo este resultado del CAC, visible en la figura 4.18. Como vemos, en este caso el resultado no está del todo desviado de la idea inicial, pero está claro que no es para nada exacto al deseado.

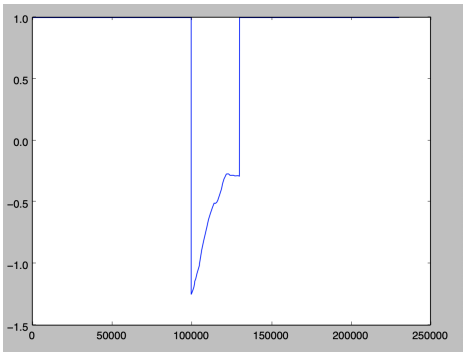


Figura 4.18: Resultado de la aplicación del Matrix profile para la segmentación de la serie con un solo cambio de patrón.

Cabe destacar, en este caso, el tiempo de ejecución del algoritmo debido a la cantidad de tiempo que ha requerido para terminar la ejecución. En este caso ha tardado muchos minutos en completarse la ejecución, por los pocos segundos que ha tardado en completarse el GGS.

Para tratar de ver si se podía afinar más el resultado y, sabiendo de que el Matrix profile podía obtener muy buenos resultados en este tipo de series, se probaron con otras combinaciones de parámetros, hasta probar con el parámetro 10000. Con este parámetro los resultados fueron óptimos, y similares a los obtenidos con los otros dos algoritmos, como se puede ver en la figura 4.19.

Simple segment

La primera prueba realizada con el Simple segment ha sido con el parámetro 10000, obte-

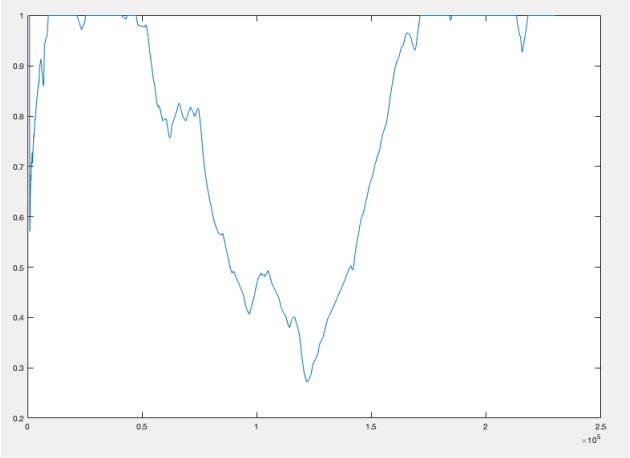


Figura 4.19: Resultado de la aplicación del Matrix profile para la segmentación de la serie con un solo cambio de patrón con el parámetro 10000.

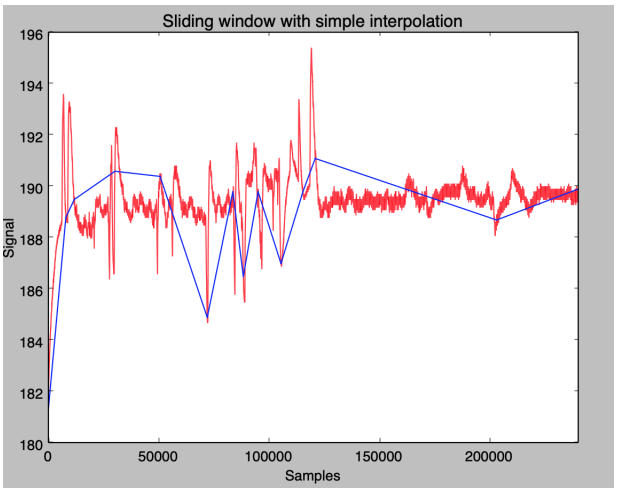


Figura 4.20: Resultado de la aplicación del Simple segment para la segmentación de la serie con un solo cambio de patrón, con el parámetro 10000.

niendo los resultados reflejados en la figura 4.20. Con ese parámetro la serie se segmenta en demasiados segmentos y no es capaz de diferenciar ese cambio claro de patrón.

Posteriormente se ha probado subiendo el parámetro hasta 100000, obteniendo los resultados mostrados en la figura 4.21. En este caso se ha conseguido un resultado mejor en comparación con el obtenido utilizando el parámetro 100000, pero sin estar aún cerca del resultado ideal buscado.

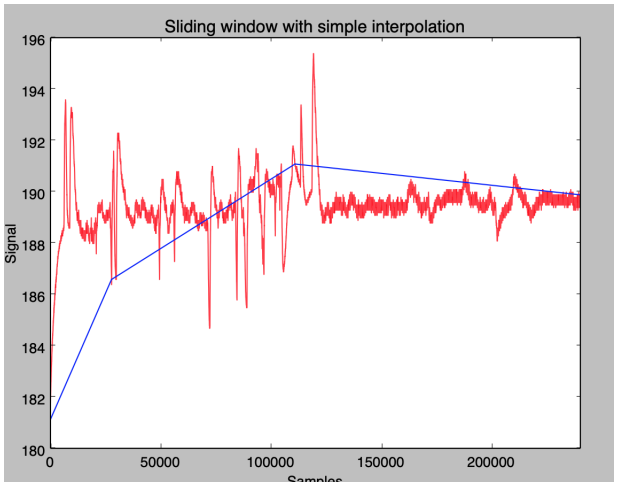


Figura 4.21: Resultado de la aplicación del Simple segment para la segmentación de la serie con un solo cambio de patrón, con el parámetro 100000.

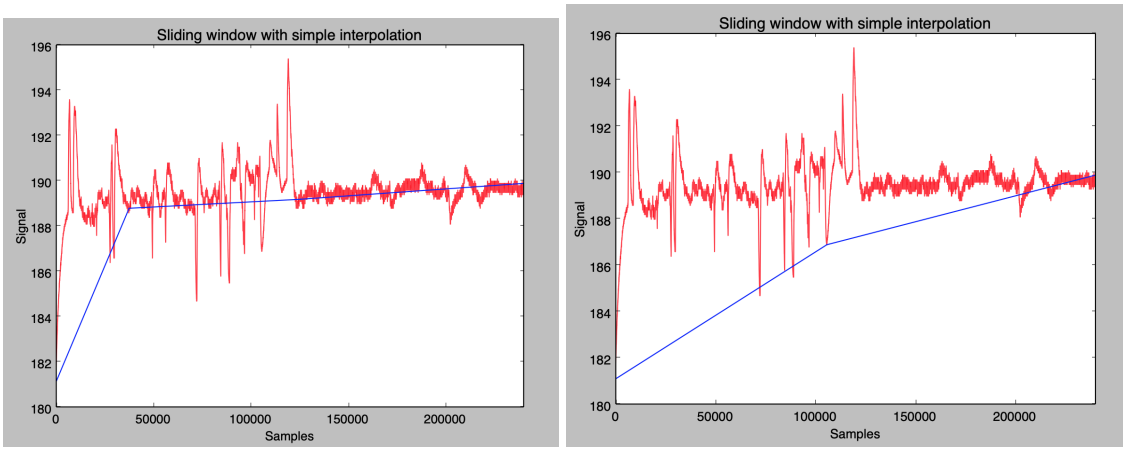
Se ha seguido aumentando el parámetro buscando la solución inicial buscada, siendo los resultados cada vez más cercanos a la misma, tal y como se observa en la figura 4.22, en la que se muestran los resultados obtenidos con 120000 y 200000.

Finalmente, tal y como se muestra en la figura 4.23, con el parámetro 250000 se logró un resultado de corte exacto respecto a la intención inicial, obteniendo dos segmentos finales diferenciando ambas morfologías de la serie.

GGS

En este caso lo único que hay que hacer es ejecutar el algoritmo indicándole como parámetro el 1, para que encuentre un solo punto de corte y segmente la serie temporal en dos. El resultado de la ejecución se muestra en la figura 4.24.

Tal y como podemos ver, en este caso el algoritmo también ha sido capaz de detectar el



(a) Parámetro 120000 **(b)** Parámetro 200000

Figura 4.22: Resultados de la aplicación del Simple segment con parámetros 120000 y 200000.

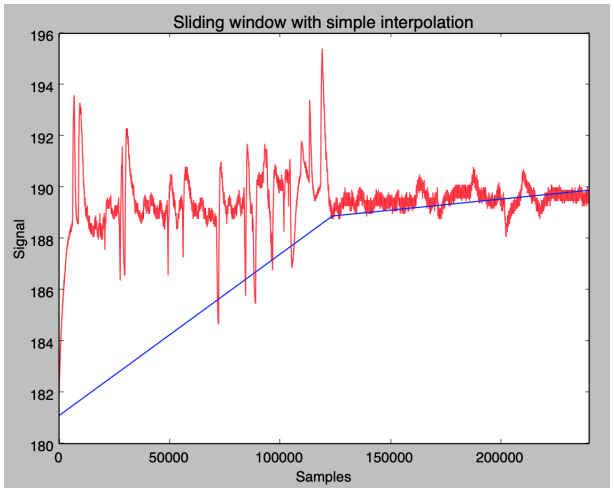


Figura 4.23: Resultado de la aplicación del Simple segment para la segmentación de la serie con un solo cambio de patrón, con el parámetro 250000.

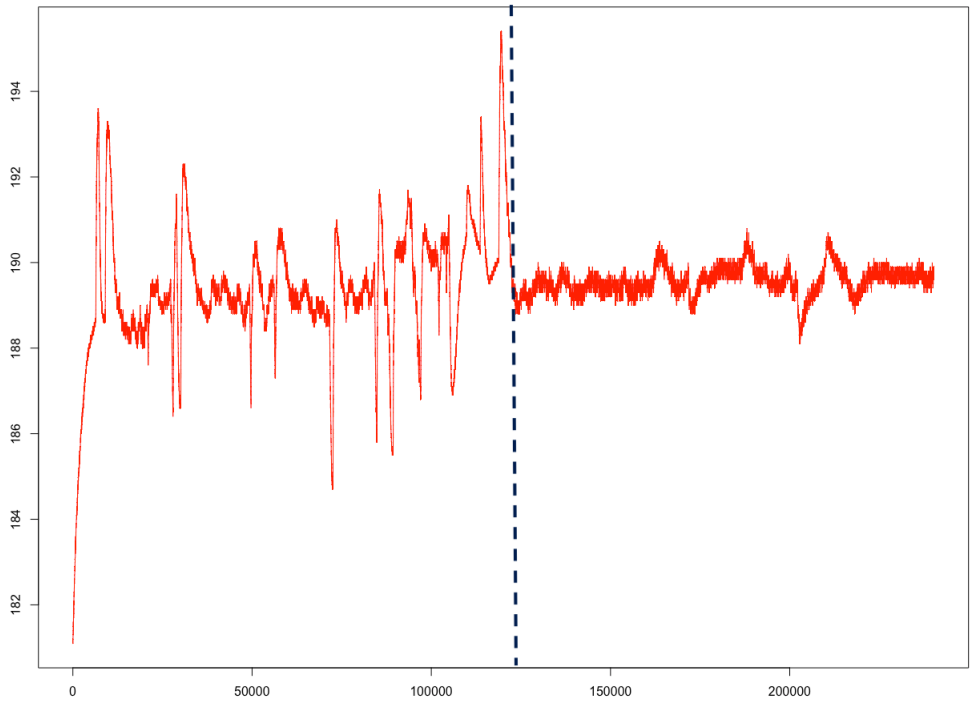


Figura 4.24: Serie temporal con patrón de diferenciación claro segmentado con el algoritmo del GGS.

punto de corte correcto, indicando el punto en el que claramente es visible el cambio de morfología de la serie.

Finalmente, otra de los tipos de series a comprobar era el de las series en las que no se identificaba ningún tipo de patrón claro ni repetitivo. Como ejemplo de ello, se presenta la serie de la figura 4.25. En este caso en la figura se presenta una posible segmentación que tendría sentido y se presentan las distintas pruebas realizadas con los tres algoritmos:

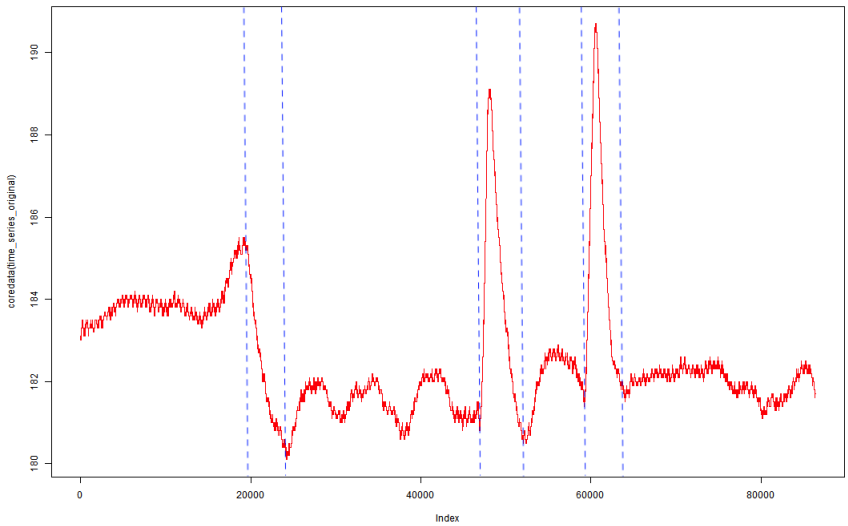


Figura 4.25: Serie temporal utilizada como prueba de la capacidad de los algoritmos de segmentación de detectar segmentos que no siguen ningún tipo de patrón claro.

Matrix profile

En este caso, se ha probado la segmentación con varios parámetros del algoritmo, llegando a la conclusión de que el valor que más acercaba la solución a la planteada inicialmente era el de 100. En la figura 4.26 se muestran los puntos por donde el algoritmo ha recomendado segmentar.

Se puede ver que la segmentación no difiere en exceso con la ofrecida, por ejemplo, por el algoritmo GGS. A pesar de ello, ha habido que dar con el parámetro de entrada correcto

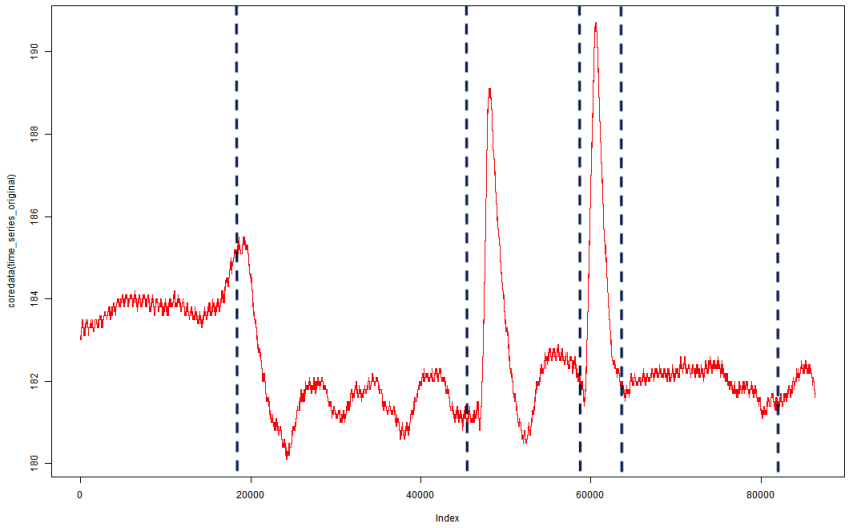


Figura 4.26: Resultado de la segmentación propuesta por el Matrix profile en una serie sin patrones claros.

para obtener este resultado, algo que en líneas generales se ha visto que es necesario para obtener resultados cercanos a lo buscado con este algoritmo.

Simple segment

En este caso se ha partido probando con el parámetro 1000 con el que, tal y como se ve en la figura 4.27, se obtienen unos resultados no muy lejanos a lo planteado como óptimo.

En este caso se ha tratado de ajustar el parámetro con distintos valores, en los que se ha llegado a la conclusión, por los resultados que se pueden ver en las figuras 4.28 (parámetro 8000) y 4.29 (parámetro 10000), de que moviéndonos entre esos valores los puntos de corte son muy similares y que con el Simple segment no se podría conseguir exactamente la segmentación deseada, aunque sí una muy similar.

GGS

En la figura 4.30 se presenta el resultado de la segmentación de dicha serie con el GGS, habiendo puesto como parámetro el 6, dado que eran los puntos de corte que podrian

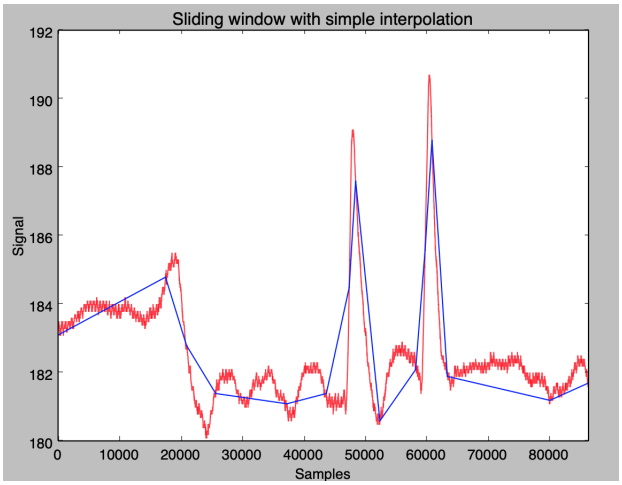


Figura 4.27: Resultado de la segmentación propuesta por el Simple segment en una serie sin patrones claros, parámetro 1000.

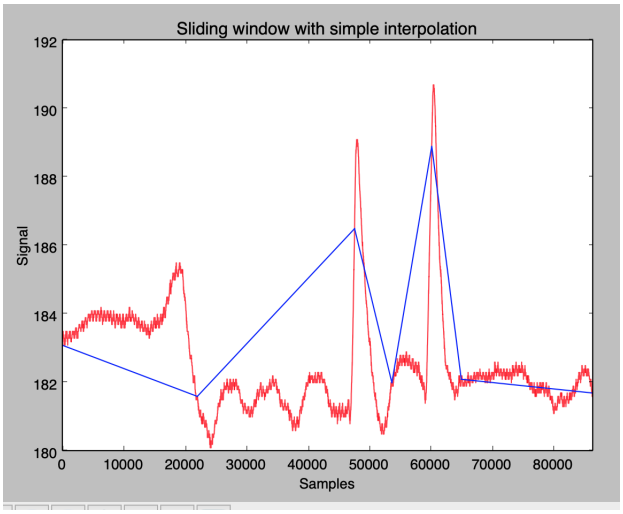


Figura 4.28: Resultado de la segmentación propuesta por el Simple segment en una serie sin patrones claros, parámetro 8000.

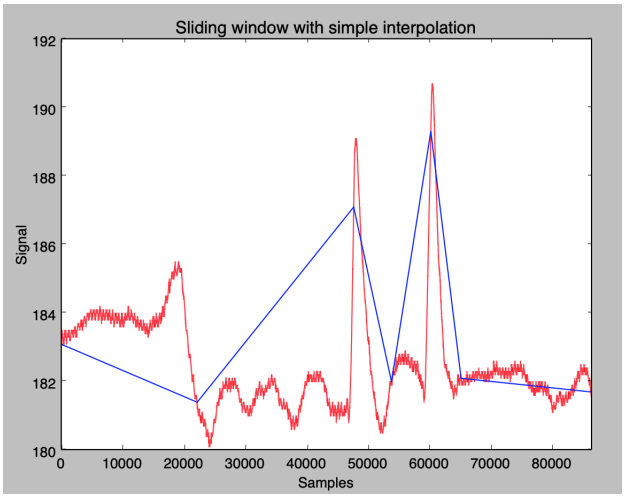


Figura 4.29: Resultado de la segmentación propuesta por el Simple segment en una serie sin patrones claros, parámetro 10000.

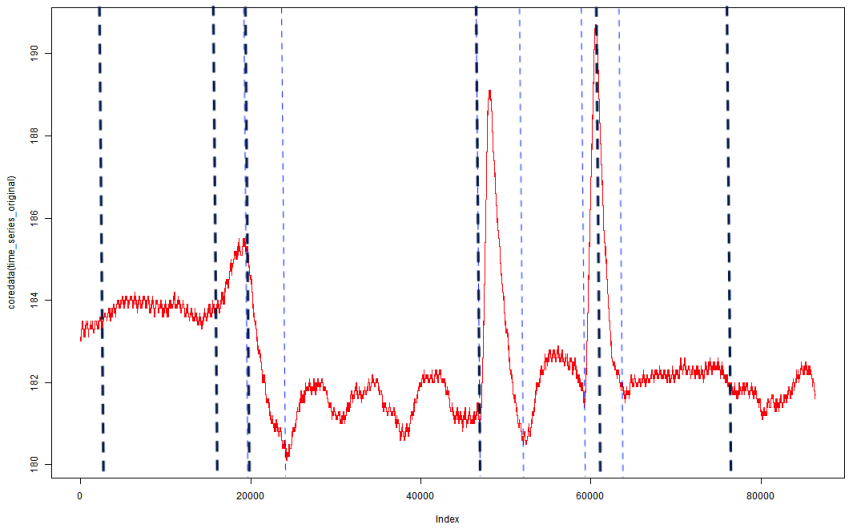


Figura 4.30: Resultado de aplicar el algoritmo GGS a una serie más caótica.

considerarse interesantes.

En este caso vemos como lo cortes no han sido tan precisos como si había ocurrido en los otros dos tipos de series. De todas maneras, los cortes planteados previo a la segmentación eran meramente intuitivos y puede ser que en realidad no sean los cortes óptimos. Hay que tener en cuenta además que el GGS, de todas las opciones que tiene en este caso de segmentar la serie con 6 puntos de corte, ha devuelto la óptima. Por tanto, casos como este dependerían ya de la posible reducción que se pudiese hacer posteriormente o del contexto de utilización de la segmentación.

4.4.2. Pruebas comparativas con alguna serie del *UCR Archive*

En adición a todas las pruebas realizadas con los tres algoritmos sobre distintas series con las que el grupo BDI trabaja, se han realizado pruebas sobre distintas series sacadas del *UCR Archive* de cara a comprobar el funcionamiento de los algoritmos en otro tipo series de datos.

Se ha decidido no incluir ninguno de los ejemplos de las pruebas realizadas ya que los resultados que se obtienen son muy similares a los que se han visto en el apartado anterior con las series del grupo BDI. Los algoritmos presentan las mismas habilidades a la hora de encontrar o no los puntos de corte deseados, manteniendo sus ventajas y sus inconvenientes.

Las conclusiones obtenidas sobre estas pruebas serán reflejadas en el siguiente apartado.

4.5. Discusión

Vistos los resultados de las pruebas realizadas con los tres algoritmos, el siguiente paso es el de interpretar los resultados y, finalmente, determinar en qué contexto puede actuar mejor cada uno de ellos y cuales son sus puntos fuertes y débiles.

Antes de centrarnos en las características individuales de cada algoritmo, caben destacar ciertos conceptos transversales a los tres algoritmos que se han ido observando a medida que se desarrollaba la fase de las pruebas. En este caso, uno de los conceptos importantes que se ha observado es el de el tiempo de ejecución de los tres algoritmos con las series temporales pertenecientes al contexto del grupo BDI.

Los tiempos de ejecución son muy altos en comparación con los tiempos necesarios para

segmentar las distintas series temporales del *UCR Archive* o de las series de ejemplo que los desarrolladores utilizan para promocionar sus algoritmos.

Los tiempos de ejecución pasan de ser de unos pocos segundos con las series del *UCR Archive* a ser incluso de varios minutos con las series del grupo BDI. Además, en las series del *UCR Archive*, el parámetro de entrada no afecta al tiempo de ejecución de la segmentación pero en las series del grupo BDI, en cambio, el valor del parámetro puede hacer que una ejecución inicial de 30 segundos pase a durar minutos.

Hay que tener en cuenta, además, que las series obtenidas del *UCR Archive* son series que salen de casos y contextos reales, y aun así tienen una apariencia de series ficticias de prueba en comparación con las del grupo BDI. Este hecho aporta un gran valor añadido tanto al trabajo que el grupo BDI desarrolla como a la realización de este proyecto, ya que pone en valor la complejidad y el volumen de los datos con los que se está trabajando, que están lejos de ser series triviales. No hay más que comprobar el ejemplo del algoritmo Matrix profile, del cual sus desarrolladores mencionan su gran potencia y velocidad de ejecución, que al enfrentarse a series de un día (86400 puntos) puede hasta llegar a necesitar unos cuantos minutos de tiempo, cuando con las series sacadas del *UCR Archive* tarda segundos en completarse.

Otro punto importante y común en los tres algoritmos es el hecho de que solo tengan un parámetro de entrada, punto positivo ya que limita las posibles combinaciones o desviaciones del resultado óptimo que pueda haber por la parte del usuario. Aún así hay que tener en cuenta que cada uno de esos parámetros no significa lo mismo y que no afecta de la misma manera al resultado final de la ejecución, tal y como se comentará en los detalles de cada uno de los algoritmos.

Teniendo en cuenta esos conceptos mencionados, a continuación se presentan las distintas conclusiones obtenidas para cada uno de los tres algoritmos de segmentación analizados:

Matrix profile

Tal y como se ha podido comprobar en la realización de las pruebas, este algoritmo no resulta del todo útil en la mayoría de series del contexto del grupo BDI. En la mayoría de series temporales probadas el algoritmo no era capaz de ofrecer una segmentación que se considerase adecuada, llegando algunos casos a presentar una solución que a todas luces no era para nada válida.

Aún así, tal y como se ha podido apreciar en alguna prueba, resulta muy preciso a la hora

de detectar un cambio de régimen en una serie temporal. El algoritmo es capaz de detectar el momento en el que una serie que está manteniendo una cierta frecuencia cambia, y pasa a tener otra frecuencia distinta. Para ese tipo de casos resulta muy interesante ya que, para detectar de manera correcta ese cambio, no depende en gran medida del parámetro de entrada, como sí lo hace por ejemplo el Simple segment.

Respecto al parámetro de entrada, los desarrolladores indican que no es muy relevante para obtener los resultados correctos pero en este caso se ha comprobado que afecta en gran medida al resultado final obtenido por el algoritmo CAC. Se han probado casos en los que pasar de un parámetro inicial de 20 a uno de 200 cambiaba por completo y mejoraba completamente la solución del problema pero que en otros casos el parámetro quizás podía rondar el 1000 o valores más altos. En definitiva, el valor depende en exceso del tamaño y la morfología de la serie que se está segmentando.

Un punto positivo que tiene respecto a su parametrización es el hecho de que se pueden establecer el número de segmentos que se quieran obtener en la finalización del algoritmo. Esto se debe a que en el último paso, el de obtener los puntos mínimos de la serie CAC, se le puede indicar el número de mínimos que se desean obtener. Esto no garantiza que los puntos encontrados vayan a ser los adecuados pero al menos permite limitarlos si por lo que sea se necesita.

En resumen, este algoritmo no parece resultar de gran utilidad en la mayoría de casos en los que el grupo BDI se maneja, debido a las carencias mencionadas.

Simple segment

En virtud de las pruebas realizadas, este algoritmo resulta más versátil, para las series utilizadas en este contexto, de lo que resulta por ejemplo el Matrix profile. El Simple segment es capaz de segmentar, utilizando un parámetro adecuado, series de prácticamente cualquier morfología, como ya hemos visto en los ejemplos de varios patrones o cambio de patrón único.

