UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

HERRAMIENTA DE ANÁLISIS DEL TIEMPO DE SUEÑO MEDIANTE ACELERÓMETROS PARA LA VALORACIÓN DEL SEDENTARISMO EN ADULTOS MAYORES

Carlos Alberto Ávila Atanasio
Tutor: David Martínez Gómez
Ponente: Jorge E. López de Vergara Méndez

MAYO 2017

HERRAMIENTA DE ANÁLISIS DEL TIEMPO DE SUEÑO MEDIANTE ACELERÓMETROS PARA LA VALORACIÓN DEL SEDENTARISMO EN ADULTOS MAYORES

AUTOR: Carlos Alberto Ávila Atanasio
TUTOR: David Martínez Gómez

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2017

Resumen

Mantener una vida saludable y equilibrada se ha convertido en los últimos años en un aspecto fundamental en las sociedades actuales. Conocer cómo afecta una vida de sedentarismo a nuestro organismo es un aspecto en el que muchos estudios se centran en la actualidad, en especial sus efectos en la aparición de enfermedades.

Este parámetro de estilo de vida debe ser medido con precisión, para lo que se hace necesario el uso de acelerómetros en estudios en los que el tiempo de medición sea de 24 horas seguidas.

Uno de los problemas que más dificulta la detección y análisis de este tiempo de sedentarismo es la presencia de los períodos de sueño. Aunque no forman parte del sedentarismo, por sus características se pueden confundir.

Esto hace que sea necesario encontrar métodos efectivos para extraer el tiempo de sueño de la actividad de una persona y diferenciarlo del tiempo de sedentarismo.

Además deben ser tenidos en cuenta la gran cantidad de datos a procesar, del orden de millones de registros para un solo individuo, por lo que se nos plantea la necesidad de crear un programa eficiente.

Dado que en estos estudios se basarán los futuros avances médicos, es necesario que las herramientas desarrolladas no sólo detecten este tiempo de sueño, sino que sean lo más precisas y contengan el menor error posible.

Este estudio permitirá desarrollar esta aplicación como parte de un proyecto más amplio, que busca mejorar la calidad de vida de las personas.

Palabras clave

Sedentarismo, sueño, acelerómetro, acelerometría, algoritmo, datos, fichero, patrones, Inteligencia Artificial, tipos de actividad, Winkler, Berg

Abstract

Maintaining a healthy and balanced lifestyle has become recently a main concern in today's societies. Knowing how a sedentary life affects our body is an issue many studies are focused in on the present, especially how it allows the appearance of diseases.

This parameter of lifestyle must be accurately measured, to do so it is necessary the use of accelerometers in studies in which the research time is 24 hours.

One of the issues that difficuts the most the detection and analysis of this sedentary time is the presence of sleeping periods. Although they are not part of the sedentary time, can be wrongly identified as such.

Due to this, it is necessary to find more effective methods of extracting the sleeping periods of time from a person's activities, so that we are able to differentiate it from the sedentary time.

In addition, the large amounts of data to be processed, in the order of millions of records for a single person, must be taken into account, so we need to create an efficient program.

Given that the future medical advances will be based in these research studies, it is compulsory that the new tools that are being developed not only are able to detect the sleeping patterns but are as accurate as possible.

This study will allow to develop this application as part of a larger project, which seeks to improve the quality of life of people.

Keywords

Sedentarism, sleep, accelerometer, accelerometry, algorithm, data, file, patterns, Artificial Intelligence, activity types, Winkler, Berg

Agradecimientos

A mi familia y a todos aquellos que me ayudaron a llegar hasta aquí.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

Glosario

TFG	Trabajo de fin de grado
Acelerómetro	Sistema o conjunto de sistemas orientado a conocer la posición de un sujeto, velocidad, portencia… y con ello su actividad física
Interfaz	Medio a través del cual el usuario puede establecer comunicación con un sistema informático
Java	Lenguaje de programación orientado a objetos, que fue desarrollado en 1995.
Weka	Software libre orientado al aprendizaje automático y a la minería de datos. Está escrito en lenguaje Java y ha sido desarrollado por la Universidad de Waikato, en Nueva Zelanda
Hibernate	Framework para la plataforma Java que agiliza la relación entre la aplicación y la base de datos

1 Introducción

1.1 Motivación

Este trabajo de Fin de Grado se engloba dentro de las actividades desarrolladas como parte del proyecto realizado en la facultad de Profesorado, concretamente en el Departamento de Educación Física, Deporte y Motricidad Humana, para el análisis del tiempo de sueño mediante acelerómetros, con el fin de valorar el sedentarismo en adultos mayores.

Para la realización de este estudio, se ha precisado el desarrollo de un sistema que analice los datos extraídos de los acelerómetros llevados por los participantes, y que posteriormente calcule el llamado tiempo de sueño (o de "luces fuera"), además de los datos detallados de este tiempo, ya que para el correcto estudio del sedentarismo es necesario diferenciar el tiempo de sedentarismo como tal y el tiempo de sueño. Un segundo objetivo planteado es encontrar el método (algoritmo) o los métodos que satisfagan los requisitos solicitados con el menor error relativo posible.

Por otra parte, con este TFG surge la oportunidad de trabajar en un proyecto en un entorno real, donde poder poner en práctica lo aprendido a lo largo de la carrera, tanto a nivel de algoritmia/programación (desarrollando herramientas basadas en el uso de métodos aprendidos en la carrera, entre ellos de Inteligencia Artificial), como a nivel de Ingeniería del Software, pasando por todas las fases del mismo (análisis, diseño, implementación, validación y pruebas).

1.2 Objetivos

El propósito de las tareas a desarrollar como parte del Trabajo de Fin de Grado es la implementación de una herramienta óptima para la obtención del tiempo de sueño en los participantes a analizar en el estudio. En particular, se han agrupado los objetivos en:

- Implementación de un sistema informático que sea capaz de devolver el tiempo de sueño o "luces fuera" de los participantes estudiados.
- Conseguir la o las herramientas más precisas posibles (con menor error relativo) para llevar a cabo esta tarea.

1.3 Fases de realización

Con el fin de poder alcanzar el cumplimiento de los objetivos marcados en el apartado anterior, se ha pasado por una serie de etapas que se encuentran recogidas en esta memoria.

- **Documentación**

Esta fase ha consistido en informarse para conocer el acelerómetro y su programa auxiliar de extracción de datos. También se ha recabado información sobre los métodos existentes de cálculo del tiempo de sueño.

- **Análisis y diseño**

Mientras que se llevaba a cabo