En líneas generales, con todas las series que se han ido segmentando utilizando este algoritmo se conseguían unos resultados válidos, en los que la segmentación indicada coincidía o era muy similar a la planteada inicialmente como posible solución correcta. Aún así, cabe destacar la importancia que tiene la correcta elección del parámetro de entrada del algoritmo, ya que influye en gran medida en el resultado a obtener, teniendo que movernos en un rango de valores de entre 200 y 80000. El problema principal se basa en la gran variación que puede llegar a tener el valor óptimo del parámetro dependiendo de la

serie segmentada, lo que dificulta en estos momentos la posibilidad de abordar la automatización o predicción del parámetro a utilizar. Si nos basamos en los casos de ejemplo ofrecidos por los desarrolladores, el parámetro utilizado se movía entorno a valores como el 0.001 o 0.0001 y en cambio con las series del grupo BDI se han tenido que llegar a utilizar valores de hasta 100000 en el parámetro. Esto, junto con lo mencionado anteriormente con los tiempos de ejecución, vuelve a poner en gran valor el trabajo desarrollado por el grupo BDI, reflejando el tamaño y complejidad de las series con las que se trabaja.

Por los resultados obtenidos en las pruebas, ya no es solo que el tamaño de la serie afecte al valor del parámetro, si no que, cambia también en gran medida dependiendo del número de cortes que se quieren obtener. Además, tal y como se ha podido comprobar, la morfología de la serie puede llegar a afectar, dándose casos en los que con la misma largura y el mismo número de cortes a obtener el parámetro óptimo ha variado excesivamente. Esto hace que el trabajo con el algoritmo sea mucho más manual que en los otros casos ya que, tal y como se ha dicho, el parámetro depende en gran medida del tamaño de la serie y del número de cortes que se buscan obtener.

A pesar de ello, a diferencia del Matrix profile, este algoritmo es capaz de obtener puntos de corte similares a los buscados, llegando en algunos casos a identificar los puntos que se consideraban ideales en esa serie temporal. Por ello, parece que este algoritmo sí que puede resultar de utilidad en este contexto, ya que nos permite obtener segmentaciones útiles con las series trabajadas y, por tanto, podría ser utilizado para realizar el posterior análisis y pruebas sobre la viabilidad de la segmentación para mejorar la reducción.

GGS

Este algoritmo podría considerarse el más versátil y el más sencillo de utilizar sin conocimiento previo del usuario. Esto es debido a la naturaleza de su único parámetro de entrada, ya que se le indica el número de puntos de corte máximos que se desean obtener. Esto permite que la utilización del algoritmo sea más sencilla y que esté en manos del propio algoritmo la capacidad de detectar los puntos de corte idóneos y no en el parámetro de entrada que el usuario tenga que ir probando hasta obtener un resultado óptimo.

Además, vistas las pruebas realizadas, el algoritmo ofrece muy buenos resultados en cuanto a las diferentes situaciones y diferentes series temporales con las que se ha trabajado. Funciona muy bien en todo tipo de casos ya que es capaz de detectar los patrones repetidos de una serie, detectar un único punto de cambio de patrón o de segmentar series algo más caóticas por los puntos que, a primera vista, podrían ser los correctos.

A pesar de ello, el GGS tiene una pega bastante grande en comparación con los otros dos algoritmos: su tiempo de ejecución. Aunque, tal y como se ha mencionado, los tres algoritmos tienen problemas en lo que respecta al tiempo de ejecución con series de muchos puntos, el GGS podría ser el que más problemas tiene a la hora de segmentar series muy grandes. El problema principal está en que el algoritmo no es ni siquiera de orden lineal y a medida de que la serie va aumentando en puntos el algoritmo va cada vez tardando mucho más en completar la segmentación. Curiosamente este algoritmo funciona de manera inversa a los otros dos respecto al tiempo de ejecución, tarda más cuantos más puntos de corte se deseen. En el Matrix profile y el Simple segment el tiempo es más alto cuando aumentamos el parámetro para poder obtener menos segmentos.

Aún con este problema, este algoritmo podría considerarse el mejor o, al menos, el más útil en estos momentos para trabajar con la segmentación de series temporales con las que el grupo BDI se maneja a diario debido a sus principales virtudes y a que sus defectos también los podemos encontrar en el resto de algoritmos en mayor o menor medida. Respecto a los resultados que se obtienen de la ejecución podría estar a la misma altura que el Simple segment, pero el hecho de que el GGS funcione indicándole el número de puntos de corte deseados le hace estar un paso por delante en este caso. Por ello, este algoritmos será el utilizado, cuando sea necesario, en el análisis y pruebas realizadas en el siguiente capítulo.

Resultados con las series del *UCR Archive*

Respecto a la habilidad de los algoritmos de obtener un corte más o menos óptimo con las series del *UCR Archive*, los resultados son muy similares a los obtenidos con las series del grupo BDI. En algunos de ellos son capaces de lograr los objetivos y en otros no, tal y como se ha visto en las pruebas realizadas con las series del grupo BDI.

Esto es debido a que, en el fondo, las series de ambos bancos de datos se pueden equiparar entre sí y los algoritmos actuarán de la misma manera ante ellos.

Si que existe una diferencia enorme, tal y como se ha dicho al comienzo del apartado, en los tiempos de ejecución de la segmentación. En las series probadas del *UCR Archive* los algoritmos tardan como mucho 2-3 segundos en ejecutarse mientras que con las del grupo BDI se tardan al menos 15-20 segundos por serie, llegando incluso a tardar minutos y hasta alguna hora si la serie es de varios días y se quieren sacar muy pocos segmentos.

Estas diferencias resultan muy notables y ponen en valor la dificultad de trabajar, probar y manejar series en cualquier tipo de investigación con estas series tan inmensas.

5. CAPÍTULO

Segmentación de series temporales

Una vez analizados los distintos algoritmos de segmentación seleccionados y determinado el más adecuado para nuestros intereses, el siguiente paso reside en comprobar si realmente merece la pena aplicar la segmentación a las series temporales previo a su reducción.

El objetivo de esto reside en comprobar si verdaderamente existe una ventaja (menor error al reconstruir, más porcentaje de reducción, ambos...) en reducir los segmentos obtenidos por el algoritmo de segmentación respecto a reducir las series completas.

Para esto, necesitamos conocer el contexto relacionado con la reducción y tener claros los indicadores que nos servirán para determinar en qué casos puede ser positiva la segmentación viendo los resultados obtenidos. En este capítulo primero analizaremos esos indicadores y todo lo que se necesita saber sobre la reducción de las series para posteriormente mostrar las pruebas realizadas y las conclusiones a las que se han llegado respecto a la reducción posterior a la segmentación.

5.1. Valoración de la reducción de las series

Para comprobar si una serie temporal ha sido reducida en mejor o peor medida, se utilizan dos elementos principales a tener en cuenta: el error en la reconstrucción de la serie y el porcentaje de reducción de la serie. Estos valores son los que determinan la calidad o cantidad de la reducción y, según los intereses finales de cada caso, se les puede dar más

o menos prioridad a la hora de aplicar una reducción a una serie temporal, siendo el caso ideal aquel en el que el error se reduzca y el porcentaje de reducción aumente.

5.1.1. Error en la reconstrucción

Cuando a una serie temporal reducida se le aplica la reconstrucción correspondiente, no siempre podemos obtener la serie idéntica a la que teníamos originalmente. De hecho, la mayoría de los algoritmos no permiten que esto suceda, siendo en la práctica un caso bastante aislado. La mayoría de las series obtenidas tienen diferencias en los valores respecto al original y para poder cuantificar dicho error aplicamos la siguiente fórmula:

$$\sqrt{\frac{sum((SO-SR)^2)}{length(SO)}} \tag{5.1}$$

siendo *SO* la Serie Original y *SR* la Serie Reducida. El valor obtenido de dicha fórmula será el que utilizaremos para valorar cuanto error tiene cada serie reconstruida respecto a la serie original, siendo un valor inferior el reflejo de una reducción más precisa.

5.1.2. Porcentaje de reducción de la serie

Otro modo de evaluar series reducidas es comparar el porcentaje de reducción respecto a las series originales íntegras. Se calcula el espacio que ocupa el fichero de la serie original almacenada en el disco y se compara con el tamaño que ocupa el fichero de la serie ya reducida. Con esto obtenemos en qué porcentaje se ha reducido la serie y podemos saber, por ejemplo, si con un algoritmo se ha reducido en mayor o menor medida que con otro.

5.2. Pruebas

En esta sección se presentan los distintos enfoques tomados en la fase del desarrollo de las distintas pruebas relacionadas con el análisis del potencial de la segmentación de series temporales. Serán presentados los conceptos necesarios para llevar a acabo dichas pruebas, junto con algún ejemplo relevante de las distintas pruebas realizadas para cada enfoque durante esta fase.

Para la realización de las pruebas se han utilizado las series de UROLA proporcionadas por el grupo BDI, las cuales habían sido pre-procesadas y previamente reducidas con los algoritmos DFT, PIP y RLE principalmente. La idea general de las pruebas consiste en, por un lado, reducir las series temporales con el algoritmo original y, por otro, reducir los segmentos previamente obtenidos (ver figura 5.1), ya sea cortados de manera arbitraria o a través del algoritmo de segmentación seleccionado en el capítulo 4.

La intención de las pruebas es la de comprobar si la segmentación es válida o no sin tener en cuenta el modo en el que se hayan obtenido los segmentos. Una vez obtenidos dichos resultados se comparan para determinar si se consigue una ganancia ya sea del error y/o del porcentaje de reducción, y si la segmentación puede llegar a ser útil en este contexto.

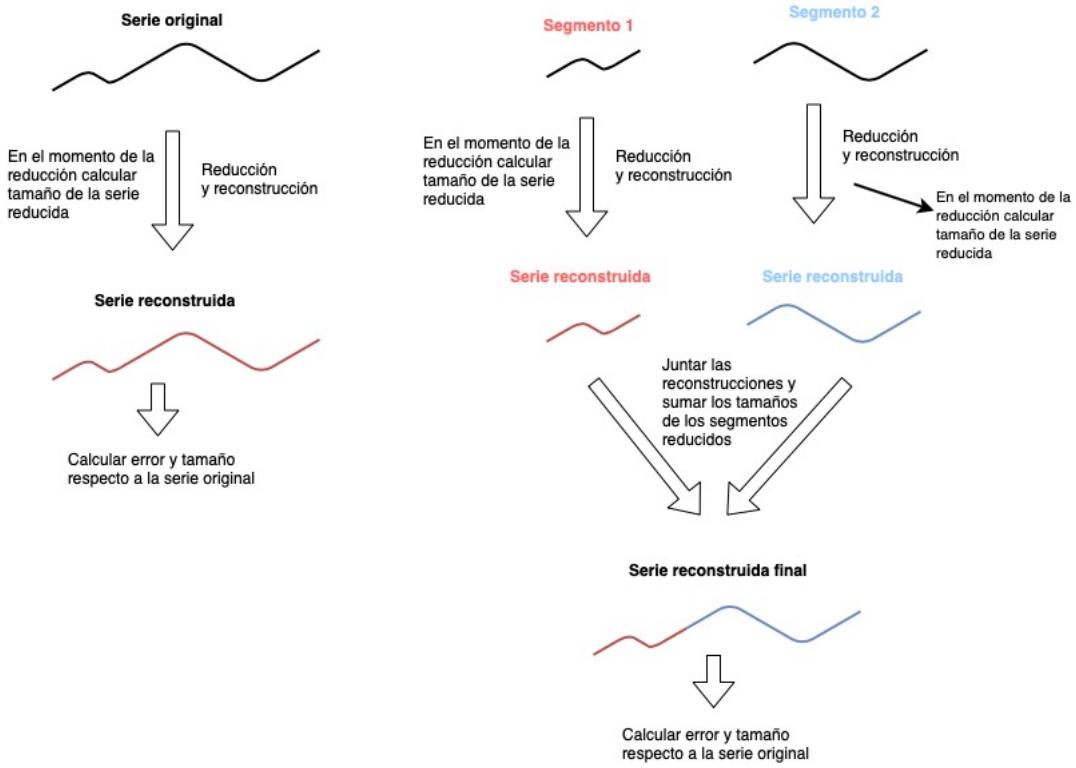


Figura 5.1: Diagrama en el que se explica el proceso llevado a cabo para la obtención y comparación de resultados entre la reducción de la serie original y la reducción de los segmentos obtenidos.

El propósito inicial de las pruebas era el de reducir los segmentos con parámetros que fueran proporcionales a la de la reducción de la serie original, para así mantener el porcentaje de reducción final y buscar una mejora en el error obtenido. Es decir, que si por ejemplo habíamos reducido originalmente una serie con DFT siendo el parámetro de entrada el 10 % de puntos de la serie, la intención era reducir con DFT cada segmento obtenido, po-

niendo como parámetro de entrada el 10 % de puntos de dicho segmento para así obtener el mismo número de puntos totales en la serie reducida.

Aún siguiendo este criterio, hay que tener en cuenta que, si aplicamos distintos algoritmos a los segmentos obtenidos para una serie temporal, no podemos seguir este mismo procedimiento ya que los puntos almacenados por cada algoritmo pueden ocupar diferente tamaño en disco. Por ejemplo, comparando los algoritmos de PIP y de DFT, observamos que 7505 puntos de DFT ocupan en disco 111KB y 7505 puntos de PIP ocupan 88KB, por tanto, haber cogido por un lado el 10 % de puntos de un segmento con PIP y por otro lado el 10 % de puntos de un segmento con DFT no nos asegura tener un porcentaje de reducción similar al original, cosa que si utilizamos para todos los segmentos la misma técnica sí podríamos.

Teniendo en cuenta este hecho, en los casos como este que hemos descrito se tendrá en cuenta tanto el error final como el porcentaje de reducción, ya que es posible que aparezcan resultados en los que el error haya empeorado pero porque hemos reducido mucho más que con la reducción original, o viceversa. Así, teniendo en cuenta eso, se pueden variar los parámetros e ir viendo los resultados para poder así equilibrar el porcentaje de reducción final respecto a la reducción de la serie original.

Este ha sido el proceso que se ha llevado a cabo para obtener el resultado final tanto de reducción como del error de las series segmentadas y así poder compararlas con la reducción original:

- Para obtener el porcentaje de reducción final se ha exportado el fichero correspondiente de cada segmento reducido y se ha calculado su tamaño en disco. Luego se han sumado dichos valores y se ha comparado con el tamaño ocupado por la serie original.
- Para obtener el error final se han ido reduciendo uno a uno los segmentos con el parámetro adecuado seleccionado en cada caso y posteriormente reconstruyéndolos también de uno en uno. Cada una de esas reconstrucciones se ha ido juntando con las demás para obtener una serie reconstruida final. Una vez con eso se calcula el error entre la serie original y esta serie que hemos construido con las reconstrucciones de cada uno de los segmentos reducidos.

Para la realización de las distintas pruebas se han implementado diversos scripts tanto en R como en Python. Han sido desarrollados con la idea de poder automatizar ciertos tipos de pruebas y de servir de ayuda en aquellas en las que requerían de una ejecución manual.

En A.2 se muestra un ejemplo de uno de los algoritmos desarrollados para poder realizar pruebas de manera más rápida y productiva.

```
reduccionSegmentos <- function(fichero){
  segmentos <- read.table(fichero)
  res <- c()
  original <- read.table(file,header = T,sep = ",",dec = ".")
  adp <- prepare4DFT(original$value)
  red <- reduceDFT(adp,nrow(original)/10)
  rec <- reconstructDFT(red)

  tamanioRedSeg <- 0

  totall <- length(original$value)

  totalll <- 0

  errorOriginal <- getRMSE(original$value,rec$y)
  tamanioOriginal <- file.size(file, units = "B")
  urlRed <- "originalREDUCIDO.csv"
  saveDFT(red,urlRed,"2019-4-29",1)
  tamanioReducido <- file.size(urlRed,units = "B")
  ratioReduccionOriginal <- 100 - (tamanioReducido/tamanioOriginal*100)

  originalRec <- c()

  for (segmento in 1:length(segmentos$V1)  ) {
    if (segmento > 1){
      seg <- read.table(file,sep = ",", dec=".",skip = segmentos$V1[segmento-1] + 1, nrows =
   segmentos$V1[segmento]-segmentos$V1[segmento-1])
      numSeg <- (segmentos$V1[segmento]-segmentos$V1[segmento-1])
      num <- numSeg/10
    } else{
      seg <- read.table(file,sep = ",", dec=".",skip = 1,nrows = segmentos$V1[segmento])
      numSeg <- segmentos$V1[segmento]
      num <- (segmentos$V1[segmento])/10
    }
    plot(seg$V2,type = "l")
    print(paste0("Segmento ", segmento, ":"))
    n <- readline(prompt="¿Qué algoritmo quieres aplicar?(1 CHEV, 2 DFT, 3 PIP) ")
    originalRec <- c(originalRec,seg$V2)
    if (n == 1){
      grado <- readline(prompt="Indica el grado del polinomio: ")
      grado <- strtoi(grado)
      nas <- seg$V2[!is.na(seg$V2)]
      adp <- prepare4CHEB(nas)
      red <- reduceCHEB(adp,grado)
      rec <- reconstructCHEB(red)
      res <- c(res,rec$y)
      saveCHEB(red,paste0("fichSegRed.csv"),"2019-4-29",1)
      print(getRMSE(seg$V2,rec$y))
```

```
48        totalll <- totalll + length(seg$V2)
49
50      } else if (n == 2){
51        nas <- seg$V2[!is.na(seg$V2)]
52        adp <- prepare4DFT(nas)
53        red <- reduceDFT(adp,num)
54        rec <- reconstructDFT(red)
55        res <- c(res,rec$y)
56        saveDFT(red,paste0("fichSegRed.csv"),"2019-4-29",1)
57        print(getRMSE(seg$V2,rec$y))
58        totalll <- totalll + length(seg$V2)
59      } else if (n == 3){
60        puntos <- readline(prompt="Indica el número de puntos: ")
61        nas <- seg$V2[!is.na(seg$V2)]
62        adp <- prepare4PIP(nas)
63        red <- reducePIP(adp,puntos)
64        rec <- reconstructPIP(red)
65        res <- c(res,rec$y)
66        savePIP(red,"fichSegRed.csv","2019-4-29",1)
67        print(getRMSE(seg$V2,rec$y))
68        totalll <- totalll + length(seg$V2)
69      } else {
70        stop("Tienes que introducir uno de esos 3 números")
71      }
72      tamanioRedSeg <- tamanioRedSeg + file.size("fichSegRed.csv",units = "B")
73    }
74    ratioSeg <- 100 -(tamanioRedSeg/tamanioOriginal*100)
75    print(ratioReduccionOriginal[1])
76    print(ratioSeg[1])
77    print(errorOriginal)
78    print(getRMSE(original$value,res))
79    fin <- res
80  }
```

En este caso tenemos un script desarrollado en R que recibe como parámetros una serie temporal y un fichero de texto con los puntos donde cortar la serie. Con esos elementos va segmentando la serie temporal y, por cada uno de los segmentos generado, muestra al usuario en pantalla su representación gráfica y pregunta con qué técnica de reducción y con qué parámetro desea reducir ese segmento. Una vez el usuario finaliza con todos los segmentos, el script calcula el error final y el porcentaje de reducción obtenidos con esa combinación seleccionada por el usuario, mostrando por pantalla tanto el error y el porcentaje de reducción de la serie completa como el error y el porcentaje de reducción de la serie segmentada.

Este ejemplo mencionado es uno de los más completos realizados pero también han sido desarrollados otros scripts similares a este para realizar otro tipo de pruebas relacionadas con la segmentación. Dichos scripts quedan reflejados en el anexo A.

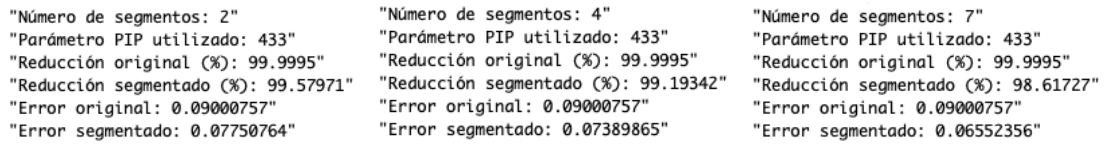
Las pruebas realizadas podrían dividirse en dos tipos: el de segmentar la serie y reducir cada segmento con la técnica usada en la serie original, y el de segmentar la serie y reducir cada segmento con la técnica ideal para dicho segmento. Cabe destacar que, en las pruebas, se presupone que las reducciones realizadas a las series originales completas son las óptimas para cada una de ellas y que la serie a tratar ha sido previamente preprocesada.

A continuación se presentan algunos ejemplos de pruebas realizadas durante el desarrollo de esta fase del proyecto:

5.2.1. Pruebas de reducción de los segmentos con la técnica original

Las pruebas realizadas en este caso van enfocadas a comprobar si una serie puede llegar a ser reducida en mayor medida si la segmentamos y a cada segmento le aplicamos la misma técnica de reducción que la aplicada a la reducción original. Con esto se busca comprobar dos cosas: que la propia técnica ofrezca mejores resultados por aplicarse a series más pequeñas o porque la técnica sea capaz de obtener mejores resultados debido a las diferencias morfológicas que puede haber entre un segmento y otro.

En las figuras 5.2 y 5.3 se presentan los resultados de una prueba realizada para comprobar la segmentación de una serie reducida originalmente con PIP y reducir dichos segmentos también con PIP, utilizando el mismo parámetro. Para ello se ha segmentado la serie temporal en 2, 4, 7, 14 y 39 segmentos, para comprobar si el número de segmentos afectaba al resultado obtenido a medida que aumentaban en número.



(a) 2 segmentos **(b)** 4 segmentos **(c)** 7 segmentos

Figura 5.2: Resultados de la aplicación del PIP a serie segmentada en 2, 4 y 7 segmentos.

Se puede comprobar como en este caso, con el algoritmo de PIP, a medida que se van aumentando el número de segmentos se va reduciendo tanto el error como la reducción total. En este caso el resultado podría generar ciertas dudas de si puede ser mejor que el original o no, ya que por un lado el resultado del error es mejor pero el resultado de la reducción peor. Para comprobar si esto podría traer consecuencias positivas, se realizó la

```
"Número de segmentos: 14"
"Parámetro PIP utilizado: 433"
"Reducción original (%): 99.9995"
"Reducción segmentado (%): 97.38307"
"Error original: 0.09000757"
"Error segmentado: 0.0619951"
```

(a) 14 segmentos

```
"Número de segmentos: 39"
"Parámetro PIP utilizado: 433"
"Reducción original (%): 99.9995"
"Reducción segmentado (%): 92.87189"
"Error original: 0.09000757"
"Error segmentado: 0.03540748"
```

(b) 39 segmentos

Figura 5.3: Resultados de la aplicación del PIP a serie segmentada en 14 y 39 segmentos.

prueba de reducir la serie original, pero esta vez multiplicando por 2, 4 y 7 el parámetro de entrada del PIP.

En la figura 5.4 se pueden ver los resultados obtenidos doblando y cuadruplicando y multiplicando por 7 el parámetro en la serie inicial, y es que se han obtenido unos resultados prácticamente idénticos a los que hemos visto en los casos de la figura 5.2 reduciendo la serie segmentada en 2, 4 y 7 trozos. Esto demuestra que los resultados obtenidos en realidad no representan ningún tipo de mejora respecto a lo que se podría lograr simplemente modificando el parámetro inicial de la reducción con la serie completa.

```
"Parámetro PIP utilizado: 866"
"Reducción original (%): 99.57638"
"Error original: 0.07794733"
```

(a) Parámetro doblado

```
"Parámetro PIP utilizado: 1732"
"Reducción original (%): 99.15263"
"Error original: 0.07086614"
```

(b) Parámetro cuadruplicado

```
"Parámetro PIP utilizado: 3031"
"Reducción original (%): 98.51472"
"Error original: 0.06189635"
```

(c) Parámetro multiplicado por 7

Figura 5.4: Resultados de la aplicación del PIP a la serie inicial multiplicando el parámetro por 2, 4 y 7.

Con el algoritmo DFT también se realizaron varias pruebas en las que se buscaba segmentar la serie con distinto número de cortes e ir probando los resultados que iban obteniéndose. Una de las pruebas que se realizó fue la de aplicar el script que se muestra en el anexo A.4. Con ese script se recorren todas las series que estaban originalmente reducidas con DFT, se les aplica el algoritmo de segmentación del Simple segment y se reducen cada uno de los segmentos con DFT.

En este caso este script fue ejecutado en una máquina virtual en Google Cloud, debido al tiempo de ejecución que requería el script. La prueba fue repetida con los parámetros del Simple segment siendo 100, 2000, 5000, 20000 y 50000 y el resumen de los resultados se puede ver reflejado en la figura 5.5.

Estos resultados reflejan el cambio medio que ha habido tanto del error como de la reducción final por cada uno de los parámetros de entrada definidos, en la que los números

```
          Error Reduccion(%)
2000  0.03001241    0.2301964
5000  0.03537072    0.2614729
20000 0.02014743    0.1818517
100   0.06347948    0.2359347
50000 0.04658639    0.1406262
>
```

Figura 5.5: Resultados medios obtenidos en las distintas ejecuciones de los scripts ejecutados en Google Cloud.

de error y reducción definidos en la figura representan la resta entre el error y reducción originales y el error y reducción segmentados. Por ello, en este caso los resultados obtenidos no son positivos ya que las reducciones se mantienen prácticamente idénticas pero los errores han empeorado respecto a los originales.

Finalmente, teniendo en cuenta las características de la técnica del DFT, en la que se convierte la serie temporal al dominio de las frecuencias (ver apartado 2.2.2 para más detalles), una idea a comprobar era la de segmentar una serie por los puntos en los que el cambio en la frecuencia fuese muy notorio. En la figura 5.6 tenemos una serie en la que vemos una clara diferencia entre la primera mitad de la serie y la segunda mitad.

Para comprobar si se podrían obtener mejores resultados, se probó a segmentar la serie por ese punto (alrededor del 125.000) y a reducir los segmentos obtenidos con el algoritmo DFT manteniendo el parámetro en un 10 % (mismo porcentaje que el utilizado con la serie completa) de los puntos del segmento. Una vez realizado, los resultados se pueden observar en la figura 5.7.

Los resultados no indican ninguna mejora sustancial respecto a la reducción original ya que el porcentaje de reducción es casi idéntico (buscado y esperado manteniendo el 10 % en las reducciones) y el error tampoco difiere en exceso del original.

5.2.2. Pruebas de reducción de los segmentos con la técnica más adecuada para cada uno

En este caso, la intención de las pruebas es comprobar si, aplicando la técnica que mejor resultados da para cada segmento, obtenemos unos resultados mejores que los de la reducción original. La idea se basa en el hecho de que existen series en las que en cierto punto

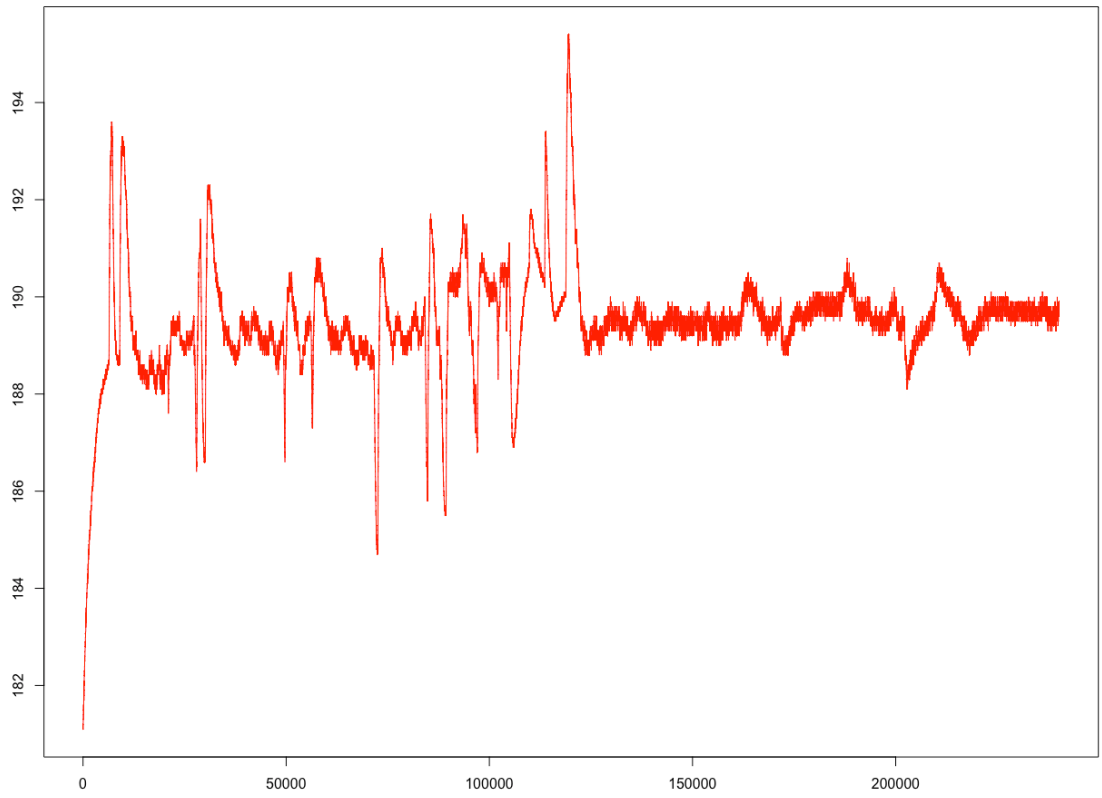


Figura 5.6: Serie utilizada para comprobar la reducción del DFT segmentando la serie por frecuencias.

```
"Reducción original (%): 95.3082108329844"
"Reducción segmentado (%): 95.450886698041"
"Error original: 0.0338597772289913"
"Error segmentado: 0.0353649499718519"
```

Figura 5.7: Resultados obtenidos de segmentar y reducir una serie buscando la diferencia en la amplitud frecuencia.

ocurre un cambio de patrón o un cambio de morfología muy notorio, siendo claramente ineficiente el algoritmo original en dicho segmento distinto.

En la serie de la figura 5.8 se puede observar de manera clara la diferencia que existe en la forma de la serie, siendo la primera parte similar a una serie del dominio de las frecuencias, y la segunda parte similar a la forma de una ecuación de segundo grado. En este caso se segmentó la serie separando esos segmentos y reduciendo la primera parte con DFT y el resto con CHEB. Tal y como se puede observar en los resultados mostrados en la figura 5.9, los números obtenidos son excelentes ya que se mantiene e incluso se mejora un poco la reducción y a su vez el error se reduce a más de la mitad.

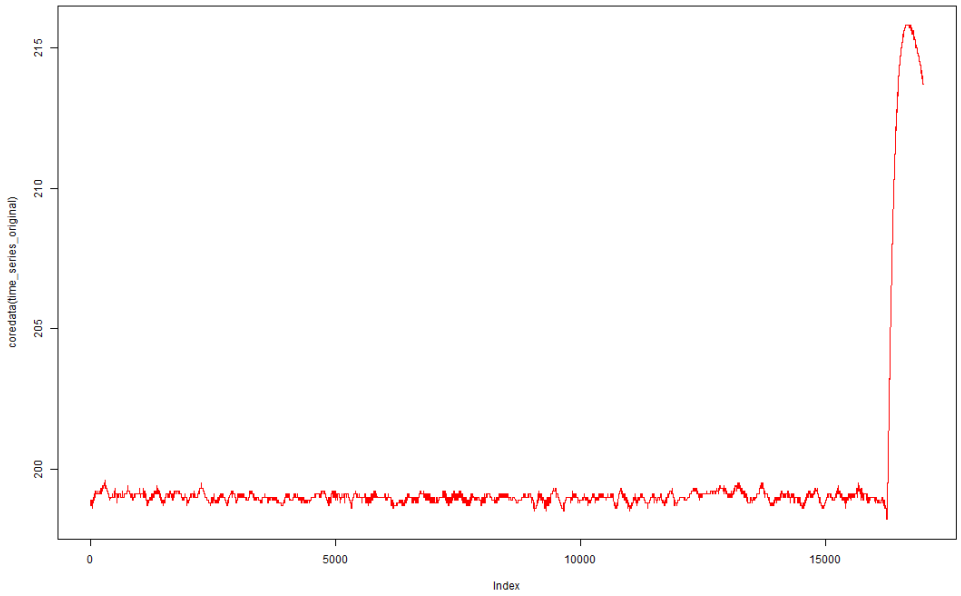


Figura 5.8: Serie con una morfología diferente en dos de sus segmentos utilizada como prueba para la reducción con distintos algoritmos.

```
"Reducción original (%): 95.1015351467816"
"Reducción segmentado (%): 95.722919472125"
"Error original: 0.0892342485457429"
"Error segmentado: 0.0424247255044637"
```

Figura 5.9: Serie con una morfología diferente en dos de sus segmentos utilizada como prueba para la reducción con distintos algoritmos.

Otra de las pruebas realizadas es la llevada a cabo con la serie representada en la figura 5.10. En ella vemos otro tramo de la serie en que a priori podemos identificar como que

requiere de una técnica de reducción distinta a la original (en este caso la DFT). En este caso, los segmentos han sido reducidos combinando DFT, PIP con parámetro 100 y CHEB con parámetro 10, viéndose los resultados reflejados en la figura 5.11.

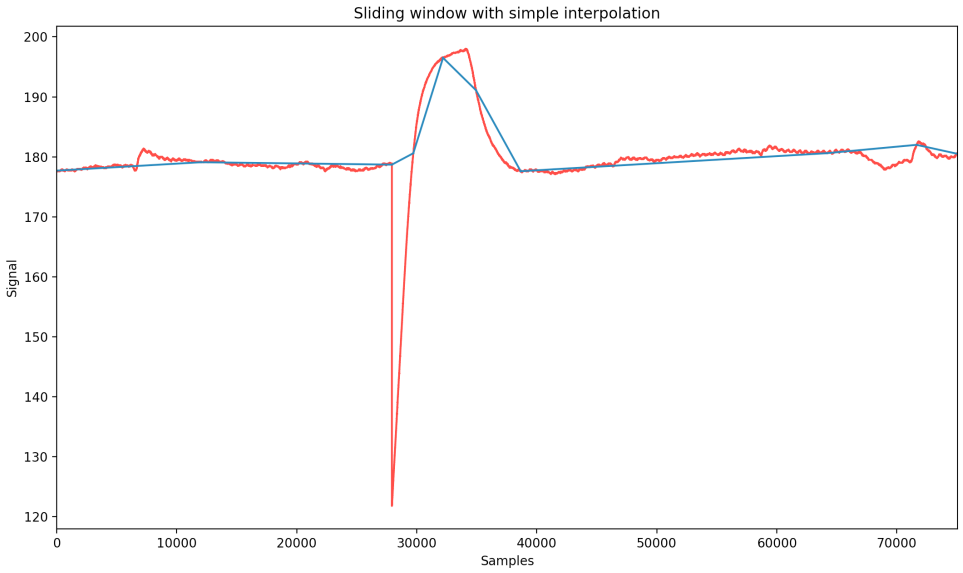


Figura 5.10: Serie utilizada como prueba para la utilización de distintas técnicas según la morfología de cada segmento obtenido. En rojo la serie temporal y cada raya en azul representa un segmento obtenido posterior a la segmentación.

```
"Reducción original (%): 94.7935450525469"
"Reducción segmentado (%): 96.1949405616493"
"Error original: 0.148317334387295"
"Error segmentado: 0.0350119249822127"
```

Figura 5.11: Resultados correspondientes a la prueba realizada con la serie de la figura 5.10

Tal y como se puede observar, los resultados en este caso son excelentes, aumentando la reducción en más de un punto y reduciendo el error de 0.14 a 0.035.

5.3. Discusión

Vistos los resultados de las pruebas realizadas se puede llegar a la conclusión de dos ideas principales: la primera, que la segmentación de series temporales puede ayudar a

conseguir una mejor reducción final de las mismas y, la segunda, que estamos antes un problema que aún tiene un largo camino por delante y un potencial de aplicación enorme.

En líneas generales, se ha visto que la idea de segmentar las series buscando una posterior mejora de la reducción no es del todo equivocada, y que se pueden obtener resultados muy positivos en algunas de las casuísticas planteadas durante el desarrollo de las pruebas. Como se ha podido ver, tenemos tanto resultados positivos como otros que no lo son tanto.

Es muy clara la ventaja que nos da segmentar una serie y poder reducir los segmentos obtenidos con técnicas distintas, ya que estamos aprovechando la capacidad de, en cada segmento, aplicarle el algoritmo que mejores resultados nos va a dar. En las pruebas se han podido observar series en las que el cambio de su morfología era muy claro y que, por ejemplo, en cierto segmento pasar de reducir con DFT a reducir con CHEB mejoraba enormemente el error obtenido en ese segmento, manteniendo una reducción similar a la original.

Como también se ha comprobado, si los segmentos obtenidos al segmentar una serie se tienen que reducir con el mismo algoritmo con el que se reducía la serie completa, en la gran mayoría de casos no se obtenía un resultado positivo. Si que nos hemos dado cuenta de que, en estos casos mencionados, en líneas generales se obtenían resultados en los que, o bajaba el error y bajaba la reducción, o subía el error y subía la reducción. Por tanto, si que podría descartarse la idea de segmentar una serie con un algoritmo y automáticamente reducir los segmentos con las misma técnica utilizada en la reducción original ya que, o los resultados no eran buenos (subía el error y baja la reducción), o dichos resultados podían conseguirse modificando el parámetro de la técnica inicial utilizado en la serie completa.

Esto que se acaba de mencionar, en cambio, tiene en parte una lectura positiva. El hecho de que en algunos casos (como el DFT o el PIP), segmentar la serie y reducir los segmentos con la técnica original nos arrojen unos resultados similares a los iniciales (como mucho variando un poco los parámetros para igualar errores y reducciones), permite que al reducir un segmento con una técnica que claramente sea mejor, los resultados totales finales mejoren respecto al original. Si, por ejemplo, los segmentos en los que se ha mantenido la técnica original obtuviesen resultados muchos peores, por mucho que mejorásemos en los segmentos en los que cambiamos de técnica los resultados globales no mejorarían o incluso empeorarían.

Respecto al tiempo de ejecución de las series, no se han realizado pruebas de medición

exacta de los tiempo utilizados por las técnicas pero por la experiencia adquirida durante la realización de las pruebas se puede concluir que los tiempos son muy similares a los de la reducción original. Debido a que las series segmentadas son más pequeñas que la original, el tiempo de ejecución de la reducción es menor, teniendo en cuenta incluso que las reducciones en series de hasta 86400 son prácticamente inmediatas. Por tanto, el tiempo de ejecución no se considera un elemento a tener en cuenta ni como elemento positivo ni como elemento negativo.

En definitiva, queda demostrado que la segmentación de series temporales puede resultar ser una aproximación muy válida para la optimización del espacio de almacenamiento consumido por las series de datos. Se ha visto que no en todos los casos es aplicable pero que existen casos bastante relevantes en los que reducir una serie segmentada permite obtener un menor error y reducir el tamaño en mayor medida.

6. CAPÍTULO

Análisis de requisitos

En este apartado se presenta el análisis realizado, previo al desarrollo de la web, en la que se han identificado las funcionalidades que requiere la plataforma y en el que se han definido los distintos casos de uso principales según el rol del usuario de la plataforma.

6.1. Funcionalidades

Posterior a una reunión con la tutora del proyecto y con los distintos miembros del grupo BDI relacionados con este proyecto, se identificaron las principales funcionalidades de la plataforma web a desarrollar. Se identificaron tres funcionalidades principales, que han sido complementadas con alguna otra funcionalidad más, no explícitas y en algunos casos intrínsecas a las propias funcionalidades principales.

A continuación se presentan dichas funcionalidades:

6.1.1. Funcionalidades principales

Segmentación de series

La primera funcionalidad principal es la de poder segmentar series temporales subidas a la plataforma. El sistema ofrecerá la posibilidad de segmentar una serie temporal con los distintos algoritmos de segmentación seleccionados y que han sido mencionados en el apartado 4.

El usuario podrá segmentar las series de manera independiente con los distintos algoritmos, introduciendo por cada uno de ellos el parámetro de entrada correspondiente y obteniendo el resultado de manera individualizada. Esto podrá realizarlo de una manera muy visual, teniendo los tres resultados en la misma página para poder así comparar de un solo vistazo qué segmentación es más óptima a primera vista.

Reducción de las series segmentadas

Una vez realizada la segmentación de la serie, el usuario podrá seleccionar la que mas le convenga y tendrá la posibilidad de reducir dicha serie segmentada con los distintos algoritmos de reducción presentados en el apartado 2.2.2. El usuario podrá visualizar tanto la serie completa segmentada como los diferentes segmentos que se han obtenido.

Se podrá reducir cada uno de los segmentos de manera individual, seleccionando para cada uno de ellos la técnica de reducción deseada y actualizando, en tiempo real, el resultado global de la reducción final obtenida, para que el usuario pueda ir comprobando segmento a segmento qué técnica es la que le permite una mejor reducción de su serie original. Los datos sobre la reducción estarán visibles en todo momento y se irán actualizando, tal y como se ha mencionado, cada vez que se reduzca un nuevo segmento.

Inicio de sesión y registro

El usuario podrá registrarse e iniciar sesión en el sistema. De hecho, como se explica en el apartado 6.2.1, este inicio de sesión será necesario si el usuario quiere utilizar las funcionalidades que la plataforma ofrece.

Visualización de casos exitosos

Se ofrecerá una lista con unos ejemplos de series temporales en los que ha sido aplicada una segmentación y posteriormente una reducción de dichos segmentos. Se mostrarán como ejemplo de casos en los que se consigue una ganancia reduciendo los segmentos obtenidos en vez de reduciendo la serie original completa.

6.1.2. Funcionalidades secundarias

Subida y almacenamiento de series temporales

De manera intrínseca a las funcionalidades de segmentación y reducción de series, es necesario que la plataforma permita subir una serie desde un fichero local del usuario y que trabaje así con ella en la herramienta.

Además, dichas series deberán ser almacenadas en alguna base de datos, para que así el usuario pudiera consultar todas las series que haya subido a la plataforma junto con los valores obtenidos de los resultados de segmentación y reducción en el caso de que ya se le hayan realizado.

Visualización de manera gráfica de las series temporales

Como funcionalidad necesaria e implícita para poder ofrecer las principales, es necesario que la plataforma pueda visualizar gráficas que representen todos los puntos de la serie temporal con la que se está trabajando. En la plataforma se podrán visualizar las series temporales subidas o almacenadas por el usuario, pudiendo hacer zoom, desplazarse por ella.

Como funcionalidad extra se podrá descargar una imagen del estado en el que se encuentre la gráfica en un momento determinado, por ejemplo: Descargar imagen de la serie sin segmentar, de la serie con los segmentos marcados o de un fragmento de la serie una vez se ha hecho zoom

Descarga de las series

El sistema ofrecerá la posibilidad de descargar tanto un fichero con la serie original segmentada como otro con la serie segmentada y reducida con los valores de la serie reducida de cada segmento. El primero de ellos se podrá realizar en cuanto hayamos seleccionado una segmentación y el otro estará accesible cuando se hayan reducido todos los segmentos de la serie temporal.

Servicio de segmentación de series

Para realizar la segmentación de las series en la plataforma, se ha decidido implementar un servicio web API REST que será capaz de devolver los puntos de segmentación de una serie que se le envíe. Se ha decidido esto para facilitar su utilización en la plataforma web y para dotar de independencia y de posible reutilización a esta funcionalidad. El diseño y desarrollo detallado de dicho servicio se ha definido de manera completa en el apartado 7.1 ya que se ha considerado que no es tan extenso y por tanto sería más óptimo presentarlo completo en un mismo apartado.

6.2. Casos de uso

A continuación se presentan los distintos elementos relacionados con la definición de los roles de la aplicación y de la estructura de los casos de uso.

6.2.1. Definición de roles

Se han definido dos roles principales para clasificar a los usuarios que trabajarán con la plataforma:

- **Usuario no conectado:** Engloba a los usuarios que acceden al sistema web pero que no han iniciado sesión. Solo podrán visualizar la página inicial con el contenido que explica las funcionalidades de la web y la página donde se menciona al autor de dicha plataforma.
- **Usuario conectado sin verificar:** Engloba a los usuarios que ya han iniciado sesión en la plataforma pero no han verificado su correo electrónico. Podrán visualizar los

ejemplos exitosos de series segmentadas y las páginas en las que se presentan y detallan de manera general los distintos algoritmos de segmentación y reducción. No podrá acceder a la herramienta de segmentación, es decir, no podrá subir sus series y trabajar con la plataforma.

- **Usuario conectado verificado:** Engloba a los usuarios que ya han iniciado sesión en la plataforma y han verificado su correo electrónico. Podrán visualizar los ejemplos exitosos de series segmentadas y las páginas en las que se presentan y detallan de manera general los distintos algoritmos de segmentación y reducción. Podrán acceder a la herramienta de segmentado y subir series con las que trabajar en ella.

6.2.2. Casos de uso

Basándonos en la jerarquía que hemos creado con la definición de los roles, el siguiente paso es el de definir la estructura de casos de uso para cada uno de los roles del sistema. Para presentar estos casos de uso se han realizado tres diagramas, uno para los usuarios no conectados (figura 6.1), otro para los usuarios conectados sin verificar (figura 6.2) y el otro para los usuarios conectados verificados (figura 6.3).

En estos diagramas aparece un actor extra que no representa ningún usuario físico que interactúe con la aplicación, si no que será una representación del propio sistema para reflejar las acciones que debe de llevar a cabo en ciertos casos de uso.

A continuación se presentan y detallan los distintos casos de uso presentados en los diagramas, desgranando los distintos eventos y acciones ocurridos en los mismos. En este caso puede que no todos los elementos que aparecen en los diagramas vayan a ser detallados porque puede que se mencionen y detallen en el marco de otro de los casos.

Iniciar sesión

Cuando el usuario pulsa sobre el botón de *Sign in* de la barra superior se abre una ventana en la que el usuario deberá introducir su email y su contraseña para poder autenticarse (ver figura 6.4).

Eventos:

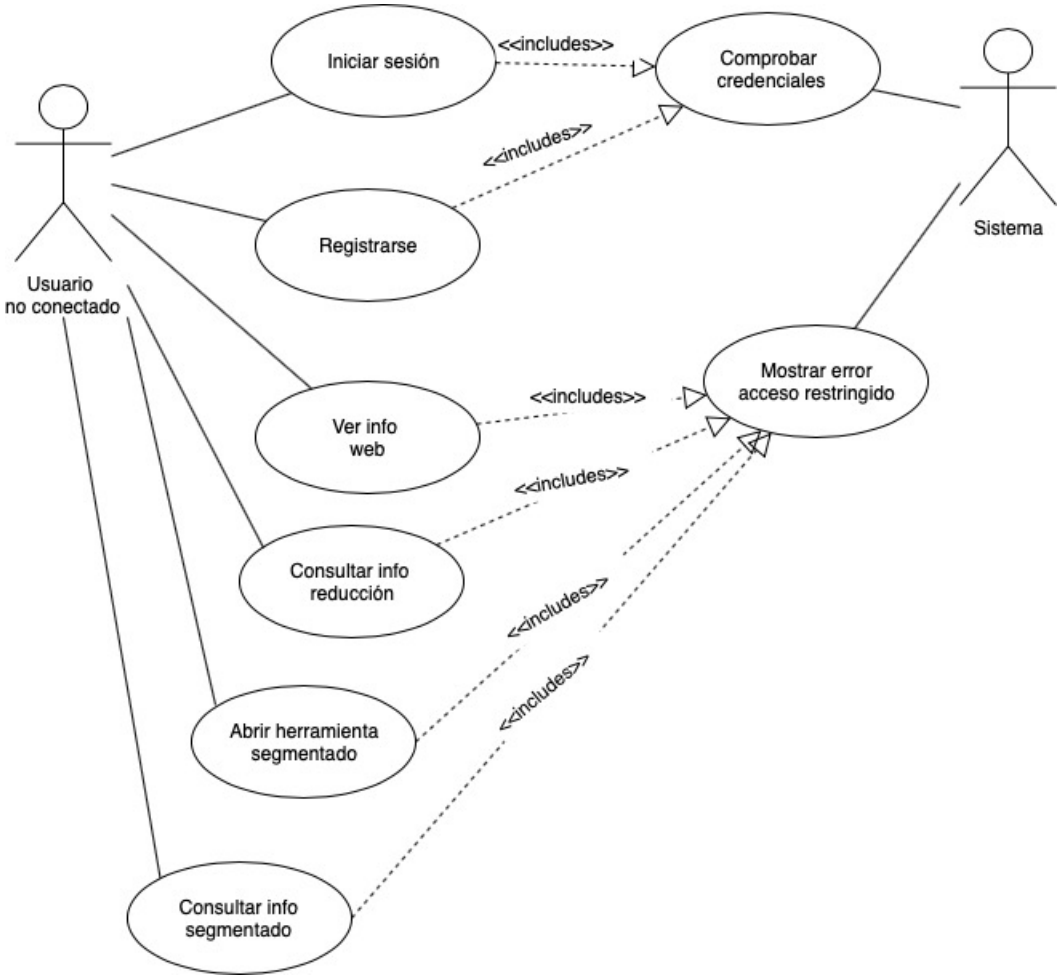


Figura 6.1: Modelo de casos de uso de los usuarios con rol de no conectados.

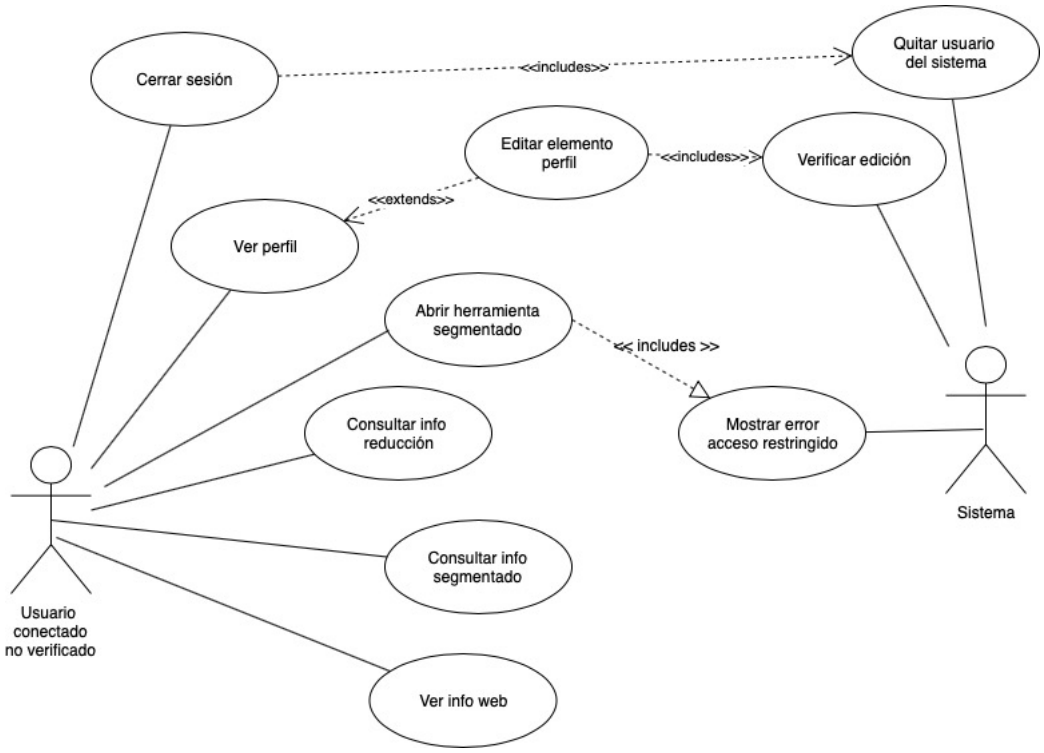


Figura 6.2: Modelo de casos de uso de los usuarios con rol de conectados sin verificar.

1. El usuario accede a la ventana de inicio de sesión a través del botón *Sign in* de la barra superior.

2. El usuario introduce sus credenciales de inicio de sesión del sistema y pulsa en el botón de *Sign in* para continuar.

3. El sistema comprobará la validez de las credenciales. Si las credenciales son correctas el sistema saluda con un mensaje de bienvenida y cerrará la ventana, configurando en el sistema que ese usuario ha sido conectado. Si las credenciales no son correctas el sistema avisará al usuario del error manteniendo la ventana abierta para que pueda volver a intentarlo.

Registrarse

Cuando el usuario pulsa sobre el botón de *Sign up* de la barra superior se abre una ven-

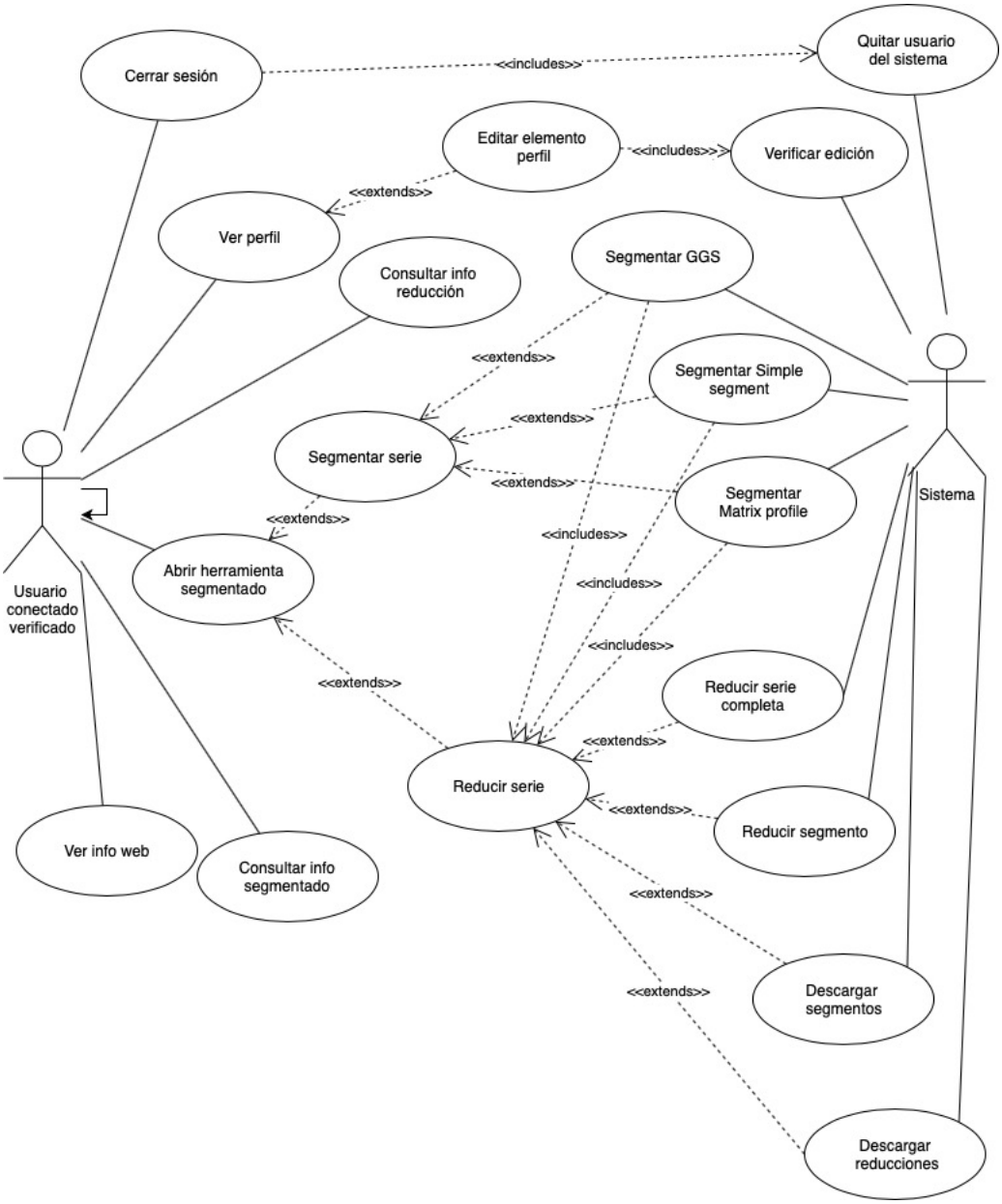


Figura 6.3: Modelo de casos de uso de los usuarios con rol de conectados verificados.

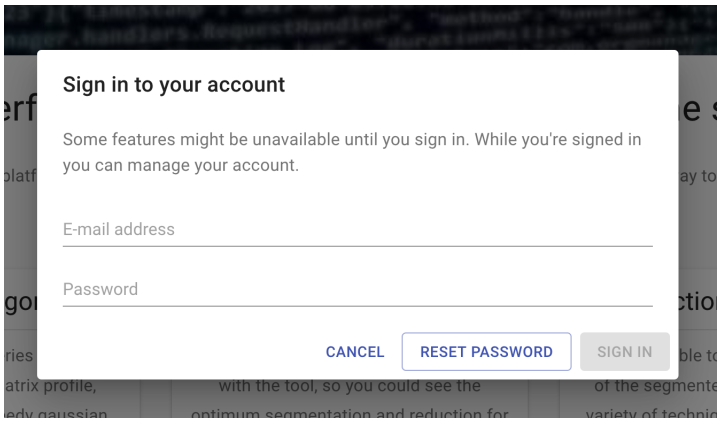


Figura 6.4: Ventana de inicio de sesión de la web.

tana en la que el usuario podrá indicar un email y una contraseña (repetida) con las que registrarse en el sistema. (ver figura 6.5).

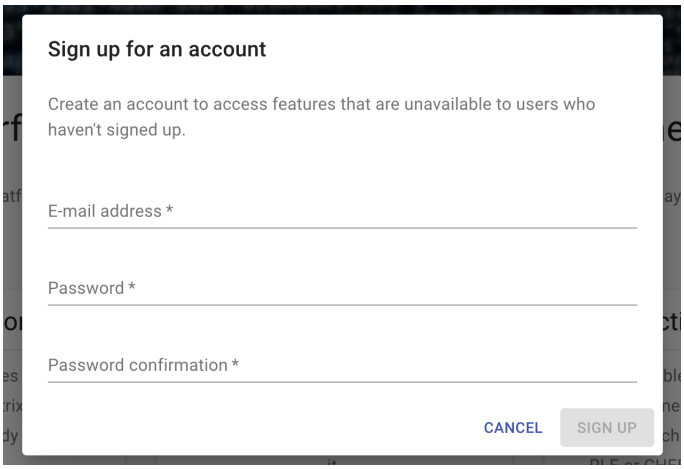


Figura 6.5: Ventana de registro de la web.

Eventos:

1. El usuario accede a la ventana de registro a través del botón *Sign up* de la barra superior.

2. El usuario introduce el usuario y las contraseñas en los campos de texto.

3. EL sistema comprueba dichas credenciales. Si el registro ha sido correcto avisará al usuario y cerrará la ventana, configurando posteriormente en el sistema como usuario conectado. En cambio, si el usuario ya existe informará del error al usuario y mantendrá la ventana abierta para que lo vuelva a intentar.

4. Si las credenciales no son correctas el sistema avisa al usuario del error manteniendo la ventana abierta para que el usuario pueda volver a intentarlo.

Ver información web

Cuando el usuario pulsa en el enlace *About us* del menú desplegable, se presenta la interfaz en la que se muestran el contenido relacionado con la información de la web y su autor. (ver figura 6.6)

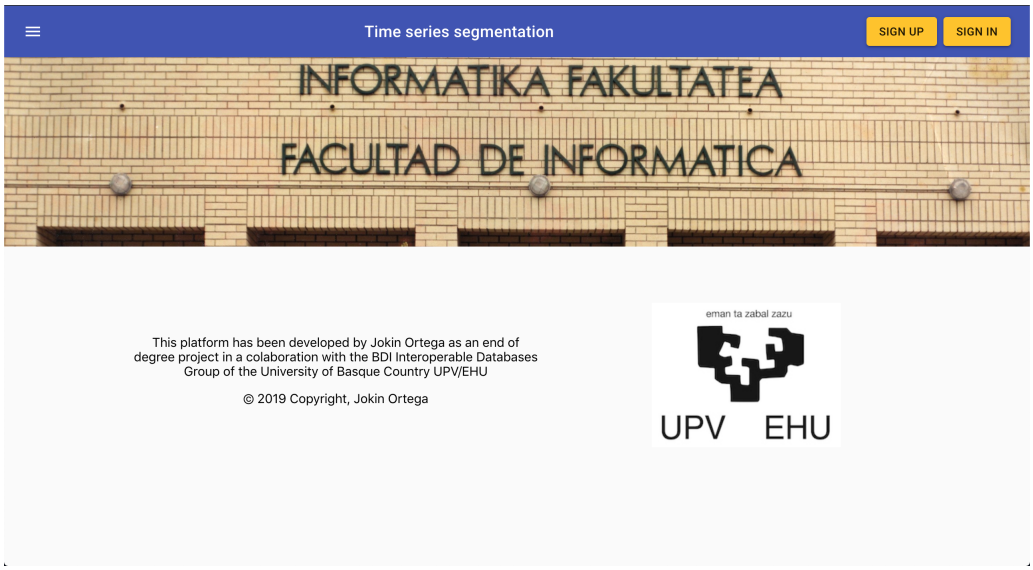


Figura 6.6: Página con la información relativa a la web y al autor.

Consultar información reducción

Cuando el usuario pulsa en el enlace *Reduction info* del menú desplegable o el enlace *see tecchnique details* accesible desde la página inicial se abrirá una interfaz en la que el usuario podrá consultar los datos principales sobre cada uno de los algoritmos de reducción implementados en el sistema. (ver figura 6.7)

Eventos:

1. El usuario accede a través de los enlaces mencionados.

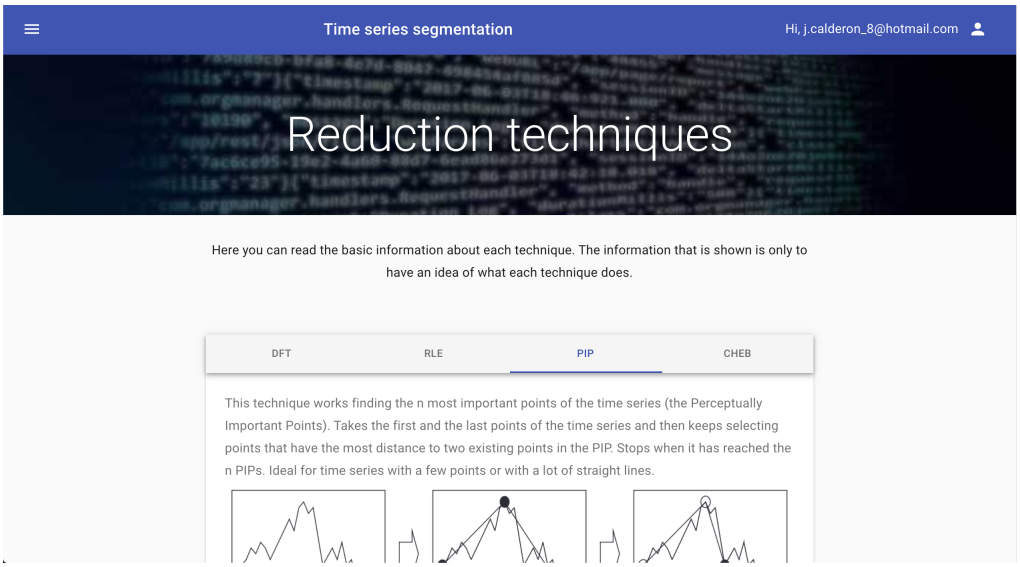


Figura 6.7: Página con la información relativa a las técnicas de reducción.

2. El sistema comprueba si el usuario ha iniciado sesión. Si el usuario no ha iniciado sesión se le redirige a la página inicial mostrándole un mensaje de acceso denegado.

3. Si el usuario está conectado se le mostrará el contenido, donde podrá seleccionar, a través de un panel con pestañas, el algoritmo sobre el que quiere visualizar la información.

Consultar información segmentado

Cuando el usuario pulsa en el enlace *Segmentation info* del menú desplegable o el enlace *see the segmentations details* accesible desde la página inicial se abrirá una interfaz en la que el usuario podrá consultar los datos principales sobre cada uno de los algoritmos de segmentación implementados en el sistema. (ver figura 6.8)

Eventos:

1. El usuario accede a través de los enlaces mencionados.

2. El sistema comprueba si el usuario ha iniciado sesión. Si el usuario no ha iniciado sesión se le redirige a la página inicial mostrándole un mensaje de acceso denegado.

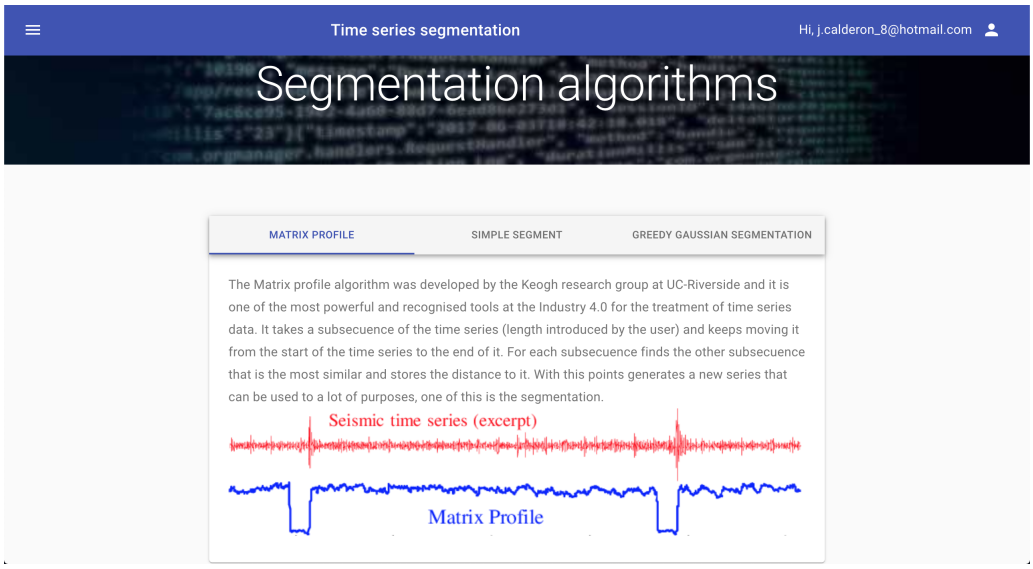


Figura 6.8: Página con la información relativa a los algoritmos de segmentación.

3. Si el usuario está conectado se le mostrará el contenido, donde podrá seleccionar, a través de un panel con pestañas, el algoritmos sobre el que quiere visualizar la información.

Abrir herramienta segmentado

El usuario podrá pulsar el enlace de *Segmentation* del menú desplegable o el botón *Open the tool* para acceder a la herramienta de segmentado de series temporales. En esa herramienta el usuario conectado verificado podrá visualizar una tabla con sus series subidas a la plataforma, trabajar con ellas o subir nuevas series. (ver figura 6.9)

Eventos:

1. El usuario accede a través de los enlaces mencionados a la herramienta.
2. El sistema comprueba si el usuario ha iniciado sesión y, si lo ha hecho, si está verificado. Si el usuario no ha iniciado sesión o no es un usuario verificado se le redirige a la página inicial mostrándole un mensaje de acceso denegado.
3. Si el usuario está conectado y verificado se le mostrará el contenido.

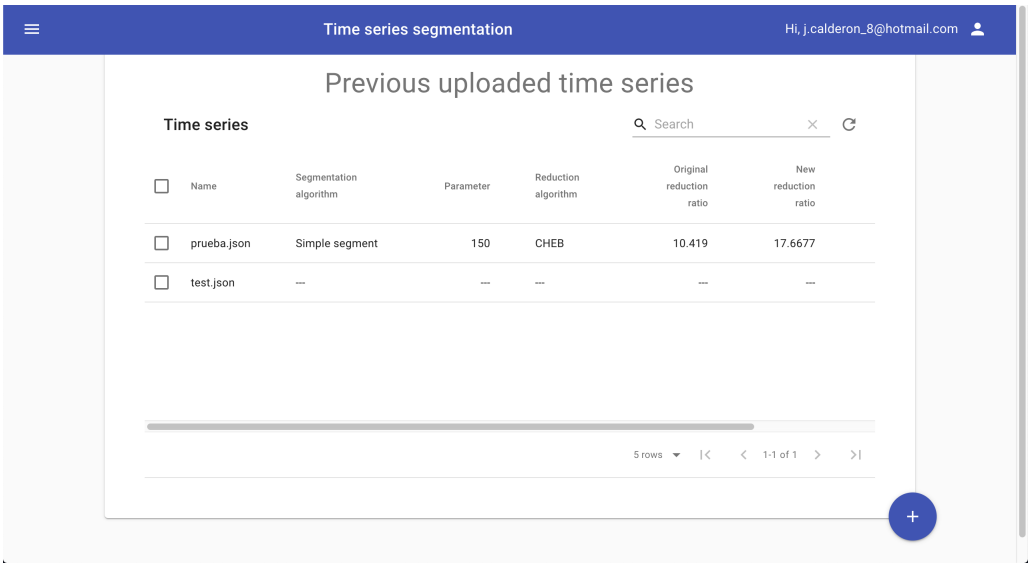


Figura 6.9: Página inicial de la herramienta de segmentado.

4. El usuario podrá realizar tres acciones: subir una nueva serie, segmentar una de las series disponibles o reducir una de las series disponibles. Las primeras dos acciones podrán realizarse en cualquier momento, la tercera solo podrá realizarse si la serie tiene ya asociada un algoritmo de segmentación y un parámetro.

Cerrar sesión

El usuario pulsa el icono del perfil en la parte superior derecha, pulsa el botón de *Sign out* y confirma en la ventana que se le muestra si desea cerrar la sesión o no.

Eventos:

1. El usuario accede a través de los enlaces mencionados y cierra la sesión.
2. El sistema se encarga de eliminar todo el contenido de la sesión que hace referencia al usuario conectado. La aplicación pasa estar en el rol de usuario no conectado.

Ver perfil

El usuario pulsa el icono del perfil en la parte superior derecha y pulsa en la opción de *settings*. Se le abrirá una ventana en la que podrá editar diversos elementos del perfil como el nombre, un avatar, editar colores de la web o consultar ultimo inicio de sesión y el día en el que se registró en el sistema. (ver figura 6.10)

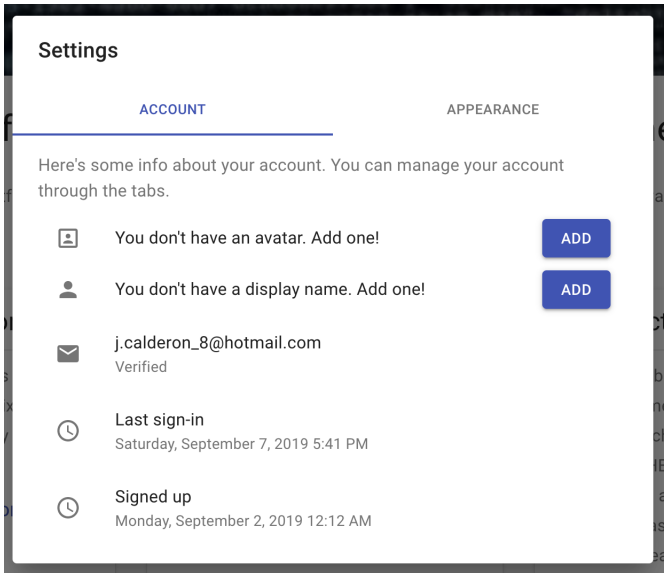


Figura 6.10: Ventana con la información y configuración relativa a la cuenta del usuario.

Eventos:

1. El usuario accede a su perfil a través de los enlaces mencionados.
2. Si el usuario decide modificar algún elemento de su perfil lo hace a través de los campos de texto o botones correspondientes.
3. El sistema verifica los cambios y los confirma o los desestima según el resultado de la verificación.

Segmentar serie

Cuando el usuario está en el caso de uso de abrir herramienta de segmentado y selecciona la opción de segmentar una serie se le abre una nueva interfaz en la que puede ver

tres gráficas diferentes representando la serie seleccionada, una por cada algoritmo de segmentación. (ver figura 6.11)

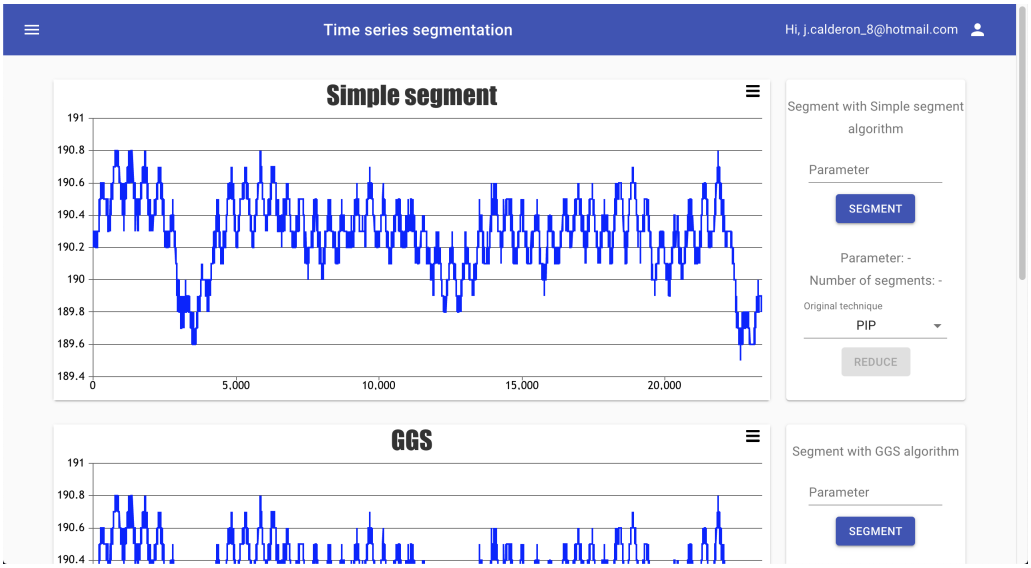


Figura 6.11: Ventana en la que el usuario puede realizar la segmentación de las series.

Eventos:

1. El usuario accede a la página tal y como se ha mencionado.
2. El usuario puede segmentar de manera independiente con cualquiera de los tres algoritmos a través del campo presentado a la derecha de cada gráfico.
3. El sistema actualiza cada gráfico en el que el usuario haga una petición de segmentación, mostrando el resultado obtenido del servicio web.
4. El usuario podrá seleccionar una de las segmentaciones ya terminadas y se pasará al caso de *reducir serie*.

Reducir serie

El usuario accede a través del caso de *abrir herramienta segmentado* o a través de la de *segmentar serie*. El usuario podrá segmentar los distintos segmentos con los diferentes

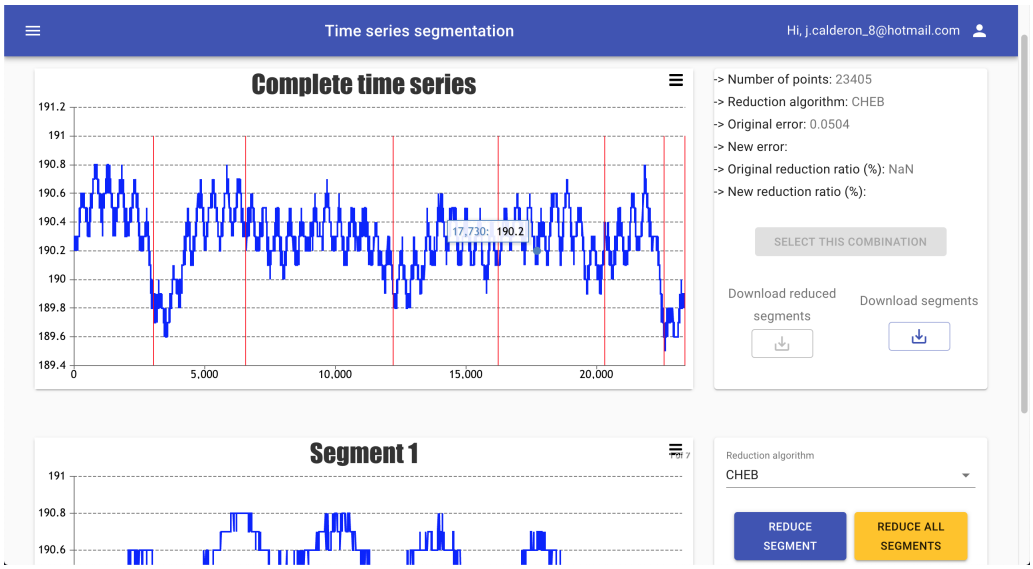


Figura 6.12: Ventana en la que el usuario puede realizar la reducción de la serie y los segmentos.

algoritmos de reducción ofrecidos, pudiendo ver los resultados de la reducción total de manera dinámica. (ver figura 6.12)

Eventos:

1. El usuario accede a la página tal y como se ha mencionado.
2. El usuario selecciona la técnica con la que quiere realizar la reducción.
3. Si el usuario selecciona un segmento y pulsa el botón de *Reduce this segment* el sistema actualizará los datos de la reducción final modificando dicho segmento.
4. Si el usuario pulsa el botón de *Reduce all segments* el sistema actualizará los datos de la reducción final modificando todos los segmentos de la serie.
5. El sistema actualizará el gráfico de la serie completa, mostrando las modificaciones pertinentes de la serie final reconstruida.

7. CAPÍTULO

Plataforma web

Con toda la información recabada con las pruebas realizadas sobre la segmentación y la reducción de series (junto con el añadido de que puede llegar a ser útil la segmentación en ciertos casos), el siguiente paso ha sido el de realizar una plataforma web en la que un ingeniero de datos pueda interactuar con las funciones de segmentación y reducción para que sea capaz obtener resultados y conclusiones por sí mismo sobre la posibilidad de que cierta segmentación sea de utilidad en su contexto.

A continuación se presentan los distintos puntos relacionados con el desarrollo de dicha plataforma:

7.1. API REST de segmentación

Lo primero que se abordó en el desarrollo de la web fue la implementación de un servicio de API REST que permitiese, para cada uno de los distintos algoritmos de segmentación, recibir los datos de una serie temporal junto con el parámetro para el algoritmo y este devolviese los puntos por donde dicha serie debía de ser segmentada. Esto permite separar las implementaciones y las funcionalidades tanto de la web como de la segmentación de series, permitiendo así facilitar el apartado de segmentación en la web a la vez que el servicio de segmentación puede ser independiente de la plataforma y así poder ser utilizado en otros contextos si así se desea.

7.1.1. Diseño y herramientas

Respecto a este servicio, lo primero que se planteó fue la plataforma donde este iba a ser implementado. Para ello se tuvieron en cuenta tanto la facilidad y conveniencia del desarrollo, como el lenguaje en el que los algoritmos de segmentación estaban o podían ser implementados. Por ello, teniendo en cuenta que todos los algoritmos de segmentación se habían implementado en Python (algunos de ellos en otros lenguajes también), era ideal encontrar una plataforma en la que poder explotar dicho lenguaje.

Es por ello que se seleccionó Django para realizar dicha implementación, ya que utiliza Python de forma nativa para construir las páginas web y que, junto con el ya mencionado Django-Rest-Framework, nos permitía implementar de manera sencilla nuestro servicio de segmentación de series. Además, como dato importante sobre la utilización de esta tecnología, esto nos ha permitido poder integrar dicho servicio en la plataforma que tiene el grupo BDI implementada en Django para sus servicios REST de extracción de características, reducción y reconstrucción de series.

Una vez fijada la herramienta, el siguiente paso era el de plantearse el diseño del servicio, fijando tanto el formato de los valores de entrada como el formato de los valores de salida. Este es un ejemplo con el formato de la petición y de la respuesta:

Petición:

```
{
    "value":[2,34,34.5,3,44,3],
    "parameter": 10
}
```

Respuesta:

```
{
    "value":[1,3,5]
}
```

7.1.2. Configuración y desarrollo

Siguiendo la guía presentada en la página web del django-rest-framework[1], se ha creado la estructura necesaria para poder desarrollar y montar el servicio[2].

En la figura 7.1 podemos ver el esqueleto de la aplicación Django creada. Debido a que en este caso el objetivo final no es el de construir una página web completa utilizando Django, se descarta comentar todos los ficheros y carpetas que tiene el proyecto, centrándonos simplemente en los que se consideran necesarios e indispensables de conocer para poder poner en marcha el servicio de segmentación de series. Estos serían:

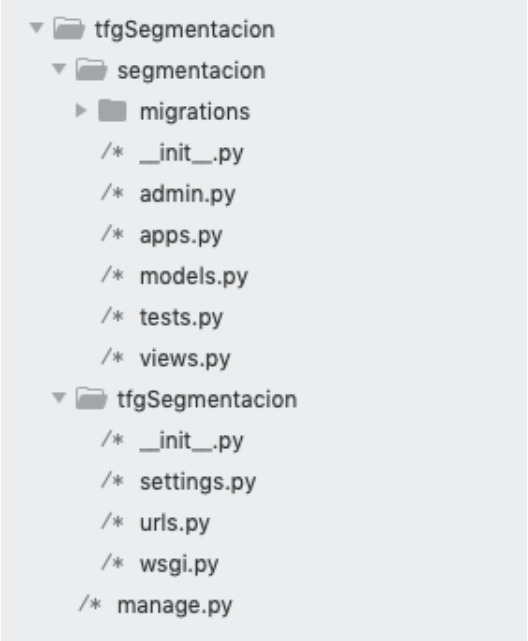


Figura 7.1: Estructura inicial del proyecto de Django.

- **tfgSegmentacion.** Carpeta que engloba todo lo referente a la configuración del proyecto.
- **segmentacion.** Carpeta que contiene todos los ficheros referentes a la aplicación que hemos generado (un proyecto puede tener y utilizar múltiples aplicaciones diferentes).

[1] Tutorial Django REST framework: https://www.django-rest-framework.org/tutorial/quickstart/

[2] Si se desea consultar el proceso seguido para la instalación de la estructura inicial de django consultar el anexo B.1

- **views.py**. Aquí se determinan los métodos o vistas que queremos implementar en nuestra aplicación. En este caso crearemos un método por cada algoritmo de segmentación que tenemos en nuestro servicio. En la figura 7.2 podemos ver un ejemplo, en la que recibimos la entrada (podemos ver si la petición es GET o POST, siendo en nuestro caso POST ya que recibimos por ahí los datos) y la procesamos con el método de segmentación del Simple segment, que es fácilmente integrable ya que está también escrito en Python. El resultado final obtenido de la ejecución del algoritmo lo devolvemos como una JSONresponse, que es lo que nos permite devolver la petición como una respuesta de API REST.

- **settings.py**. En este fichero se configuran los distintos elementos del proyecto, como por ejemplo aplicaciones a utilizar en el proyecto, directorios, bases de datos... pero en este caso solo tendremos que configurar la variable installed-apps, a la que tendremos que añadir, aparte de las que ya tenga, las siguientes líneas:

```
NSTALLED_APPS = (
    'rest_framework',
    'segmentacion',
)
```

  Con esto lo que hemos hecho es indicar al proyecto que tiene que utilizar dichos módulos o aplicaciones al ejecutarse.

- **urls.py**. En este fichero se configuran las URL con las que se podrán acceder a nuestros distintos servicios. En ellos se indicará el formato de la URL y se enlazará cada patrón con el método de los creados en views.py que queremos que se ejecute. En la figura 7.3 vemos un ejemplo en el que, al indicarle la ruta direccionIP/series/-segmentar/simple ejecutará el método que hemos definido en views.py y que hemos visto en la figura 7.2.

- **manage.py**. Este fichero lo ejecutaremos para arrancar nuestro servicio y dejarlo activo escuchando peticiones.

Una vez hemos realizado estas configuraciones lo que haremos será arrancar la aplicación en local y que se quede a la espera de peticiones y poder comprobar su funcionamiento:

```
./manage.py runserver
```

```
def segmentar_serie_simple(request):
    if request.method == 'POST':
        json = JSONParser().parse(request)
        data = json['value']
        parameter = json['parameter']
        resp = segmentacion.segmentar(data,parameter)
        return JSONResponse(resp)
```

Figura 7.2: Método de ejemplo del fichero views.py que recibe la petición y devuelve el resultado de la segmentación.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^series/segmentar/simple', views.segmentar_serie_simple),
]
```

Figura 7.3: Ejemplo de configuración del fichero urls.py.

7.1.3. Integración

Una vez hemos hecho todas las pruebas en local, el siguiente paso ha sido poner el servicio accesible online para poder utilizarlo en todas las pruebas posteriores con la página web o con cualquier prueba que requiera segmentación. Para ello lo primero que se ha hecho ha sido dejar el proyecto accesible en una máquina virtual creada en la plataforma de Google Cloud en el apartado de Compute engine[3].

Se ha creado una máquina básica con el Sistema Operativo Debian sin entorno gráfico, una instalación que está entre las opciones que Compute engine ofrece. Con trabajar en el sistema a través de consola nos ha sido suficiente para poner en marcha el servicio. Para ello lo que tenemos que hacer es instalar en la máquina tanto Python como Django y Django REST framework de la misma manera que lo teníamos en la máquina de desarrollo.

Una vez hecho eso copiamos el proyecto a la máquina nueva y ejecutamos el mismo comando que hemos ejecutado en local para arrancar el proyecto, pero con una pequeña modificación:

./manage.py runserver 0.0.0.0

Le hemos añadido como parámetro esa dirección 0.0.0.0 que indica al programa por qué

[3]Compute engine: https://cloud.google.com/compute/?hl=es

direcciones tiene que escuchar las peticiones. Si lo dejamos como antes, sin indicarle ningún tipo de dirección, lo que hará será escuchar simplemente peticiones que se le hagan desde la dirección de localhost.

En las últimas fases de la finalización del desarrollo se decidió proceder a la integración del servicio implementado, pasándolo esta vez a la máquina que tiene online, también en Google Cloud, el grupo BDI. Ellos implementan unos servicios de tratamientos de series: Extracción de características, reducción, reconstrucción... La idea es poder aumentar la capacidad de dicho servicio web, y que integre la opción de segmentación de series con cualquiera de los tres algoritmos que hemos implementado y que pueda ser explotada por cualquier necesidad futura del grupo.

Es en este momento en el que el beneficio de haber desarrollado el nuevo servicio en Django se ve reflejado, ya que la integración no puede ser más sencilla. Como toda la estructura de Django y el proyecto ya está creado en la implementación del grupo BDI, los pasos que debemos dar tienen más relación con el contenido del proyecto.

Debemos copiar todos los ficheros relacionados con los algoritmos de segmentación (los ficheros donde estén implementados los algoritmos como tal y todos sus ficheros necesarios, todos externos al esqueleto de Django). Una vez los tenemos solo tenemos que añadir al fichero views.py los métodos que ya teníamos creados en nuestra implementación y en el fichero urls.py configurar las url con las que se podrá acceder a dichos métodos. Una vez hemos hecho esto ya tenemos integrado en la implementación del grupo BDI el servicio de segmentación desarrollado.

Este servicio y todos los mencionados están accesibles mientras el BDI mantenga la máquina en marcha con la sintaxis

```
http://35.205.214.223/api/[METODO]/
```

y se puede comprobar su funcionamiento con el enlace http://35.205.214.223/api/prueba/, recibiendo respuesta si el servicio está activo.

7.2. Página web

A continuación se presenta el proceso para llevar a cabo de la plataforma web, tanto en el apartado del diseño como en el apartado del desarrollo.

7.2.1. Diseño

Estructura del sistema web

Tal y como se han mencionado en el apartado 3.4 sobre las herramientas, la web estará basada en React, complementado con el paquete material-ui[4] para lograr una estética Material design[5] y conectado con la plataforma de Firebase de Google para almacenamiento y gestión de usuarios. Utilizará también el servicio API REST ofrecido por el grupo BDI para la realización de las reducciones de series temporales, así como el servicio desarrollado que se ha introducido en el apartado 7.1 para realizar las segmentaciones.

Dentro de la utilización de Firebase, el almacenamiento de ficheros se realizará tanto con el módulo de Firestore como con el de Storage, según el tipo de elemento que se almacene. La gestión de usuarios se realizara con el módulo de Authentication, identificando a los usuarios con un correo electrónico y una contraseña.

Estructura de la Base de Datos

En este caso, tal y como hemos dicho, utilizaremos tanto Firestore como Storage para almacenar las series temporales de los usuarios junto con sus características y datos relevantes. La idea inicial era la de almacenar toda la información en Firestore, pero al analizar la cantidad de valores que se tendrían que almacenar se llegó a la conclusión de que eso no sería viable así que se decidió añadir el módulo del Storage.

Por tanto, lo que finalmente se ha diseñado es un esquema en el que en Firestore se guardan los atributos de la serie como el nombre, algoritmo con el que se ha segmentado, parámetro utilizado... mientras que en Storage se almacenarán los ficheros que contengan los valores de las series temporales, que son los de mayor tamaño. En las figuras 7.4 y 7.5 se puede ver la estructura general de la BD de Firestore, es decir, la manera en la que están organizadas las colecciones y los documentos, siendo los elementos de color azul las correspondientes a las colecciones, los elementos de color rojo a los documentos y los

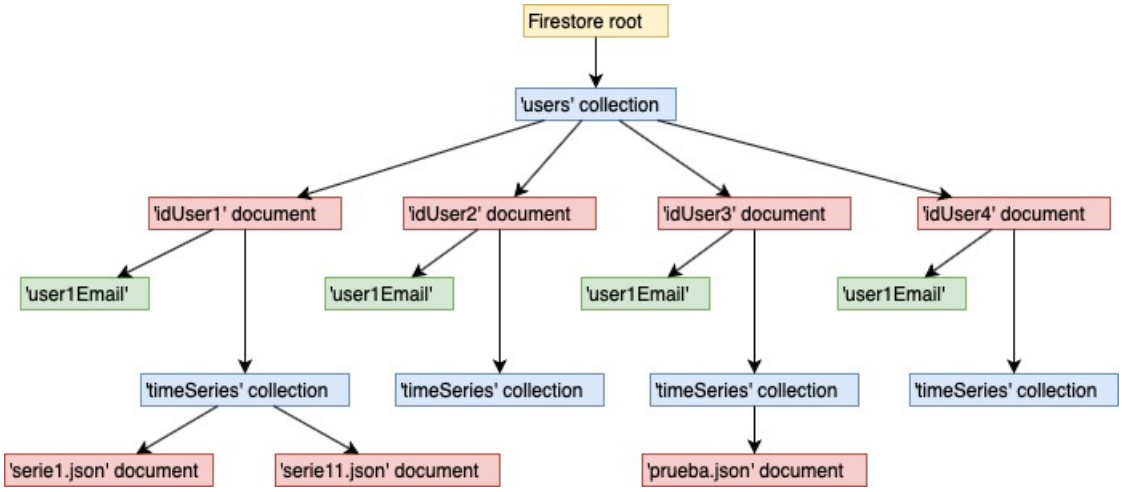


Figura 7.4: Parte de la estructura de la BD de Firestore, concretamente la que se centra en los ficheros de los usuarios.

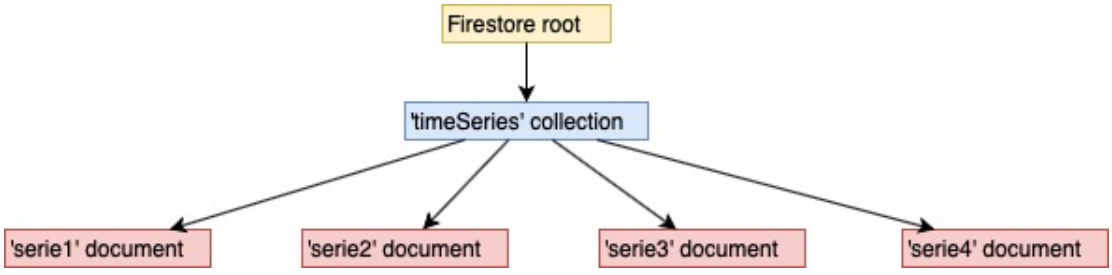


Figura 7.5: Parte de la estructura de la BD de Firestore, concretamente la que se centra en la colección de series que se mostrarán en el apartado de casos de éxito.

elementos de color verde a campos básicos de datos.

Concretamente en la figura 7.4 vemos la estructura referente a la colección donde almacenamos los correos de los usuarios junto con sus series almacenadas. Estos documentos con el id de los usuarios se van generando automáticamente a medida que los usuarios se van registrando en el sistema, utilizando el id generado por el módulo de Authentication.

En la figura 7.5 se encuentra la estructura de la colección que contiene las series temporales, introducidas manualmente en el desarrollo, que servirán para el apartado de series temporales de éxito en la segmentación.

En el último nivel de dicha estructura, tanto de la parte de featuredSeries como de la parte de users, tendríamos lo que son las propias series temporales que están almacenadas en la Base de Datos. Estas series tienen el mismo formato en ambos casos, ya que puede que lleguen a tener la misma utilidad o que se necesite la misma información sobre ellas en la web. A continuación podemos ver la estructura del documento y las propiedades que contiene:

```
{
    "name": "prueba.json",
    "newError": 0.02,
    "newReductionRatio": 10.23,
    "originalError": 0.0043,
    "originalReductionRatio": 20.74,
    "parameter": 100,
    "reconsValues": "fbOsmM0CmH43/prueba_rec.json",
    "reductionAlg": "PIP",
    "segmentationAlgorithm": "Simple segment",
    "segments": [{"segment: 3305, "technique": "PIP"}, ...],
    "values": "fbOsmM0CmH43/prueba.json"
}
```

Quizás los elementos menos triviales de entender que tendríamos ahí serían los de **values** y **reconsValues**. El contenido que tienen ambos tiene el mismo formato y es el de la ruta que tiene en el almacenamiento de Storage el fichero con los valores de la serie temporal, tanto el fichero original (values) como el fichero reconstruido después de segmentar (re-

[4]Material-ui: https://material-ui.com
[5]Material design: https://material.io/design/

consValues). De esta manera se ha conseguido complementar la utilización de Firestore y Storage, pudiendo almacenar por un lado los datos de la serie y por otro enlazar al fichero de la serie en sí.

7.2.2. Desarrollo

Tal y como ya se ha comentado en el apartado 3.4, el elemento principal del desarrollo de la plataforma ha sido React.js, ya que ha sido con la que se ha construido toda la parte de la interfaz (junto con el ya mencionado Material-ui) y con la que se han manejado los elementos relacionados con la lógica de la aplicación.

En esta sección se tratará de plasmar el proceso que se ha llevado a cabo durante el desarrollo de la aplicación web. Debido a que una explicación punto a punto o demasiado detallada del código sería excesiva para este documento, se tratarán de plasmar los distintos puntos clave básicos que permitan entender cómo ha sido la implementación en cada una de las secciones de la web.

A continuación se detallan dichos puntos realizados para la creación y puesta en marcha de la aplicación, complementado con los conceptos que se consideran de utilidad sobre la tecnología utilizada en cada momento.

Creación del esqueleto de la web

Tal y como se recomienda en el tutorial oficial de la web de React[6], la idea era la de crear la aplicación utilizando el entorno de *Create React App*[7] ya que nos genera todo el entorno para poder implementar una web con React.

Create React App te permite crear desde cero una web tipo Single Page App ofreciendo un entorno con todas las herramientas, ficheros, estructura y paquetes necesarios.

En este caso nos ofrece la posibilidad de trabajar tanto con *yarn*[8] como con *npm*[9]. En este caso se ha utilizado la opción de *npm* aunque con cualquiera de las dos se podría trabajar sin ningún problema.

[6]Tutorial React: https://es.reactjs.org/docs/create-a-new-react-app.html
[7]Create React App: https://github.com/facebook/create-react-app
[8]yarn: https://yarnpkg.com/lang/en/
[9]npm: https://www.npmjs.com

Antes de haber comenzado a desarrollar la web utilizando *Create React App*, se buscó la manera de integrar el Material Design de Google en una aplicación de React. Como resultado de dicha búsqueda se encontró Material-ui, un paquete para React que nos ofrecía una gran cantidad de componentes y vistas con diseño tipo Material design fácilmente integrables en React y que era ampliamente utilizada y sobre la que había muy buena documentación.

Al leer la documentación de Material-ui, se encontró un apartado de proyectos de ejemplo[10] en el que se ofrecen ciertos proyectos de base, que contienen ciertas funcionalidades base de las que poder partir en la realización de nuestro proyecto. Entre los ejemplos se encuentra el llamado *React + Material-UI + Firebase*[11], y que resultaba ideal para comenzar nuestro proyecto debido a que contenía 3 de los elementos clave que la aplicación web iba a necesitar.

React + Material-UI + Firebase está basado en *Create React App*, sobre lo que posteriormente se instalan los módulos de Material-ui, React Router y Firebase, permitiéndonos así arrancar con una base sólida en la que ya tenemos configurado el entorno de los principales pilares de nuestra aplicación. Partiendo de la base configuarada por *Create React App*, esto es lo que nos ofrecen los otros módulos instalados:

- **React Router**[12]**:** Nos ofrece un manejo del enrutamiento dentro de nuestra web de manera sencilla y potente, en la que podemos manejar tanto los enrutamientos a través de las direcciones url o incluso pudiendo enrutar a una nueva página dentro de otra a través de código Javascript. Nos permite incluso poder mantener propiedades de la página anterior y utilizarlos en la página actual. En esta configuración inicial viene implementado un enrutamiento de ejemplo básico, en el que está configurada una sola url que muestra cierto contenido si la url solicitada coincide.

- **Firebase:** En este caso viene implementada la gestión de los usuarios del sistema utilizando el componente Auth de Firebase. Una vez configurado Firebase con las credenciales del proyecto, es capaz de gestionar el registro y el inicio de sesión en la aplicación, almacenando los usuarios creados en Firebase.

- **Material-ui:** Todas estas funcionalidades descritas vienen implementadas en una interfaz base que ha sido creada utilizando Material-ui, en el que tenemos una barra superior que se mantiene fija en todas las páginas de la aplicación con los botones

[10] Ejemplos material-ui: https://material-ui.com/es/getting-started/example-projects/
[11] React + Material-UI + Firebase: https://github.com/Phoqe/react-material-ui-firebase
[12] React Router: https://reacttraining.com/react-router/web/guides/quick-start

de gestión del inicio de sesión y el registro, o con el botón de gestión de perfil, según sea el caso.

Lo interesante de todo esto es la transparencia que ofrece al usuario en la instalación y configuración de dichos elementos , ya sean todos los que *Create React App* de base configura como los que posteriormente obtenemos con *React + Material-UI + Firebase*.

Para poder instalar este entorno y comenzar con el desarrollo de la aplicación tenemos que clonar el proyecto al que le daremos el nombre de tfg-web-react:

git clone https://github.com/Phoqe/react-material-ui-firebase.git tfg-web-react

Esto nos clonará la estructura de carpetas del proyecto, que podemos ver en la figura 7.6. Dentro de dichos ficheros, estos son los más destacables a explicar:

- **public.** Correspondiente a la instalación básica de React, contiene el fichero principal de la web, el index.html, en el que se introducen los elementos html/css necesarios para la ejecución de la web y donde se irán metiendo los componentes o elementos que en cada momento tengan que ser mostrados.
- **src.** Carpeta que engloba todo el contenido del proyecto: componentes, ajustes, ficheros... En ella encontramos el fichero index.js, que será el punto de entrada del código JavaScript de nuestra web y que en este caso está configurado para renderizar el componente App.js de inicio. En esta carpeta nos encontramos tambien con el fichero index.css que actúa como fichero css principal de la aplicación, aunque en este caso al utilizar Material-ui la gestión del css se hace de otra manera. En esta carpeta se puede guardar cualquier tipo de fichero JavaScript o CSS que necesitemos para nuestra aplicación.
- **src/App.** Correspondiente a la instalación básica de React, en esta carpeta tenemos el fichero App.js, que contiene el componente principal de nuestra aplicación y que es el primero en ser renderizado por el index.js. En este caso además es donde está implementada la gestión del registro y de los usuarios logueados.
- **src/content-dialogs-layout-tabs.** En estas carpetas está todo el contenido en lo referente a los distintos componentes que utiliza la aplicación. En content y layout se almacenan los componentes referentes a las páginas completas de la aplicación (en content elementos más referentes a la gestión del contenido y en layout elementos

más centrados en la vista en sí, que reciben los elementos desde los componentes de contents y los muestran) y en dialogs y tabs se organizan los componentes que son diálogos del sistema o elementos como la barra superior de la web.

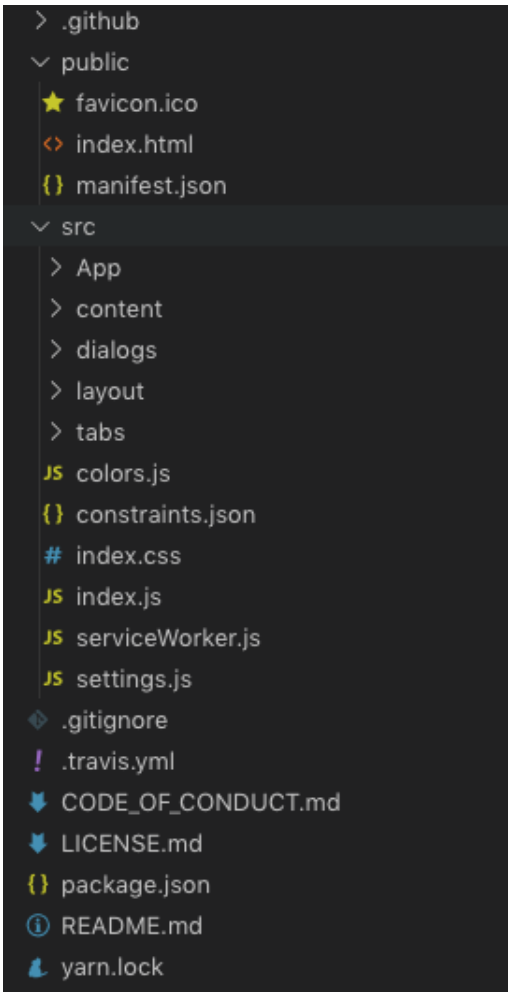


Figura 7.6: Estructura de carpetas y ficheros creada con el *React* + *Material-UI* + *Firebase*.

Lo siguiente que debemos de hacer una vez que tenemos la estructura será la de instalar todas las dependencias de paquetes que la aplicación requiere, y que nos vienen ya configuradas en el fichero package.json. Lo único que tenemos que hacer será situarnos en la carpeta del proyecto y ejecutar el siguiente comando:

npm install

Esto lo que hará será leer el fichero package.json que ya hemos mencionado e irá de uno en uno descargando todos los paquetes de la lista desde los repositorios de *npm*.

Al terminar se nos generará una carpeta llamada **node_modules** donde estarán todos los ficheros descargados e instalados por *npm*.

Con esto ya estaría el proyecto en perfectas condiciones para ponerlo en marcha y continuar con el desarrollo, para esto el entorno *Create React App* nos ofrece el siguiente comando:

npm start

Ejecutando este comando en la carpeta del proyecto se pone en marcha la aplicación en modo desarrollo y estará accesible desde el navegador en la siguiente dirección:

http://localhost:3000

Lo útil que tiene además trabajar con este sistema es que todos los cambios que se vayan realizando a través del editor de código en los ficheros de la carpeta src serán automáticamente recargados y mostrados en la página del navegador. Esto permite un desarrollo más fluido y con una visualización de cambios muy rápida y productiva.

Características principales del desarrollo con React

Para poder comprender la manera en la que el desarrollo ha sido llevado a cabo, es conveniente presentar los elementos distintivos del desarrollo con la tecnología React.

React basa su funcionamiento en la implementación de unas clases a las que se les llama componentes. Dichos componentes representan un objeto o un elemento que se va a presentar en la interfaz y que reúne unas ciertas propiedades y elementos característicos. Esto permite ir mostrando de manera dinámica distintos contenidos según el estado o las condiciones en las que se encuentre la aplicación.

Estos componentes reciben como entrada ciertos parámetros a los que se les llama *props* y los utilizan o los tratan de la manera que se desee para finalmente, a través del método llamado render, mostrar en pantalla el contenido deseado. Para poder llevar a cabo el uso de estos componentes de manera potente, se ayudan de un lenguaje llamado JSX[13], una

[13]JSX: https://es.reactjs.org/docs/introducing-jsx.html

extensión de la sintaxis de JavaScript desarrollado para describir la interfaz de usuario que se debe mostrar.

Este lenguaje permite la generación de elementos, que no son mas que la descripción de lo que quieres que se muestre en pantalla (una etiqueta HTML, otro componente de React...). La idea general de este lenguaje es la de evitar la separación del maquetado y la lógica de una página, juntándolas en un mismo objeto (componente) que maneje ambas.

La sintaxis es una sintaxis muy similar a la de un lenguaje de plantillas o de etiquetas pero en realidad es mucho más potente que todo eso ya que permite crear tanto elementos React como objetos y elementos de JavaScript, pudiendo todos ellos almacenarse en variables y pasarse como parámetro entre componentes. Es un lenguaje similar a la programación declarativa[14] pero que contiene toda la potencia del lenguaje JavaScript. Esto se debe a que dento del JSX podemos poner cualquier código escrito en JavaScript si lo colocamos entre dos llaves.

En la figura 7.7 podemos ver un ejemplo de un componente utilizado en el proyecto, que ha sido simplificado para entender el funcionamiento básico. En este caso recibe unas *props*, que son el nombre de la clase que va a tener la etiqueta y texto del contenido que se va a mostrar, y devuelve un párrafo que utiliza dichos parámetros de entrada recibidos. En este caso es un ejemplo bastante simple, que obviamente se complica en mayor medida en otros componentes del proyecto.

```
class AboutUsContent extends Component {
    render() {
        const { divClass } = this.props;
        const { description } = this.props;

        return (
            <div className={divClass}>
                <p>{description}</p>
            </div>
        )

    }
}
```

Figura 7.7: Ejemplo de un componente simple que recibe propiedades y las utiliza en el resultado a mostrar.

A parte de las propiedades de entrada, los componentes pueden disponer de un estado,

[14]Programación funcional: Mirar igual referencia a publicación

que repercute en el contenido o en el modo que se va a renderizar el elemento. Este estado se almacena como un objeto dentro de la propia clase en una propiedad llamada state. Dentro de ese objeto podrá crearse una nueva propiedad por cada elemento del estado que queramos manejar.

Este estado puede ser modificado en cualquier momento en cualquier método de la clase, con lo que puede ser modificado en mitad de la ejecución, permitiendo alteraciones que hacen modificar el contenido de la vista de manera automática. Esta modificación se hace a través del método *setState()*, en el que indicamos que propiedad del estado queremos modificar (pueden ser más de una a la vez) y el valor nuevo que queremos asignarle. Eso si, si queremos utilizar alguna de las propiedades en el renderizado tenemos primero que definir un valor por defecto en la constructora de la clase ya que si no al renderizarse el componente intentaría acceder a un elemento que no existe.

Esto nos abre muchas posibilidades porque en el momento en el que nosotros alteramos el estado del componente, este se vuelve a renderizar automáticamente. Por ejemplo, tal y como podemos ver en el ejemplo de la figura 7.8, podemos asignar al contenido de un elemento que vamos a mostrar en la vista un atributo del estado del componente. Hemos definido en la constructora el valor inicial de la propiedad y cuando el valor del estado cambie al pulsar el botón de la interfaz, el contenido de la etiqueta <Typografy>cambiará y se actualizará de manera automática al nuevo valor del estado. En este caso demuestra, de manera muy simplificada, actualizaciones que se realizan en la web de etiquetas en las que se muestran valores que son resultado de las operaciones de reducción o segmentación realizadas.

Este caso es uno de los ejemplos más simples que podemos ver del uso del estado de un componente, pero podemos realizar otras acciones algo más complejas. Por ejemplo, como vemos en la figura 7.9, podemos utilizar el valor de un estado para determinar directamente el componente que vamos a renderizar, no como en el caso anterior que determinaba el valor que tenia cierto componente. En este caso, como vemos, el valor del estado al renderizar el componente determina hacia que “camino“ va a ir nuestra vista final. Si el valor del estado es el inicial y no se ha cambiado, en pantalla se nos mostrará un logo que nos indica que la página aún está cargando. Si el estado cambia porque el componente ya ha terminado de realizar los cálculos o ejecuciones necesarias, la página se vuelve a renderizar y ahora nos muestra otro elemento diferente, en este caso sería el contenido original de la página.

En este mismo ejemplo se ha introducido también otro de los elementos diferenciadores

```
class ReductionPage extends Component {

    constructor(props) {
        super(props);

        this.state = {
          newError: "-"
        };
    }

    onClick = () => {
        var newValue = "Some value after calculation";
        this.setState({ newError: newValue });
    }

    render() {

        const { buttonClass } = this.props;

        return (
            <div>
                <Typography color="textSecondary" variant="h4" align="center" >New error: {this.state.newError}</Typography>
                <Button color="primary" className={buttonClass} variant="contained" onClick={this.onClick}>Reduce</Button>
            </div>
        )

    }
}
```

Figura 7.8: Ejemplo de la utilización del estado de un componente para cambiar el valor mostrado por un texto en pantalla de manera dinámica.

de un componente de React, que es el del ciclo de vida del componente. El componente, la primera vez que es renderizado por primera vez pasa por el estado *componentWillMount*, en cuanto está montado pasa por el *componentDidMount* y cuando el componente deja de ser renderizado pasa por el estado de *componentWillUnmount*. En este caso utilizamos el método *componentDidMount()* para recoger una única vez los datos que queremos mostrar en la interfaz y que así no esté realizando dicha obtención de los datos cada vez que se renderiza el componente por cualquier cambio de estado.

Como último apunte respecto a React, puede darse el caso en el que la renderización causada por un cambio de estado nos lleve a problemas de rendimiento con la web. Esto ha ocurrido en nuestro caso particular ya que teníamos, por ejemplo, dos gráficas en una interfaz mostrando unos datos que tenía que procesar previo a dibujar la gráfica. El problema está en que hay un campo de texto en el mismo componente que cambia un estado cada vez que el campo es modificado (inserción o adición de texto) y por tanto hacía que dichos gráficos se tuviesen que volver a dibujar, junto con los cálculos previos que conllevaban.

Para tratar de evitar casos como estos, React ofrece en cada componente un método llamado *shouldComponentUpdate()*. Este método se ejecuta cada vez que un componente va a ser renderizado y si no se hace un “override“ del método por defecto la respuesta será

```
class GraphPage extends Component {

    constructor(props) {
        super(props);

        this.state = {
          loading: true
        };
    }

    onClick = () => {
        var newValue = "Some value after calculation";
        this.setState({ newError: newValue });
    }

    componentDidMount = () => {
        var data = ""; //Realizar las acciones necesarias para obtener los datos a mostrar
        this.setState({ loading: false });
    }

    render() {

        if (this.state.loading) {
            return <CircularProgress />
        }

        return (
            <div>
                Contenido o componentes a mostrar en la página
            </div>
        )

    }
}
```

Figura 7.9: Ejemplo de la utilización del estado del componente y de los elementos del ciclo de vida.

```
class Grafico extends Component {

    shouldComponentUpdate(nextProps) {
        if (nextProps.options.data.length === 1) {
            return false;
        } else {
            return true;
        }
    }

    render() {

        const { options } = this.props;

        return (
        <CanvasJSChart options={options} />
      );
    }
  }
```

Figura 7.10: Ejemplo de la utilización *shouldComponentUpdate* para evitar renderizar componentes sin necesidad real de hacerlo.

true y la renderización siempre se realizará. En este caso en el ejemplo de la figura 7.10 podemos ver como ha sido solucionado esto en el componente que renderiza la gráfica. Lo que se hace es, dentro del método *shouldComponentUpdate()*, analizar ciertos elementos para decidir en qué casos el componente debe renderizarse en en qué otros no.

En este caso vemos este componente que tendría que devolver el gráfico, que si los datos que recibe como parámetro no han variado de tamaño desde la anterior vez se considera que no tiene que renderizarse y por lo tanto se devuleve un false. Por tanto en ese caso no se renderizará pero en el resto de casos si, así se ha pasado de un delay de hasta un segundo a una introducción de números en el campo de texto instantánea.

El resto de vistas e interfaces de la web han sido realizadas partiendo de estas bases, aunque como es lógico en muchos casos acaban siendo algo más complejas que las que aquí se han comentado. Junto con estas bases se integran otros elementos como los que se explican en los siguientes puntos de este apartado, permitiendo así obtener una aplicación rica en funcionalidades complejas e intuitivas, útiles para lograr su objetivo final.

Enrutamiento de páginas con React Router

React Router es un paquete para React que permite manejar el enrutamiento de la web

de una manera sencilla y potente. El componente principal y básico para realizar el enrutamiento es el de *Route*. Como vemos en la figura 7.11 se define un elemento Route por cada dirección a la que se quiere dirigir y se indica el path (lo que viene a ser la url que se pedirá al querer navegar) y qué componente se tiene que renderizar al cargar dicha url. La lista de elementos Route definidos será comprbada cada vez que se realice una de las siguientes acciones, si coincide con una de ellas será renderizado dicho componente y si no se le renderizará el componente definido para indicar una página no encontrada:

```
<Route path="/" exact render={() => (<HomeContent isSignedIn={isSignedIn} title={settings.title} store={firestore} storage={storageRef} />)} />
<Route path="/about-us" exact render={() => (<AboutUsContent />)} />
```

Figura 7.11: Dos de los elementos Route definidos en el menú de la web.

- **Atributo *to*:** Tal y como se ve en la figura 7.12, en un enlace podemos colocar el atributo *to* e indicarle el nombre de la url a la que queremos redirigirnos. Esto hará que cuando se pulse en el enlace busque en la lista de elementos de Route que tenemos definido a ver si alguno coincide. En este caso este método ha sido el utilizado para la creación del menú lateral de la aplicación, definiendo cadaelemento con un enlace con este atributo definido.

```
<Link to="/segmentation-info" color="inherit" style={{ textDecoration: 'none', padding: '10px', color: '#000000' }} className={classes.Link}>
  Segmentation info
</Link>
```

Figura 7.12: Ejemplo de la utilización del atributo *to* en un link del menú de la web.

- **Redirect**: Este elemento nos permite ejecutar una redirección sin tener que hacerlo a través del atributo *to* de algún enlace. Nos ofrece otro método distinto de redirección que podemos colocar en métodos o funciones de la aplicación y redireccionar en casos en los que el flujo de la aplicación sea la que necesite del direccionamiento y no sea causado, tal y como hemos dicho, por la pulsación de un enlace. A diferencia de utilizar un enlace o el history, cuando realizamos la redirección con este método el navegador no almacena la página anterior y por tanto, al pulsar el botón de página previa, esta página no será la página desde la que hemos ejecutado el Redirect si no que será la anterior que el navegador tenía almacenada. Se puede ver en la figura 7.13.

- **History push:** Al indicar a un componente que trabaje con React Router, el componente pasa a tener acceso al atributo history del navegador y puede utilizarlo

para introducir un nuevo componente al mismo. Esto nos permite, a través de código JavaScript, ejecutar una redirección y además aprovechar y pasarle al siguiente componente unas propiedades que necesitemos que reciba. Esto ha sido utilizado en la página para determinados momentos en los que la ejecución de ciertas consultas y cálculos determinaban el momento de la redirección, necesitando el siguiente componente algunos de dichos resultados. Esto se puede ver en el ejemplo de la figura 7.14, en la que se necesitaba esperar a calcular valores relacionados con los segmentos de la serie para posteriormente dirigirse a otra página que necesitaba de los valores resultantes de dichos cálculos. Para recoger los valores enviados en el componente de destino lo único que tenemos que hacer es acceder al objeto location.state dentro de las propiedades del componente: *this.props.location.state*. Dentro de este objeto estarán todos los valores que el componente anterior le ha enviado.

- **Barra del navegador:** Como es lógico, cada vez que se inserte una ruta en la barra de direcciones del navegador se revisará la lista de elementos Route definidos y redireccionará a la que coincida.

```
redirect = () => {
  return <Redirect to="/somewhere/else" />;
}
```

Figura 7.13: Ejemplo de la utilización de Redirect en un método del componente.

```
onClick = () => {

    //Cálculos para obtener los datos a pasar

    this.props.history.push({
      pathname: '/reduction',
      state: {
        segmentationAlgorithm: "Matrix profile",
        parameter: this.state.parameter1,
        segments: res, serie: points,
        algoritmoRed: this.state.val,
        valoresOriginales: this.state.valoresSerie1,
        arraySegmentos: segmentsArray,
        segmentos: this.state.segmentos1,
        filename: this.props.location.state.filename}
    });
}
```

Figura 7.14: Ejemplo de la utilización del método history.push para ejecutar una redirección.

Introducción y recogida de datos utilizando Firebase

Otro de los puntos importantes del desarrollo de la web es el de la utilización de Firestore y Storage como base para el almacenamiento y consulta de las series temporales de los usuarios del sistema. Para utilizar Firestore se reutiliza la conexión a Firebase utilizada para el módulo de Authentication. Para utilizar Storage, en cambio, tenemos que añadir una nueva línea en dicha configuración rellenando el campo llamado *storageBucket*, poniendo el nombre del *bucket* o recurso[15] que previamente se ha tenido que crear en la consola de Firebase.

Tanto para utilizar los métodos de Firestore como para los de Storage necesitamos importar firebase y obtener las referencias de cada uno de los objetos que manejan las dos plataformas. Lo primero que tenemos que hacer es ejecutar la inicialización de FIrebase una única vez:

firebase.initializeApp(<credenciales firebase>)

Una vez inicializada, en la figura 7.15 se puede ver la obtención tanto del objeto de Firestore como del objeto con la referencia a Storage. A continuación se presentan las bases utilizadas en la implementación y utilización de dichos elementos:

```
const firestore = firebase.firestore();

const storage = firebase.storage();

const storageRef = storage.ref();
```

Figura 7.15: Obtención de los objetos necesarios para hacer las llamadas a los métodos de Firestore y Storage.

Para ambos módulos han sido definidas unas reglas de acceso en la plataforma, para que solamente los usuarios conectados puedan acceder a sus propios documentos. En las figuras 7.16 y 7.17 se pueden observar las reglas de Firestore y Storage respectivamente.

En el caso de Firestore solo el propio usuario puede acceder a sus ficheros o crear nuevos y solo los usuarios conectados pueden acceder a las *featured series* de la página inicial de la web. En Storage solo el usuario puede ver y escribir en su directorio.

[15]Storage bucket: https://cloud.google.com/storage/docs/key-terms?hl=es-419

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
     allow read, write: if request.auth.uid == userId;
    }
    match /featuredSeries/{document=**} {
      allow read: if request.auth.uid != null;
    }
  }
}
```

Figura 7.16: Reglas definidas en Firestore.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{userId}/{allPaths=**} {
      allow read, write: if request.auth.uid == userId;
    }
  }
}
```

Figura 7.17: Reglas definidas en Storage.

Firestore

Tanto para obtener elementos como para guardarlos, tenemos que hacer referencia a las colecciones o a los documentos pertenecientes a la ruta del elemento utilizando los métodos *collection()* y *doc()*. En el caso de realizar una petición de consulta, una vez tenemos esa referencia tenemos dos opciones: Que queramos obtener un solo documento o que queramos obtener todos los documentos de una colección.

En las figuras 7.18 y 7.19 tenemos dos ejemplos de un acceso tanto a un documento como a una colección, en el que vamos enlazando los métodos mencionados hasta que llegamos a lo que queremos obtener, ya sea una colección en uno de los casos o un documento en el otro. Como se puede observar, en ambos casos se utiliza el método *get()* para obtener los datos que necesitamos, y eso se debe a que la diferencia reside en la manera que tratamos lo que nos haya devuelto la promesa del *get()*.

Si la referencia a la que le hemos llamado con el *get()* es un documento, lo que nos devuelve la promesa es el documento en sí, si es que el documento existe. En el caso de la colección, lo que nos devuelve la promesa es una querySnapshot que lo llaman, en la que tenemos la lista de documentos que ha recuperado, que tendremos que recorrer para ir tratando cada documento de uno en uno.

En el caso de querer agregar un nuevo elemento a Firestore, lo que habrá que hacer es generar la misma referencia que cuando se quería realizar una petición de un elemento, solo que en este caso con esa referencia tenemos que ejecutar el método set. En la figura

```
var docRef = this.props.firestore.collection("users").doc(user.uid).collection("timeSeries").doc(name);

docRef.get().then(function(doc) {
    if (doc.exists) {
        // Realizar acciones con el documento obtenido
    } else {
        // Tratar errores de documento no existente
    }
}).catch(function(error) {
        // Tratar errores que hayan ocurrido durante la petición
});
```

Figura 7.18: Ejemplo de estructura de petición de un solo documento a Firestore.

```
var docRef = this.props.datos.collection('featuredSeries');
docRef.get().then(function(docs) {
    docs.forEach(function(doc) {
        // Tratar cada documento como corresponda
    })
}).catch(function(error) {
    // Tratar errores al obtener los documentos
});
```

Figura 7.19: Ejemplo de petición de todos los documentos de una colección.

7.20 se muestra un ejemplo en el que se ha realizado esta adición en este caso cuando un usuario introduce una serie nueva y por tanto hay que crearla en el sistema.

```
this.props.firestore.collection('users').doc(this.props.user.uid).collection('timeSeries').doc(this.state.filename).set({
    values: name,
    segmentationAlgorithm: "---",
    parameter: "---",
    reductionAlg: "---",
    originalReductionRatio: "---",
    newReductionRation: "---",
    originalError: "---",
    newError: "---",
    reconsValues: "---",
    name: this.state.filename
});
```

Figura 7.20: Ejemplo de creación de un nuevo documento en Firestore.

Si lo que queremos es modificar algún elemento o introducir nuevas propiedades sin eliminar las existentes, lo que se hace, como podemos ver en la figura 7.21, es utilizar el método set de la misma manera que en el caso anterior pero añadiendo el método *merge(true)* para indicarle que junte los atributos ya existentes con los que se querían añadir. En este caso esto se ha utilizado, por ejemplo, cuando ya teníamos una serie creada tal y como se veía en la figura 7.20 y necesitábamos actualizar sus datos cuando el usuario había decidido guardar los datos de la reducción.

```
docRef.set({
    segmentationAlgorithm: this.props.location.state.segmentationAlgorithm,
    parameter: this.props.location.state.parameter,
    originalError: this.state.errorOriginal,
    originalReductionRatio: this.state.reduccionOriginal,
    newReductionRation: this.state.reduccionNueva,
    newError: this.state.errorNueva,
    reconsValues: name,
    reductionAlg: this.props.location.state.algoritmoRed,
    segments: segs
}, { merge: true });
```

Figura 7.21: Ejemplo de actualización de un documento en Firestore utilizando *set()* y *merge()*.

En este caso existe también el método *update* pero si se ejecuta sobre una referencia de un documento no existente nos dará un error, mientras que el método *set()* con el *merge()* siempre funcionará ya que en caso de no existir, el documento será creado.

Storage

En la web, cuando un usuario sube un fichero, este es almacenado en una carpeta con su identificador único, teniendo así cada usuario su propia carpeta. Esto nos permite una mejor organización a la hora de restringir los permisos y adecuarlos al contexto. Como se ve en la figura 7.22, esto se consigue utilizando el método child, que si dicha ruta existe, añadirá en ella el fichero y, si no es así, creará la carpeta del usuario. A este método se le ha introducido la ruta completa que tendrá el fichero, que en este caso será el ID del usuario y el nombre del fichero.

Para colocar un fichero en dicha ruta se encadena el método child con el método put, al que se le tiene que pasar un objeto de tipo BLOB[16] (el fichero). En la figura se puede observar como se ha obtenido ese fichero de tipo BLOB desde el filereader que se ha encargado de leer el fichero subido por el usuario.

Para obtener un fichero de Storage, como se puede ver en la figura 7.23, tenemos que utilizar el mismo método child para indicar la ruta del fichero solo que en este caso tenemos que enlazarlo con el método *getDownloadURL()*. Esto nos devolverá una promesa que contiene como respuesta una URL de descarga del fichero. COn esta URL tenemos dos opciones: la que se ha llevado acabo en la figura, que es la de obtener el blob de dicho

[16]Blob: https://developer.mozilla.org/es/docs/Web/API/Blob

```
const content = fileReader.result;
var obj = JSON.parse(content);
var jsonse = JSON.stringify(obj);
var blob = new Blob([jsonse], {type: "application/json"});
var name = this.props.user.uid + "/" + this.state.filename;

this.props.storageRef.child(name).put(blob);
```

Figura 7.22: Lectura de un fichero subido por el usuario y almacenado en una ruta única en Storage para dicho usuario.

enlace para poder tener el objeto como tal o la de introducir la URL obtenida directamente en un componente o etiqueta HTML (href de una imagen por ejemplo).

Tal y como hemos dicho, en este caso se necesitaba de laobtención del fichero como tal, para poder así utilizar sus valores. Esto se ha hecho a través de una petición *XMLHttpRequest* con la que obtenemos el fichero en tipo BLOB, que posteriormente se lee con un Filereader para obtener los datos en tipo texto y posteriormente convertirlos a un objeto.

```
this.props.storageRef.child(this.props.user.uid + "/" + data.name).getDownloadURL().then(function(url) {
    var xhr = new XMLHttpRequest();
    xhr.responseType = 'blob';

    xhr.onload = function(event) {
        var blob = xhr.response;
        var reader = new FileReader;

        reader.onload = function() {
            var resu = reader.result;
            var obj = JSON.parse(resu);

            // Acciones con el objeto recibido (obj)
        };

        reader.readAsText(blob);

    };

    xhr.open('GET', url);
    xhr.send();

  }).catch(function(error) {
    //A cciones en caso de error
});
```

Figura 7.23: Ejemplo de obtención de un fichero almacenado en Storage.

Cabe mencionar, que tanto en el caso de Firestore como en el de Storage, las peticiones que se realizan son todas asíncronas, así que puede darse el caso en el que esto nos pueda llevar a algún tipo de error. En estos caso este hecho se gestiona de la misma manera que ha sido gestionado en las peticiones a los servicios API REST, utilizando el método *async* y el operador *await*. En este caso, lo que se tendría que hacer sería rodear la petición en

un método con *async* y utilizar el operador await para la petición del método *get()* para que la ejecución no continúe hasta que se reciba respuesta de Firebase.

Introducción de gráficos en pantalla con CanvasJSChart

Como herramienta de tratamiento de series temporales de datos, la aplicación requería de utilizar gráficas para la representación visual de dichas series y de sus modificaciones de cara a la utilidad para el usuario. Para ello se ha utilizado la herramienta CanvasJS-Chart, una librería de JavaScript que permite dibujar gráficas de todo tipo con opciones de actualización de los datos o de zoom y desplazamiento.

Esta herramienta funciona a través de JavaScript pero viene integrada a React, ofreciéndonos un componente fácilmente integrable y mostrable en cualquier aplicación de React. Dicho componente simplemente requiere de un parámetro de entrada en el que se le indican los ajustes relacionados con el formato de los ejes, rangos de los datos, tipo de gráfica (línea, tarta, barras...) y lo propios datos que va a contener la gráfica. En la figura 7.24 se puede observar un ejemplo de la representación de la gráfica en la interfaz de la web.

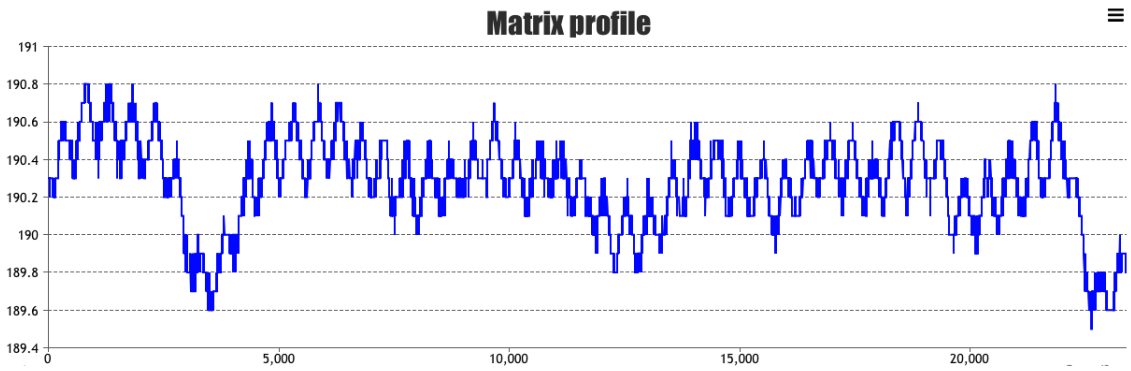


Figura 7.24: Representación de una gráfica con CanvasJSChart en la web.

En la figura 7.25 se muestra el formato que tiene, en este caso, el objeto que contiene los ajustes necesarios para mostrar la gráfica. En este caso tenemos estos ajustes pero la librería permite detallar en mayor medida ciertos aspectos que en este caso no han sido necesarios. A estas opciones vistas en la figura le falta el atributo data, que hace referencia a los datos en sí que tienen que ser representados. En ese caso, en cuanto al formato de los datos, en la figura 7.26 se muestra el formato que tiene en este caso una gráfica que muestra dos elementos diferentes, por un lado la serie temporal en un color y por el otro

unas barras verticales indicando la posición de cada segmento en otro color. Para añadirlos solo tenemos que formar esa estructura de datos en una variable y asignarle dicha variable al atributo data de las opciones.

```
const options1 = {
    zoomEnabled: true,
    animationEnabled: true,
    exportEnabled: true,
    title: {
      text: seg
    },
    axisY: {
      includeZero: false,
      gridDashType: "dash"
    }
};
```

Figura 7.25: Objeto que representa las propiedades necesarias para representar una gráfica con CanvasJS. En este caso a falta de la introducción de los datos en este objeto.

En este caso los datos mostrados son un pequeño ejemplo para que se comprenda la estructura de los mismos. En la aplicación estos datos se rellenan en tiempo de ejecución y son variables dependiendo del tamaño y de los puntos de la serie a mostrar.

Los datos de una gráfica que ya ha sido dibujada pueden ser modificables en tiempo de ejecución y ser reflejados esos cambios en la gráfica. La librería permite realizar esto de una manera muy sencilla, y es la de asociar el atributo data (los datos en sí, los puntos de la serie) del parámetro de opciones a una variable JavaScript que hayamos definido. Así, cuando el valor de dicha variable cambie, la gráfica lo detectará y se actualizará mostrando los nuevos datos añadidos. Este caso ha sido útil para representar una serie que no estaba segmentada y convertirla a la misma serie pero con los segmentos pintados. Se ha conseguido actualizando los datos de la variable cuando el servicio de segmentación nos devolvía los puntos por los que la serie debía de ser segmentada.

En líneas generales es una buena librería y fácil de integrar y utilizar en un entorno de React. Ha sido utilizada, de la manera explicada, para representar todas las gráficas necesarias en la web.

Realización de peticiones a los servicios API REST

Para realizar las peticiones a los servicios API REST de segmentación, reducción y re-

```
max = "Valor máximo de la serie de datos (nú";
options.data = [
  {
    type: "line",
    lineColor: "blue",
    dataPoints: [
      {x: 1, y: 192},
      {x: 2, y: 193},
      {x: 3, y: 193.4},
      {x: 4, y: 192},
      {x: 5, y: 193},
      {x: 6, y: 193.4}
    ]
  },
  {
    type: "column",
    dataPoints: [
      {label: "", x: 2, y: Math.ceil(max), color: "red"},
      {label: "", x: 4, y: Math.ceil(max), color: "red"},
      {label: "", x: 6, y: Math.ceil(max), color: "red"}
    ]
  }
];
```

Figura 7.26: Ejemplo del formato de los datos que necesita CanvasJS para representarlos. En este caso serían dos gráficas que se representan juntas en distintos colores, simulando una serie y los divisores de los segmentos

construcción se ha utilizado la API Fetch[17] se incluye de forma nativa en JavaScript. Permite realizar peticiones HTTP asíncronas de una manera sencilla a través del método fetch(), devolviéndonos una promesa con el resultado de la petición realizada.

En la figura 7.27 se presenta la sintaxis básica del método fetch(), que ejemplifica una simple petición de tipo GET, que devolvería una promesa que es tratada con los *then()*, para que en cuanto se reciba la respuesta pueda utilizarse ese JSON recibido.

```
fetch('http://example.com/movies.json')
  .then(function(response) {
    return response.json();
  })
  .then(function(myJson) {
    console.log(myJson);
  });
```

Figura 7.27: Ejemplo de petición HTTP básica de tipo GET realizada con el método fecth.

[17]Fetch API: https://developer.mozilla.org/es/docs/Web/API/Fetch_API/

Debido a la necesidad de realizar peticiones de tipo POST, la sintaxis utilizada en el proyecto no ha sido exactamente esa, tal y como se puede ver en la figura 7.28. Este ejemplo presentado es el que se utiliza en la petición a uno de los algoritmos de segmentación implementados como servicio web. Se pueden ver los atributos que se han añadido a la petición fetch para indicarle que es de tipo POST y para mandarle el contenido, el cuerpo del mensaje.

```
obj = {}; //Objeto con los datos a mandar
string = JSON.stringify(obj);

fetch('http://35.205.214.223/api/segment/ggs/', {
  method: 'POST',
  body: string,
  headers: {
    "Content-type": "application/json; charset=UTF-8"
    }
}).then(response => {
    return response.json();
  }).then(json => {

    this.setState({
      numberOfSegments3: json.value.length,
      segmentos3: json.value
    });

    this.finishLoading();
});
```

Figura 7.28: Petición HTTP de segmentado, de tipo POST utilizando el método fetch.

Posteriormente se recoge la promesa y a través del método then, tal y como se veía en el ejemplo básico, se ejecutan las acciones pertinentes, en este caso entre otras cosas se actualiza el estado del componente con algunos de los valores de la respuesta obtenida.

La estructura del método fetch como tal es idéntica en todos los casos en los que se han tenido que hacer peticiones a servicios API REST, pero en algunos casos el modo de ejecutarlas ha diferido debido al hecho de que son peticiones asíncronas. Esto supone una ventaja en mucho casos ya que nos permite realizar ciertas acciones en paralelo y ganar tiempo de ejecución de cara al usuario de la aplicación. Aún así, en algunos casos generan algún tipo de conflicto con ciertas otras acciones que necesitan los datos que el fetch recoge para realizar sus acciones.

Para solucionar ese hecho se ha utilizado en esos casos la función async[18] junto con el operador await, ambos nativos de JavaScript. La función async permite definir una función

[18] Async: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/funcion_asincrona

como función asíncrona para que a la hora de ejecutarse, JavaScript tenga conocimiento de que la función es asíncrona. Por otra parte, el operador await, sirve para indicar al sistema que espere hasta la finalización de la ejecución de una promesa.

Tal y como se ve en la figura 7.29, esto nos permite englobar la función de fetch en otra función que declaramos como función asíncrona. Entonces ya se puede colocar el operador await antes del fetch y, por tanto, durante la ejecución el sistema esperará a que el método finalice y tenga todos los valores accesibles. En este caso que mostramos, incluso, han tenido que realizarse unas peticiones en bucle, según el número de elementos presentes en una variable. Como lo que hacemos es almacenar la función en una variable, lo que tenemos que hacer para ejecutarla es llamar al objeto *getData()* en el que hemos almacenado la función.

```
const getData = async () => {
    for (const element of this.props.location.state.arraySegmentos) {

      await fetch('http://35.205.214.223/api/reduce/', {
          method: 'POST',
          body: pipS,
          headers: {
            "Content-type": "application/json; charset=UTF-8"
        }
        }).then(response => {
            return response.json();
          }).then(json => {
            //Almacenamos los datos recibidos
      });
    };
}
```

Figura 7.29: Ejemplo en el que se engloba el fetch en una función asíncrona y se comlementa con el await para que el sistema espera a la finalización del fetch para continuar.

En este ejemplo presentado se han eliminado las líneas ajenas a la estructura en sí, ya que son asignaciones a variables, cambios de estado, cálculos... que aislados de esta manera no ofrecen información útil. Este caso ha sido utilizado para, por cada segmento de la serie, reducirlo utilizando todos los algoritmos de segmentación y, por tanto, se necesitaba que las operaciones estuviesen completadas para que el usuario no realizase ninguna acción hasta que este proceso hubiese terminado.

7.2.3. Alojamiento

Elección de hosting

Una de las ventajas de haber desarrollado la web con react y Firebase, y basarse en servicios API REST, es su fácil despliegue en cualquier servidor de alojamiento y puesta en producción.

En este caso, se ha elegido desplegar la página web en la plataforma 000webhost[19] debido, principalmente, a que es gratuito y a pesar de ello ofrece un servicio más que notable al nivel de peticiones y memoria necesaria para la web en estos momentos. A pesar de las limitaciones que puede tener, ya sea de accesos simultáneos o de tamaño de la web, se ha considerado que en esta primera etapa la plataforma no tendrá la cantidad de visitas que puedan acarrear algún fallo, y que si así se diese, eso sería algo puntual. Por tanto, el servidor volvería a funcionar en un corto lapso de tiempo y podría volver a consultarse sin problema la web.

De todas maneras, se tiene en cuenta que esto se realiza de manera temporal, en lo que la herramienta se asienta y puede determinarse su utilidad futura en los trabajos reales del grupo BDI y si puede llegar a ser o no una herramienta con mayor afluencia de usos y peticiones. Si en un futuro el grupo desea mejorar esas condiciones del servidor o quiere añadir más funcionalidades a la herramienta podrían hacerlo y alojar la página web en una plataforma más completa o en el lugar que más les conviniese.

Finalmente, tal y como ya se menciona en el apartado de la planificación, la web está alojada en la dirección `https://timeseriessegmentation.000webhostapp.com`.

Si se desea conocer el proceso del despliegue, consultar anexo B.3.

[19] 000webhost: `https://es.000webhost.com`

8. CAPÍTULO

Pruebas

Durante el desarrollo del proyecto han sido llevadas a cabo ciertas pruebas para comprobar el correcto funcionamiento de las funcionalidades o utilidades que se iban desarrollando. Estas pruebas han sido aplicadas en todos los ámbitos del desarrollo, ya que en este caso se han comprobado aspectos tan dispares como el funcionamiento adecuado de los scripts desarrollados para pruebas de segmentado, implementación correcta del segmentado en el servicio web, pruebas de seguridad de accesos no autorizados en las páginas de la web, funcionamiento adecuado y consistente de la segmentación y la reducción en la web o almacenamiento correcto de las series en la Base de Datos.

A continuación se presentan las pruebas realizadas sobre la página web, quedando disponibles en el anexo D las pruebas correspondientes al servicio web y a la fase del análisis de los algoritmos:

En este caso se presentan las principales pruebas realizadas durante el desarrollo de la web. Se da por hecho la correcta implementación de los resultados de los servicios web y plataformas externas que se han utilizado, valorando los resultados y acciones derivados de su utilización.

Tabla 8.1: Pruebas realizadas durante el desarrollo de la plataforma web

Descripción	Salida esperada	Salida real	Resultado

Continúa en la siguiente página

Continuación de la Tabla 8.1

Almacenamiento en Firebase del usuario registrado en el sistema	El usuario registrado debe aparecer en la lista de usuarios del módulo Auth de Firebase	El usuario aparece en la lista al registrarse	CORRECTO
Inicio de sesión en el sistema con un usuario existente	El usuario inicia sesión y recibe un mensaje de bienvenida y de inicio correcto	Salida esperada	CORRECTO
Inicio de sesión en el sistema con un usuario no existente	El usuario inicia sesión y recibe un mensaje de error	Salida esperada	CORRECTO
Prueba de acceso no autorizado a una página	Un usuario no logueado intenta acceder a una de las páginas de acceso a usuarios logueados, es redireccionado al inicio y le es mostrada una ventana indicándole que el acceso está restringido sin hacer login	Salida esperada	CORRECTO
Prueba de funcionamiento de la conexión y almacenamiento de documentos en Firestore	En la consola de Firestore podremos ver el documento subido desde la web	Salida esperada	CORRECTO

Continúa en la siguiente página

Continuación de la Tabla 8.1

Prueba de funcionamiento de la conexión y almacenamiento de ficheros en Storage	En la consola de Storage podremos ver el documento subido desde la web	Error a la hora de subir el fichero	INCORRECTO (cod. 3.1)
Prueba de funcionamiento de la conexión y almacenamiento de documentos en Firestore	En la consola de Storage podremos ver el documento subido desde la web	Salida esperada	CORRECTO
Prueba de creación del documento del usuario en Firestore cuando este se registra	Se crea en Firestore un documento en la colección *users* con el id del usuario	Salida esperada	CORRECTO
Prueba de que un fichero subido por el usuario en la web es almacenado donde corresponde	El fichero debe de estar almacenado en la carpeta única del usuario en Storage y los datos de la serie en la colección del usuario llamada *timeSeries*	Salida esperada	CORRECTO
Prueba de utilización del método *fetch*, comprobar petición de prueba	Se debe recibir respuesta del servicio web, recibiendo la palabra 'respuesta'	Salida esperada	CORRECTO

Continúa en la siguiente página

Continuación de la Tabla 8.1

Comprobación de la petición al algoritmo de segmentación	Lista de los segmentos obtenidos, debe de coincidir con la lista obtenida al segmentar en local	Salida esperada en algunos casos, los resultados no eran consistentes en todas las peticiones	CON ERRORES (cod 3.2)
Comprobación de la petición al algoritmo de segmentación	Lista de los segmentos obtenidos, debe de coincidir con la lista obtenida al segmentar en local	Salida esperada	CORRECTO
Comprobación de la reducción de una serie completa en la web	Los resultados de la reducción deben coincidir con los obtenidos en local	Salida esperada	CORRECTO
Prueba de resultado obtenido en la reducción de la serie ya segmentada	El resultado final de la reducción por segmentos debe de ser coherente con el resultado realizado en local	Salida con valores incorrectos, totalmente distintos a la salida esperada	INCORRECTO (cod. 3.3)
Prueba de resultado obtenido en la reducción de la serie ya segmentada	El resultado final de la reducción por segmentos debe de ser coherente con el resultado realizado en local	Salida no consistente, resultados varían con los mismos parámetros y serie	INCORRECTO (cod. 3.4)
Prueba de resultado obtenido en la reducción de la serie ya segmentada	El resultado final de la reducción por segmentos debe de ser coherente con el resultado realizado en local	Obtenidos resultados esperados	CORRECTO

Continúa en la siguiente página

Continuación de la Tabla 8.1

Actualización de los datos guardados en Firebase cuando el usuario selecciona guardar la reducción	Los datos de la serie almacenada deben de ser actualizados con los nuevos valores modificados en la web	Salida esperada obtenida	CORRECTO

Análisis de las pruebas erróneas

- **3.1.** No se había completado la conexión con el módulo de Storage ya que no se había creado un *bucket* ni se había configurado dicho *bucket* en los parámetros de configuración de Firebase en los ajustes de la web. Una vez configurado la siguiente prueba fue exitosa.

- **3.2.** Los segmentos obtenidos no eran siempre correctos. Había un problema a la hora de segmentar la serie original por los puntos obtenidos, solapando en algunos casos algunos elementos y, por tanto, no siempre obteniendo los mismos valores. Se precisó el corte de la serie y los resultados siguientes fueron correctos.

- **3.3.** Se enviaban los datos incorrectos al servicio web, obteniendo así, como es obvio, unos resultados no correspondientes a la serie original.

- **3.4.** Se corrigió el hecho de mandar los datos incorrectos pero en este caso los resultado no siempre eran correctos. El problema estaba en que las peticiones de reducción a los servicios se hacen de manera asíncrona, por ello los resultados se iban obteniendo cuando la interfaz ya había cargado. Entonces al realizar las acciones con la interfaz los datos no siempre estaban preparados y se obtenían los resultados inconsistentes. Para solucionarlo se utilizó el método async/await ya mencionado, no mostrando la interfaz hasta que todos los datos estuviesen listos para su uso. Una vez con todos los datos listos las acciones del usuario devolvían resultados correctos.

9. CAPÍTULO

Seguimiento y control

Durante el desarrollo del proyecto se han ido realizando revisiones periódicas del estado del mismo respecto a la planificación inicial e implantando los cambios necesarios en cada momento para asegurar la viabilidad del proyecto.

En este apartado se resumen las distintas incidencias y cambios ocurridos en el proyecto en relación a la planificación, así como el resumen de las distintas desviaciones respecto a las horas y fechas panificadas inicialmente.

9.1. Incidencias

Durante el desarrollo del proyecto han ocurrido una serie de incidencias que han ido afectando al mismo modificando en gran medida las fechas de finalización. Alguna de las incidencias estaba contemplada en la gestión de riesgos pero la que más incidencia ha tenido no estaba contemplada previamente.

9.1.1. Aprendizaje y manejo de las series temporales y las técnicas de reducción

Uno de los posibles problemas previstos en la gestión de riesgos era la posibilidad de que costase comenzar a entender y trabajar con las series temporales y las técnicas de reducción de series temporales.

Como ya se verá en la desviación de horas, se subestimo el tiempo necesario para el entendimiento y comprensión de dichos elementos y conllevó a retrasar las siguientes tareas a las que esta predecía. De todas maneras, el retraso no era suficiente como para poner en peligro la finalización del proyecto, tal y como se menciona en la gestión de riesgos.

9.1.2. Trabajar en una empresa en paralelo al desarrollo del TFG

Desde principio de Febrero a finales de Abril estaba contemplado realizar unas prácticas por las mañanas en una empresa, habiéndose tenido en cuenta para el desarrollo de la planificación y las fechas de finalización del proyecto.

La incidencia ha venido cuando en la empresa se me propuso seguir trabajando de Mayo en adelante, manteniendo el horario de media jornada por las mañanas. Decidí aceptarlo, asumiendo la consecuente replanificación y retraso en la finalización del proyecto, que estaba inicialmente pensado para Junio pasó a retrasarse más de dos meses, hasta Septiembre. Esto es debido a que las horas diarias dedicadas al TFG a partir de ese momento serían aproximadamente la mitad de lo que originalmente estaban planeadas.

9.1.3. Problemas con la tecnología a utilizar en la web

Aunque en la planificación planteada en el capítulo 3 se presente React.js como la plataforma a utilizar en el desarrollo del proyecto, esa no fue la primera idea planteada.

Inicialmente la idea era la de desarrollar la aplicación web utilizando Symfony[1], un *framework*[2] basado en PHP y que previamente había conocido trabajando en la empresa. La idea era la de aprovechar el conocimiento previo y profundizarlo con la realización del TFG. El problema llegó en una de las reuniones mantenidas con la tutora y con los miembros del grupo BDI, en la que se plantearon las funcionalidades que debería tener la web.

Después de dicha reunión, se llegó a la conclusión de que las características de Symfony no eran las adecuadas para la realización de la aplicación web y fue cuando, previo análisis, se decidió implementarla con React.js.

[1] Symfony:
[2]

Esta decisión llevaría, inconscientemente, a retrasos o a malas previsiones en la realización del sistema web, ya que la herramienta era totalmente nueva para mi y, por tanto, no resultaba tan rápido como desarrollar con elementos y tecnologías ya conocidas de antemano.

9.2. Desviaciones de tiempo

Debido a los problemas mencionados en el apartado anterior, los tiempos de finalización de las tareas y los paquetes de trabajo no han sido los planificados inicialmente, alterando notablemente la fecha de finalización del proyecto. En líneas generales, la finalización del proyecto fue retrasada hasta Septiembre debido a la falta de horas diarias que se le podrían dedicar. Todas las fases y tareas que se estaba llevando y se llevarían a cabo a partir de Mayo retrasaron su finalización. A parte de eso, el propio desconocimiento de ciertos elementos y conceptos hizo que algunas previsiones de horas fuesen superadas y retrasasen la finalización.

Para más detalle, en el anexo C se pueden visualizar en detalle las dos tablas que representan las fechas reales de finalización de las tareas y las horas reales dedicadas a las mismas.

10. CAPÍTULO

Conclusiones y líneas a seguir

En este capítulo se describen las conclusiones obtenidas del desarrollo del proyecto, tanto la consecución de los objetivos planteados como las posibles vías futuras a poder seguir.

10.1. Conclusiones

La principal idea del proyecto era la de tratar de demostrar que la segmentación de series temporales permite almacenarlas de manera más óptima y para ello desarrollar una aplicación web que permitiese a ingenieros de datos subir sus series temporales, pudiendo segmentarlas y reducirlas para poder determinar dichas reducciones óptimas. En este sentido se puede concluir que los objetivos han sido llevados a cabo con éxito, reforzando la idea de que, a pesar de la complejidad que presenta, la segmentación de series temporales tiene un gran potencial y un futuro relevante en el ámbito de la Industria 4.0.

Las conclusiones detalladas de este TFG se han dividido en dos principales enfoques: conclusiones a nivel técnico y conclusiones a nivel personal. A continuación se detallan dichos enfoques:

10.1.1. Conclusiones a nivel técnico

El resultado global respecto a la realización de los conceptos técnicos del proyecto es muy positiva, habiendo llevado a cabo el TFG dentro de un entorno con diversos conocimien-

tos técnicos de gran complejidad. Se han conseguido completar los objetivos técnicos planteados, siendo los siguientes:

- **Análisis de distintos algoritmos de segmentación:** Se ha realizado una búsqueda y posterior análisis de ciertos algoritmos de segmentación encontrados en distintas publicaciones científicas, identificando los que podrían resultar de utilidad y comparando los resultados obtenidos sobre distintos bancos de datos.
- **Análisis del potencial de la segmentación de series temporales:** Se ha realizado un análisis del potencial de la segmentación a la hora de obtener una mejor reducción de una serie temporal, realizando distintas pruebas con los distintos algoritmos de reducción principales e interpretando los distintos resultados obtenidos.
- **Desarrollo de un servicio web de tipo API REST de segmentación de series temporales:** Se ha desarrollado un servicio web API REST de segmentación de series que ha sido incluido en el proyecto implementado por el grupo BDI de reducción y reconstrucción de series. Este servicio implementado tiene la funcionalidad de segmentar series temporales mediante el uso de los tres algoritmos seleccionados para el desarrollo del proyecto: Matrix profile, Simple segment y Greedy Gaussian Segmentation.
- **Creación de un sistema web de segmentación y reducción de series temporales:** Se ha implementado una plataforma web que permite la subida de series temporales en formato JSON, pudiendo segmentarla utilizando tres algoritmos diferentes y posteriormente reducir cada uno de esos segmentos con cuatro técnicas distintas de reducción. Permite a un ingeniero de datos visualizar de manera interactiva y rápida si una cierta segmentación puede llegar a ser útil en su contexto.

10.1.2. Conclusiones a nivel personal

Las conclusiones respecto a los objetivos planteados a nivel personal son positivos. En líneas generales se han adquirido diferentes tipos de conocimientos relacionados con las distintas fases del pre-procesado de los datos y del tratamiento de series temporales de la Industria 4.0. Los objetivos alcanzados son los siguientes:

- **Conocimiento del dominio relacionado con la Industria 4.0 y el pre-procesado de los datos:** El conocimiento que tenía sobre el entorno de la Industria 4.0 antes de comenzar el TFG era superficial y lejano a lo necesario para la realización del proyecto. Durante el TFG he conocido el proceso relacionado con la recogida y el

pre-procesado de los datos, entendiendo la dificultad y la labor que tiene esta parte del trabajo para poder posteriormente utilizar esos datos para su análisis. Entender el entorno me ha permitido comprender de donde vienen los datos con los que he trabajado y valorar la complejidad y el carácter de realidad que aporta trabajar directamente involucrado en un proyecto de colaboración real.

- **Aprendizaje del tratamiento, segmentación y reducción de series temporales:** El concepto de las series temporales me era totalmente desconocido antes del inicio del TFG y, una vez finalizado, he comprendido las maneras que existen de visualizarlas, interpretarlas y de tratarlas. El conocer y entender los datos con los que he trabajado me ha permitido en momentos puntuales comprender qué algoritmo de reducción podría ir mejor con cierta serie, entender qué puntos de segmentación podrían ser mejores o detectar algún tipo de anomalía presente en una serie. Por otra parte, el tratamiento de los datos ha sido esencial tanto para la fase del análisis de los algoritmos de segmentación como para el análisis del potencial de la segmentación.
- **Implementación con nuevas herramientas de desarrollo web:** Previo a la realización del proyecto no había desarrollado ningún tipo de plataforma utilizando una tecnología que permita desarrollar páginas tipo *One page*. A través del desarrollo del proyecto se ha logrado alcanzar un nivel medio en el manejo de este tipo de tecnologías, en este caso con React.js. Esto ha permitido cumplir con el objetivo de desarrollar la aplicación web , logrando las funcionalidades y objetivos planteados previamente en el análisis de requisitos. El haber realizado la aplicación web con esta tecnología me ha permitido abrir mi conocimiento del desarrollo web a otra manera totalmente distinta de estructurar un sistema y que está en auge. A nivel personal esto permitirá tener un conocimiento que en un futuro puede resultar interesante a la hora de afrontar cualquier proyecto o trabajo relacionado con el desarrollo web.
- **Capacidad de gestión y desarrollo de un proyecto de esta magnitud:** Abordar un proyecto de este nivel, con tantos conceptos totalmente nuevos y con la gestión que hay que realizar del mismo hace ver la dificultad y el esfuerzo y dedicación que requiere. La inexperiencia que tenía en relación a un trabajo de esta magnitud hace que la planificación no sea perfecta y que se cometan errores a la hora de confeccionarla y gestionarla, comprendiendo la importancia que tiene la experiencia en cualquier tipo de proyecto de este calibre. A pesar de ello se ha conseguido llevar a cabo el proyecto y cumplir con los objetivos planteados con la satisfacción de

haber comprobado el ser capaz de, a pesar de la inexperiencia, gestionar todos los imprevistos o problemas encontrados.

10.2. Mejoras futuras

A pesar de haber obtenido unos resultados positivos y un producto acorde con los objetivos planteados, este ámbito no deja de estar en continua revisión y mejora. Esto hace que puedan surgir nuevas aproximaciones a la segmentación o nuevos objetivos más ambiciosos por parte del grupo BDI.

Por ello se tienen en cuenta dichas posibles mejoras, sin dejar de lado las mejoras que en este momento pueden ser identificadas y que podrían ser abordadas en un futuro si se plantease un aumento del alcance.

10.2.1. Mejoras en el análisis de algoritmos de segmentación

El análisis de los distintos algoritmos de segmentación ha sido llevado a cabo según los objetivos y el alcance definidos pero podría realizarse en mayor profundidad o ser actualizado teniendo en cuenta estas posibles mejoras:

- **Revisión periódica de las publicaciones relacionadas con las segmentación de series temporales:** Tal y como ha sido mencionado, el ámbito de la Industria 4.0 avanza constantemente así que es muy posible que con el tiempo vayan apareciendo nuevas publicaciones sobre técnicas de segmentación así como mejoras de las ya existentes (precisión, tiempo de ejecución...). Por ello, si se desea mantener al día la plataforma, una línea futura podría ser la de ir realizando nuevas búsquedas y análisis de las novedades que se vayan encontrando, recogiendo las que resulten útiles y actualizando los resultados y comparaciones respecto a los algoritmos seleccionados en este proyecto.
- **Ampliación de los orígenes de las series temporales durante las pruebas:** En este proyecto han sido utilizados dos orígenes de datos: las series utilizadas por el grupo BDI recogidas de las máquinas de Urola y ciertas series de datos recogidas en el *UCR Archive*. Una posible mejora podría ser la de ampliar el espectro de orígenes de datos utilizados durante el análisis y pruebas, buscando obtener una mayor cantidad de elementos a comprar y, por tanto, mayor riqueza en los resultados obtenidos.

10.2.2. Mejoras en el servicio web de segmentación

Teniendo cuenta que el servicio web cumple con la función necesaria a nivel de desarrollo informático, depende de los propios algoritmos de segmentación para poder mejorarlo o actualizarlo. En este caso las mejoras que podrían realizarse estarían estrechamente relacionadas con las mejoras mencionadas sobre el apartado 10.2.1, ya que cualquier cambio relacionado con los algoritmos de segmentación llevaría a una actualización del servicio web de segmentación.

10.2.3. Mejoras en la aplicación web desarrollada

Tal y como ha sido mencionado en las mejoras del servicio web de segmentación, la actualización relacionada con los algoritmos de segmentación llevaría a una actualización de las funcionalidades de la aplicación web. Además, se pueden plantear ciertas líneas de mejora relacionadas con distintos aspectos de la plataforma, como podrían ser:

- **Actualización, eliminación o adición de algoritmos de segmentación:** Tal y como se ha mencionado en el apartado anterior, cualquier cambio relacionado con los la lista de algoritmos de segmentación reflejados en la web debe de ser actualizada en la plataforma. Por ejemplo, si en un futuro se encuentra otro algoritmo de segmentación funcional y se decide implementar en el servicio web, dicho algoritmo deberá ser introducido dentro de los algoritmos disponibles en la plataforma web.
- **Rediseño de las interfaces de la aplicación:** En el desarrollo de esta aplicación web no se ha puesto excesivo esfuerzo en el refinamiento y rediseño de las interfaces, ya que, al ser una herramienta prototipo, la importancia de la misma está en la capacidad de llevar a cabo las funcionalidades de manera correcta. Por tanto, una posible línea de mejora futura podría ser la de comenzar a rediseñar la plataforma si se considera que se pasará a darle un uso más intensivo o extendido a otro público.
- **Modificación de los criterios de diferenciación de los roles:** Una posible modificación podría ser la de rediseñar los roles de la aplicación para adecuarlos a cierto contexto que pueda surgir. Por ejemplo, si la aplicación va a pasar a ser utilizada por un grupo de personas que queremos que tengan diferente rol se podría definir cierta característica distintiva de cada rol y no utilizar la de la verificación del correo implementada actualmente.
- **Posibilidad de determinar el parámetro de cada reducción:** En la implemen-

tación actual, la herramienta está configurada para realizar las reducciones con un parámetro fijado para que la aplicación sea más dinámica y permita al usuario ver los cambios de manera más rápida. Una posible línea de mejora podría ser implementar, a parte de la manera que ya está desarrollada, una opción en que el usuario pueda introducir el parámetro de entrada de la técnica de reducción, si así lo desea.

- **Aceptar más formatos de ficheros de series temporales:** Si se considera oportuno debido a necesidades futuras, podría introducirse la posibilidad de permitir en la herramientas otro tipo de formatos de ficheros a parte del JSON, como pueden ser: CSV, TXT...
- **Integrar la parte de la aplicación web perteneciente a Firebase con la que tiene el grupo de investigación:** Si en un futuro se decide acoplar la base de datos de esta herramienta a la de la aplicación web i4TSPS desarrollada por el grupo BDI, podría utilizarse el entorno de Firebase del que ya disponen. Esto permitiría, por ejemplo, utilizar los usuarios ya disponibles en la plataforma I4TSPS en la herramienta de segmentado y que así cada usuario pueda acceder con la misma cuenta a ambas plataformas. Esto requeriría una modificación de ciertas configuraciones tanto en la aplicación de segmentación como en la de I4TSPS.

Anexos

A. ANEXO

Scripts desarrollados para la realización de pruebas relacionadas con la segmentación

En este anexo se presentan los distintos scripts desarrollados durante la fase de análisis del potencial de la segmentación de series temporales de cara a la optimización de la reducción global.

A.1. Script en R que permite segmentar la serie indicada con los segmentos indicados y reducirlos usando DFT.

Este algoritmo permite segmentar la serie con los segmentos indicados y por cada uno de ellos reducirlo utilizando la técnica DFT con parámetro el 10 % de los puntos de la serie.

```
1  autoSeg <- function(fichero, segmentos){
2    segmentos <- read.table(segmntos)
3    file <- fichero
4    res <- c()
5
6    for (segmento in 1:length(segmentos$V1)  ) {
7      if (segmento > 1){
8        if (segmento == 2){
9
10       }else{
11         seg <- read.table(file,sep = ",", dec=".",skip = segmentos$V1[segmento-1], nrows =
     segmentos$V1[segmento]-segmentos$V1[segmento-1])
12         num <- (segmentos$V1[segmento]-segmentos$V1[segmento-1])/10
```

```
      }
    } else{
      seg <- read.table(file,sep = ",", dec=".",skip = 1,nrows = segmentos$V1[segmento])
      num <- (segmentos$V1[segmento])/10
    }
    nas <- seg$V2[!is.na(seg$V2)]
    adp <- prepare4DFT(nas)
    red <- reduceDFT(adp,num)
    rec <- reconstructDFT(red)
    res <- c(res,rec$y)
  }
  fin <- res
}
```

A.2. Script en R que permite segmentar la serie indicada con los segmentos indicados y reducirlos usando PIP.

Este algoritmo permite segmentar la serie con los segmentos indicados y por cada uno de ellos reducirlo utilizando la técnica PIP con parámetro 433.

```
autoSeg <- function(fichero, segmentos){
  segmentos <- read.table(segmntos)
  file <- fichero
  res <- c()
  num <- 433

  for (segmento in 1:length(segmentos$V1)  ) {
    if (segmento > 1){
      if (segmento == 2){

      }else{
        seg <- read.table(file,sep = ",", dec=".",skip = segmentos$V1[segmento-1], nrows =
    segmentos$V1[segmento]-segmentos$V1[segmento-1])
      }
    } else{
      seg <- read.table(file,sep = ",", dec=".",skip = 1,nrows = segmentos$V1[segmento])
    }
    nas <- seg$V2[!is.na(seg$V2)]
    adp <- prepare4PIP(nas)
    red <- reducePIP(adp,num)
    rec <- reconstructPIP(red)
    res <- c(res,rec$y)
  }
  fin <- res
}
```

A.3. Script en Python que recorre todos los ficheros y los segmenta utilizando el algoritmo del Simple segment y reduce los segmentos utilizando DFT.

Este algoritmo recorre todo el sistema de fichero en busca de las series, las segmenta utilizando Simple segment y reduce cada segmento con el algoritmo de reducción utilizado originalmente en dicha serie. Una vez reducidos todos los segmentos de cada serie calcula el error y el porcentaje de reducción de la serie reconstruida y la compara con la reducción original. Todos estos resultado se van almacenando en un fichero que será almacenado en el sistema con todos esos resultados mencionados.

```
from matplotlib.pylab import gca, figure, plot, subplot, title, xlabel, ylabel, xlim,show
from matplotlib.lines import Line2D
from rpy2.robjects import r
from os import listdir
from os import walk
from os.path import join
import segment
import fit
import pandas as pd
import numpy as np
import sys
import os

from rpy2.robjects.packages import SignatureTranslatedAnonymousPackage

string = """
    source("./DFT.R")
    source("./PIP.R")

    getRMSE <- function(so, sr){ getRMSE <- (sum((so - sr)^2)/length(so))^(1/2) }


    automatizarSegmentacionDFT <- function(fichero,segments){
    segmentos <- read.table(segments)
    original <- read.table(fichero,header = T,sep = ",",dec = ".")

    nas <- original$value[!is.na(original$value)]
    adp <- prepare4DFT(nas)
    red <- reduceDFT(adp,nrow(original)/10)
    rec <- reconstructDFT(red)

    datos <- c()

    tamanioOriginal <- file.size(fichero, units = "B")
```

```
35      urlRed <- "originalREDUCIDO.csv"
36      saveDFT(red,urlRed,Sys.Date(),1)
37      tamanioReducido <- file.size(urlRed,units = "B")
38      ratioReduccionOriginal <- 100 - (tamanioReducido/tamanioOriginal*100)
39
40      serie_or <- rec
41
42      segs <- vector(mode = "list")
43      porcentajes <- c()
44      res <- c()
45
46      total <- nrow(original)/10
47
48      j <- 1
49      for (segmento in 1:length(segmentos$V1)) {
50      if (segmento > 1){
51      if (segmento == length(segmentos$V1)){
52      numSeg <- (segmentos$V1[segmento]-segmentos$V1[segmento-1]+1)
53      } else {
54      numSeg <- (segmentos$V1[segmento]-segmentos$V1[segmento-1])
55      }
56      seg <- read.table(fichero,sep = ",", dec=".",skip = segmentos$V1[segmento-1]+1, nrows =
         numSeg)
57      num <- numSeg/10
58
59      } else{
60      seg <- read.table(fichero,sep = ",", dec=".",skip = 1,nrows = segmentos$V1[segmento])
61      numSeg <- segmentos$V1[segmento]
62      num <- (segmentos$V1[segmento])/10
63      }
64
65      segs <- append(segs,seg)
66
67      }
68
69      j <- 1
70      coef <- c()
71
72
73      tamanioRedSeg <- 0
74
75      res <- c()
76      i <- 2
77      aaa<-c()
78      while (i <= length(segs)) {
79
80      aaa<-c(aaa,num)
81
82      num <- length(segs[i]$V2)/10
83      adp <- prepare4DFT(segs[i]$V2)
84      red <- reduceDFT(adp)
85      rec <- reconstructDFT(red)
```

```
86      res <- c(res,rec$y)
87      i <- i+2
88      saveDFT(red,"fichSegRed.csv",Sys.Date(),1)
89      tamanioRedSeg <- tamanioRedSeg + file.size("fichSegRed.csv",units = "B")
90      }
91
92      ratioSeg <- 100 -(tamanioRedSeg/tamanioOriginal*100)
93      datos <- c(getRMSE(original$value,serie_or$y),getRMSE(original$value,res),
         ratioReduccionOriginal[1],ratioSeg[1])
94      automatizarSegmentacion <- datos
95
96      }
97
98      automatizarSegmentacionPIP <- function(fichero,segments){
99      segmentos <- read.table(segments)
100     original <- read.table(fichero,header = T,sep = ",",dec = ".")
101
102     #nas <- original$value[!is.na(original$value)]
103     adp <- prepare4PIP(original$value)
104     red <- reducePIP(adp,433)
105     rec <- reconstructPIP(red)
106
107     datos <- c()
108
109     tamanioOriginal <- file.size(fichero, units = "B")
110     urlRed <- "originalREDUCIDO.csv"
111     savePIP(red,urlRed,Sys.Date(),1)
112     tamanioReducido <- file.size(urlRed,units = "B")
113     ratioReduccionOriginal <- 100 - (tamanioReducido/tamanioOriginal*100)
114
115     serie_or <- rec
116
117     segs <- vector(mode = "list")
118     porcentajes <- c()
119     res <- c()
120
121     total <- nrow(original)/10
122
123     j <- 1
124     for (segmento in 1:length(segmentos$V1)) {
125     #print(segmentos$V1[segmento])
126     if (segmento > 1){
127     if (segmento == length(segmentos$V1)){
128     numSeg <- (segmentos$V1[segmento]-segmentos$V1[segmento-1]+1)
129     } else {
130     numSeg <- (segmentos$V1[segmento]-segmentos$V1[segmento-1])
131     }
132     seg <- read.table(fichero,sep = ",", dec=".",skip = segmentos$V1[segmento-1]+1, nrows =
         numSeg)
133     num <- numSeg/10
134
135     } else{
```

```
136     seg <- read.table(fichero,sep = ",", dec=".",skip = 1,nrows = segmentos$V1[segmento])
137     numSeg <- segmentos$V1[segmento]
138     }
139
140     segs <- append(segs,seg)
141
142     }
143
144     j <- 1
145     coef <- c()
146
147
148     tamanioRedSeg <- 0
149
150     res <- c()
151     i <- 2
152     aaa<-c()
153
154
155     while (i <= length(segs)) {
156     param <- 433
157     if (length(segs[i]$V2) < 433){
158         if (length(segs[i]$V2) <= 400){
159             if (length(segs[i]$V2) <= 40){
160                 if (length(segs[i]$V2) <= 2){
161                     param <- 2
162                 } else {
163                     param <- floor(length(segs[i]$V2)/2)
164                 }
165             } else {
166                 param <- floor(length(segs[i]$V2)/20)
167             }
168         } else {
169             param <- length(segs[i]$V2)/200
170         }
171     }
172     #we <- as.numeric(nas)
173     adp <- prepare4PIP(segs[i]$V2)
174     #nas <- adp[!is.na(adp)]
175     red <- reducePIP(adp,param)
176     rec <- reconstructPIP(red)
177     res <- c(res,rec$y)
178     i <- i+2
179     savePIP(red,"fichSegRed.csv",Sys.Date(),1)
180     tamanioRedSeg <- tamanioRedSeg + file.size("fichSegRed.csv",units = "B")
181     }
182
183     ratioSeg <- 100 -(tamanioRedSeg/tamanioOriginal*100)
184     datos <- c(getRMSE(original$value,serie_or$y),getRMSE(original$value,res),
        ratioReduccionOriginal[1],ratioSeg[1])
185     automatizarSegmentacion <- datos
186
```

```
187     }
188 
189     """
190 
191 powerpack = SignatureTranslatedAnonymousPackage(string, "powerpack")
192 
193 sys.setrecursionlimit(1000000000)
194 
195 max_error = 10
196 
197 def returnReductionMethod(filename):
198     sep = filename.split("_")
199     sep2 = filename.split("/")
200     dir = listdir("reduced_time_series/UROLA/E12L10/WMT_GTFE7Q_4/45R3QF/"+sep[1]+"/"+sep2[2])
201     if (dir[0] != "DFT" and dir[0] != "PIP"):
202         return dir[1]
203     else:
204         return dir[0]
205 
206 def draw_plot(data,plot_title):
207     plot(range(len(data)),data,alpha=0.8,color='red')
208     title(plot_title)
209     xlabel("Samples")
210     ylabel("Signal")
211     xlim((0,len(data)-1))
212 
213 def draw_segments(segments):
214     ax = gca()
215     for segment in segments:
216         line = Line2D((segment[0],segment[2]),(segment[1],segment[3]))
217         ax.add_line(line)
218 #        print(segment[2])
219 
220 
221 
222 def get_line(x1, y1, x2, y2):
223     points = []
224     issteep = abs(y2-y1) > abs(x2-x1)
225     if issteep:
226         x1, y1 = y1, x1
227         x2, y2 = y2, x2
228     rev = False
229     if x1 > x2:
230         x1, x2 = x2, x1
231         y1, y2 = y2, y1
232         rev = True
233     deltax = x2 - x1
234     deltay = abs(y2-y1)
235     error = float(deltax / 2)
236     y = y1
237     ystep = None
238     if y1 < y2:
```

```
        ystep = 0.10
    else:
        ystep = -0.10
    for x in np.arange(x1, x2 + 0.1,0.1):
        if issteep:
            points.append((y, x))
        else:
            points.append((x, y))
        error -= deltay
        if error < 0:
            y += ystep
            error += deltax
    # Reverse the list if the coordinates were reversed
    if rev:
        points.reverse()
    return points


fin = np.array([['Serie','Error original','Error reducido','Reduccion original','Reduccion
     segmentado','Algoritmo']])
for (path,carpetas,archivos) in walk("Seriesaa"):
    for i in archivos:
        nombreFich = join(path,i)
        filename,extension = os.path.splitext(nombreFich)
        if (extension == '.csv'):
            res = open(filename+".txt","w+")
            with open(join(path,i)) as f:
                file_lines = f.readlines()
            print("Ha abierto el fichero " + nombreFich)
            metodo = returnReductionMethod(filename)
            data = [float(x.split(",")[1].strip()) for x in file_lines[1:86500]]
            if data.count(0) == len(data) or metodo == "DFT":
                print("Todos los elementos son 0")
            else:
                segments = segment.slidingwindowsegment(data, fit.interpolate, fit.sumsquared_
     error, max_error)
                for seg in segments:
                    res.write("%d\r\n" % seg[2])
                res.close()
                strings = filename.split("/")
                if (metodo == "PIP"):
                    resul = powerpack.automatizarSegmentacionPIP(nombreFich,filename+".txt")
                    resultados = np.append([strings[2]],resul)
                    resultados = np.append(resultados,metodo)
                    fin = np.append(fin,[resultados],axis=0)
                elif (metodo == "DFT"):
                    print("DFT")
#resul = powerpack.automatizarSegmentacionDFT(nombreFich,filename+".txt")



df = pd.DataFrame(fin)
```

```
289 df.to_csv(r'resultadoPrueba.csv',index=False,header=False)
```

A.4. Script para convertir las series del *UCR Archive* .

Este algoritmo permite pasar las series del *UCR Archive* del formato en el que vienen a un formato en CSV que el grupo BDI utiliza en sus series.

```
1  file_name <- "C:/Users/kevin/Desktop/EthanolLevel_TRAIN"
2
3  #Load Data (Si el fichero es muy grande puede que tengas que leer Ìlnea a Ìlnea)
4  data <- read.table(file_name, dec = ".", header = F) #504 Series x 1 label & 1751 timestamps
5  data <- data[, -1] #Remove label
6
7  #Get time Series
8  ts_1 <- data[1, ] #Select first time series
9  ts_1 <- as.data.frame(t(ts_1)) #Transpose it
10 ts_1 <- data.frame(timestamps = 1:nrow(ts_1), values = ts_1[, 1]) #Ponemos un timestamp false que
       es una secuencia desde 1 hasta el numero de valores de la serie
11 #La serie ya esta en el formato con el que solemos trabajar
12
13 plot(ts_1$values, type = "l")
14
15 #Extract all the series into a folder and plot them
16 output_dir <- "/Volumes/Datos/SeriesUCR"
17 for(i in 1:nrow(data)){
18   ts <- data[i, ] #Select first time series
19   ts <- as.data.frame(t(ts)) #Transpose it
20   ts <- data.frame(timestamps = 1:nrow(ts), values = ts[, 1])
21   ts_name <- paste(c(output_dir, "/ts_", i, ".csv"), sep = "", collapse = "")
22   plt_name <- paste(c(output_dir, "/ts_plot_", i, ".png"), sep = "", collapse = "")
23   write.table(ts, ts_name, row.names = F, col.names = T, sep = ",", dec = ".")
24   png(plt_name, width=1080, height = 720)
25   plot(ts$values, type = "l")
26   dev.off()
27 }
```

B. ANEXO

Comandos y procesos utilizados en la instalación, configuración y desarrollo del servicio web de segmentación y de la aplicación web

B.1. Instalación inicial de Django-rest-framework

Lo primero que necesitamos es instalar tanto Django como django-rest-frameworks, que sealizará a través de pip[1]. Previamente en el sistema está instalada la versión 2.7.10 de Python para que instalemos estos módulos con pip[2]:

pip install django

pip install djangorestframework

Con estos dos comandos tendremos lo necesario para poder crear nuestra aplicación Django con la posibilidad de crear un nuevo servicio web. Lo siguiente que tendremos que hacer será movernos al directorio donde queremos crear nuestro proyecto y crear dicho proyecto introduciendo este comando:

django-admin.py startproject tfgSegmentacion

[1] PIP: Gestor de paquetes de python, se instala junto a la instalación de Python a partir de las versiones 2.7.9 (o superior) y 3.4 (o superior)

[2] Instalar Python: https://www.python.org/downloads/

Una vez tenemos el proyecto entramos a la carpeta tfgSegmentacion creada y creamos la aplicación que irá dentro del proyecto:

```
django-admin.py startapp segmentacion
```

Una vez seguidos estos pasos tendremos el esqueleto de nuestro servicio ya creado.

B.2. Comandos utilizados durante el desarrollo de la aplicación web

Comando utilizado para instalar la estructura del proyecto:

```
git clone https://github.com/Phoqe/react-material-ui-firebase.git tfg-web-react
```

Comando para instalar un nuevo paquete en el proyecto:

```
npm install [nombrePaquete]
```

Comando para arrancar el proyecto:

```
npm start
```

Comando para compilar el proyecto y obtener la versión de producción:

```
npm run-script build
```

B.3. Proceso de despliegue del sistema web

El proceso de despliegue de la página web con React es sencillo y no requiere más que de la ejecución de un comando. Abrimos la terminal y nos vamos a la carpeta del proyecto, una vez allí no tenemos más que ejecutar el siguiente comando:

```
npm run-script build
```

Con esto, dentro de la carpeta del proyecto, se nos genera un directorio de nombre build, que contiene todos los ficheros que tenemos que alojar en el servidor para poner en funcionamiento la web.

Por tanto, lo único que tenemos que hacer es subir dichos ficheros al servidor a través del *file manager* que nos ofrece el hosting. Al copiar los ficheros generados la plataforma web estará disponible para acceder de manera inmediata.

C. ANEXO

Periodos de finalización y horas reales dedicadas a las distintas tareas del proyecto

Tabla C.1: Tabla en la que se indican las fechas de finalización reales de los distintos paquetes de trabajo y tareas del proyecto.

Tareas	Fecha final prevista	Final real	Desviación
Conocimiento del dominio (CD)	15/02/19	19/02/19	+ 4 días
Adquisición de conocimiento (AC)	15/02/19	19/02/19	+ 4 días
AC1: Conocer contexto del dominio	09/02/19	09/02/19	
AC2: Conocer formatos series temporales	15/02/19	19/02/19	+ 4 días
AC3: Conocer plataforma I4TSPS	29/01/19	29/01/19	
Análisis algoritmos de segmentación (AAS)	25/03/19	15/04/19	+ 3 semanas
Búsqueda de algoritmos (BA)	20/02/19	21/02/19	+ 1 día

Continúa en la siguiente página

Continuación de la Tabla C.1

Tareas	Fecha final prevista	Final real	Desviación
BA1: Recogida de los principales algoritmos de segmentación	18/02/19	20/02/19	+ 2 días
BA2: Analizar la factibilidad del uso de cada algoritmo	19/02/19	21/02/19	+ 1 día
BA3: Realizar la selección definitiva	20/02/19	21/02/19	+ 1 día
Pruebas de algoritmos (PRA)	25/03/19	15/04/19	+ 3 semanas
PRA1: Definir las pruebas a realizar	23/02/19	15/03/19	+ 3 semanas
PRA2: Implementar los algoritmos que no estén implementados	02/03/19	02/03/19	
PRA3: Realizar las pruebas con los distintos algoritmos	20/03/19	10/04/19	+ 3 semanas
PRA4: Analizar los resultados obtenidos y sacar conclusiones	25/03/19	15/04/19	+ 3 semanas
Análisis de segmentación en relación a la reducción (ASR)	02/05/19	05/06/19	+ 1 mes
Pruebas (PR)	25/04/19	02/06/19	+ 5 semanas
PR1: Definir las pruebas a realizar	15/04/19	26/05/19	+ 5 semanas
PR2: Realizar scripts convenientes para las pruebas	18/04/19	20/04/19	+ 2 días
PR3: Realizar las pruebas con los distintos tipos de series	25/04/19	02/06/19	+ 5 semanas

Continúa en la siguiente página

Continuación de la Tabla C.1

Tareas	Fecha final prevista	Final real	Desviación
Conclusiones (CO)	02/05/19	05/06/19	+ 1 mes
CO1: Analizar los resultados obtenidos en las pruebas	26/04/19	03/06/19	+ 5 semanas
CO2: Plasmar las conclusiones obtenidas de los resultados	02/05/19	05/06/19	+ 5 semanas
Sistema web (SW)	10/06/19	25/07/19	+ 6 semanas
Servicio web (SER)	09/05/19	24/06/19	+ 6 semanas
SER1: Diseño del servicio web	04/05/19	15/06/19	+ 6 semanas
SER2: Elaboración del servicio web	07/05/19	18/06/19	+ 6 semanas
SER3: Realización de pruebas en local	08/05/19	20/06/19	+ 6 semanas
SER4: Desplegar el servicio en un servidor personal	08/05/19	20/06/19	+ 6 semanas
SER5: Integrar el servicio web en el del grupo BDI	09/05/19	24/06/19	+ 6 semanas
Página web (PW)	10/06/19	25/07/19	+ 6 semanas
PW1: Análisis de requisitos	09/05/19	25/06/19	+ 6 semanas
PW2: Diseño general de de la web	11/05/19	27/06/19	+ 6 semanas
PW3: Elaboración de los casos de uso principales	13/05/19	29/06/19	+ 6 semanas
PW4: Preparación del entorno de desarrollo de la web	14/05/19	01/07/19	+ 6 semanas
PW5: Elaborar la página web	10/06/19	25/07/19	+ 6 semanas

Continúa en la siguiente página

Continuación de la Tabla C.1

Tareas	Fecha final prevista	Final real	Desviación
PW6: Revisión de lo elaborado	10/06/19	25/07/19	+ 6 semanas
Gestión (G)	01/07/19	15/08/19	+ 6 semanas
Planificación (P)	10/05/19	10/05/19	
P1: Identificación de requisitos	04/02/19	10/02/19	+ 1 semana
P2: Realización de la planificación inicial	18/02/19	22/02/19	+ 4 días
P3: Modificación de la planificación inicial	10/05/19	10/05/19	
Seguimiento y control (SyC)	01/07/19	15/08/19	+ 6 semanas
SyC1: Reuniones con la tutora a lo largo del proyecto	01/07/19	18/08/19	+ 6 semanas
SyC2: Elaboración de documento de anotación de horas	15/06/19	15/08/19	+ 2 meses
SyC3: Recopilación de información relevante	15/06/19	15/08/19	+ 2 meses
SyC4: Contraste del seguimiento con la planificación	15/06/19	15/08/19	+ 2 meses
Documentación (D)	01/07/19	02/10/19	+ 3 meses
Memoria (MEM)	15/06/19	15/09/19	+ 3 meses
MEM1: Preparar el entorno para desarrollar la memoria	10/02/19	10/02/19	
MEM2: Desarrollar la memoria	15/06/19	05/09/19	+ 3 meses
Preparar defensa (DEF)	01/07/19	15/09/19	+ 2 meses y 1/2
DEF1: Identificar conceptos que se presentarán	17/06/19	10/09/19	+ 2 meses y 1/2

Continúa en la siguiente página

Continuación de la Tabla C.1

Tareas	Fecha final prevista	Final real	Desviación
DEF2: Generar elementos audiovisuales como apoyo	20/06/19	13/09/19	+ 2 meses y 1/2
DEF3: Preparar la defensa	01/07/19	15/09/19	+ 2 meses y 1/2

Tabla C.2: Tabla en la que se indican las horas estimadas y las horas reales invertidas en cada tarea y paquete de trabajo.

Tareas	Horas estimadas	Horas reales
Conocimiento del dominio (CD)	27	42
Adquisición de conocimiento (AC)	27	42
AC1: Conocer contexto del dominio	5	10
AC2: Conocer formatos series temporales	20	30
AC3: Conocer plataforma I4TSPS	2	2
Análisis algoritmos de segmentación (AAS)	67	83
Búsqueda de algoritmos (BA)	17	13
BA1: Recogida de los principales algoritmos de segmentación	10	10
BA2: Analizar la factibilidad del uso de cada algoritmo	5	2
BA3: Realizar la selección definitiva	1	1
Pruebas de algoritmos (PRA)	50	70
PRA1: Definir las pruebas a realizar	5	15
PRA2: Implementar los algoritmos que no estén implementados	30	20
PRA3: Realizar las pruebas con los distintos algoritmos	10	30

Continúa en la siguiente página

Continuación de la Tabla C.2

Tareas	Horas estimadas	Horas reales
PRA4: Analizar los resultados obtenidos y sacar conclusiones	5	5
Análisis de segmentación en relación a la reducción (ASR)	61	77
Pruebas (PR)	55	70
PR1: Definir las pruebas a realizar	5	10
PR2: Realizar scripts convenientes para las pruebas	20	20
PR3: Realizar las pruebas con los distintos tipos de series	30	40
Conclusiones (CO)	7	7
CO1: Analizar los resultados obtenidos en las pruebas	2	2
CO2: Plasmar las conclusiones obtenidas de los resultados	5	5
Sistema web (SW)	103	140
Servicio web (SER)	20	20
SER1: Diseño del servicio web	1	1
SER2: Elaboración del servicio web	10	10
SER3: Realización de pruebas en local	2	2
SER4: Desplegar el servicio en un servidor personal	5	5
SER5: Integrar el servicio web en el del grupo BDI	2	2
Página web (PW)	93	120
PW1: Análisis de requisitos	2	2
PW2: Diseño general de de la web	5	2
PW3: Elaboración de los casos de uso principales	5	5
PW4: Preparación del entorno de desarrollo de la web	1	1
PW5: Elaborar la página web	70	100

Continúa en la siguiente página

Continuación de la Tabla C.2

Tareas	Horas estimadas	Horas reales
PW6: Revisión de lo elaborado	10	10
Gestión (G)	36	40
Planificación (P)	15	19
P1: Identificación de requisitos	2	2
P2: Realización de la planificación inicial	10	15
P3: Modificación de la planificación inicial	3	2
Seguimiento y control (SyC)	21	21
SyC1: Reuniones con la tutora a lo largo del proyecto	10	10
SyC2: Elaboración de documento de anotación de horas	2	1
SyC3: Recopilación de información relevante	4	5
SyC4: Contraste del seguimiento con la planificación	5	5
Documentación (D)	49	67
Memoria (MEM)	41	61
MEM1: Preparar el entorno para desarrollar la memoria	1	1
MEM2: Desarrollar la memoria	40	60
Preparar defensa (DEF)	8	6
DEF1: Identificar conceptos que se presentarán	1	1
DEF2: Generar elementos audiovisuales como apoyo	5	3
DEF3: Preparar la defensa	2	2
Total	343	449

D. ANEXO

Pruebas realizadas durante el análisis de algoritmos y el desarrollo del servicio web

D.1. Pruebas sobre los algoritmos de segmentación

En este caso se muestran las pruebas realizadas en relación con los algoritmos de segmentación y todos los scripts desarrollador para la realización del análisis de la segmentación. No se abordan las pruebas en el sentido en el que han sido planteadas en el apartado 4.4, si no que se abordan más en la comprobación de que los métodos utilizados para desarrollar dichas pruebas devuelvan los valores correctos y que funcionen de la manera que se espera.

Tabla D.1: Pruebas realizadas sobre los distintos scripts desarrollados para la comparación de algoritmos de segmentación y análisis del segmentado.

Descripción	Salida esperada	Salida real	Resultado

Continúa en la siguiente página

Continuación de la Tabla D.1

Descripción	Salida esperada	Salida real	Resultado
Script desarrollado para realizar pruebas de segmentación convierte bien los datos CSV al formato necesario para trabajar con R	Los datos del campo *value* del CSV tienen que ser los mismos que los que originalmente había en el fichero y tener un formato determinado	Los datos obtenidos eran los correctos pero el formato no era el correcto	CON ERRORES (cod. 1.1)
Script desarrollado para realizar pruebas de segmentación convierte bien los datos CSV al formato necesario para trabajar con R	Los datos del campo *value* del CSV tienen que ser los mismos que los que originalmente había en el fichero y tener un formato determinado	Los datos obtenidos eran los correctos y tenían el formato adecuado	CORRECTO
Comprobación del script que segmenta una lista de valores por los puntos indicados	Los mismos puntos que la serie de datos original pero separados en distintos arrays	Valores de la serie solapados e incorrectos en el resultado final	INCORRECTO (cod. 1.2)
Comprobación del script que segmenta una lista de valores por los puntos indicados	Los mismos puntos que la serie de datos original pero separados en distintos arrays	Valores segmentados correctamente, todos los valores originales pero separados	CORRECTO

Continúa en la siguiente página

Continuación de la Tabla D.1

Descripción	Salida esperada	Salida real	Resultado
Script que recorre el sistema de ficheros leyendo todas las series de datos alojadas	El programa devuelve la lista de ficheros en formato csv encontrados en la ruta indicada	Devuelve el resultado esperado	CORRECTO
Comprobación del script que recibe el parámetro para segmentar y el algoritmo con el que reducir	Resultado del error final en la reducción tanto de la serie segmentada como de la serie comleta. Deberá ser el mismo que haciendo el cálculo a mano ejecutando de uno en uno las segmentaciones y las reducciones	Resultados inconsistentes, con los mismos valores de entrada no devuelve los mismos valores de salida	INCORRECTO (cod. 1.3)
Comprobación del script que recibe el parámetro para segmentar y el algoritmo con el que reducir	Resultado del error final en la reducción tanto de la serie segmentada como de la serie completa. Deberá ser el mismo que haciendo el cálculo a mano ejecutando de uno en uno las segmentaciones y las reducciones	Devuelve el resultado esperado	CORRECTO

Análisis de las pruebas erróneas

- **1.1.** No se había cargado la serie con el tipo de datos de R correcto. Se obtuvieron los datos pero con otro formato. Una vez se entendió el formato se obtuvo sin problemas.
- **1.2.** A la hora de segmentar los valores en el script se solapaban, esto ocurría porque el bucle que los iba segmentando trataba mal los puntos solapaba datos.
- **1.3.** El script creado no trataba de manera correcta los resultados de la reducción de cada segmento, no realizaba los cálculos correctos. Una vez revisado su funcionamiento fue el adecuado.

D.2. Pruebas sobre la implementación del servicio web

Aquí se presentan las pruebas realizadas durane el desarrollo del servicio web de segmentación de series.

Tabla D.2: Pruebas realizadas durante el desarrollo del servicio web de segmentación.

Descripción	Salida esperada	Salida real	Resultado
Comprobar puesta en marcha del servicio web	Al enviar una petición con la url *ip/-prueba/* debería devolver un mensaje de repuesta	Devuelve el mensaje esperado	CORRECTO
Comprobar que el algoritmo de segmentación añadido al servicio funciona correctamente	Devuelve la serie recibida segmentada según lo indicado por el algoritmo, debe devolver el mismo valor que realizando la segmentación de manera local	Devuelve el resultado esperado	CORRECTO

Continúa en la siguiente página

Continuación de la Tabla D.2

Comprobar que el algoritmo de segmentación añadido al servicio funciona correctamente, concretamente en el Matrix Profile	Devuelve la serie recibida segmentada según lo indicado por el algoritmo, debe devolver el mismo valor que realizando la segmentación de manera local	Devuelve un error y no termina la ejecución	INCORRECTO (cod. 2.1)
Comprobar que el algoritmo de segmentación añadido al servicio funciona correctamente, concretamente en el Matrix Profile	Devuelve la serie recibida segmentada según lo indicado por el algoritmo, debe devolver el mismo valor que realizando la segmentación de manera local	Devuelve el resultado esperado	CORRECTO
Comprobación del funcionamiento del servicio una vez implantado en la máquina de Google Cloud (los tres algoritmos)	Devuelve la serie recibida segmentada según lo indicado por el algoritmo, debe devolver el mismo valor que realizando la segmentación de manera local	Devuelve un error de conexión debido a permisos tipo CORS[1]	INCORRECTO (cod. 2.2)

Continúa en la siguiente página

[1]CORS: https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS

Continuación de la Tabla D.2

Comprobación del funcionamiento del servicio una vez implantado en la máquina de Google Cloud (los tres algoritmos)	Devuelve la serie recibida segmentada según lo indicado por el algoritmo, debe devolver el mismo valor que realizando la segmentación de manera local	Se obtienen los resultados esperados	CORRECTO

Análisis de las pruebas erróneas

- **2.1.** La implementación del algoritmo Matrix Profile no era correcta a la hora de subirla a la aplicación Django del servicio. Se detectó el error y la siguiente prueba fue correcta.
- **2.2.** No estaba configurada la seguridad con CORS para que pudiese ser accedido. Se configuró en la aplicación de Django lo necesario para funcionar con CORS y a partir de ahí el servicio funcionaba puclicado online.

Bibliografía

[Agrawal et al., 1993] Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient similarity search in sequence databases. In Lomet, D. B., editor, *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Cai and Ng, 2004] Cai, Y. and Ng, R. (2004). Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 599–610, New York, NY, USA. ACM.

[Cover and Hart, 2018] Cover, T. M. and Hart, P. E. (2018). Nearest Neighbor Pattern Classification. pages 1–6.

[Dau et al., 2018] Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2018). The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[Fu-Lai Chung et al., 2004] Fu-Lai Chung, Tak-Chung Fu, Ng, V., and Luk, R. W. P. (2004). An evolutionary approach to pattern-based time series segmentation. *IEEE Transactions on Evolutionary Computation*, 8(5):471–489.

[Gharghabi et al., 2017] Gharghabi, S., Ding, Y., Yeh, C. M., Kamgar, K., Ulanova, L., and Keogh, E. (2017). Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 117–126.

[Hallac et al., 2019] Hallac, D., Nystrup, P., and Boyd, S. (2019). Greedy gaussian segmentation of multivariate time series. *Advances in Data Analysis and Classification*, 13(3):727–751.

[Li et al., 2019] Li, G., Tan, J., and Chaudhry, S. S. (2019). Industry 4.0 and big data innovations. *Enterprise IS*, 13(2):145–147.

[Madden, 2012] Madden, S. (2012). From databases to big data. *IEEE Internet Computing*, 16(3):4–6.

[Robinson and Cherry, 1967] Robinson, A. H. and Cherry, C. (1967). Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, 55(3):356–364.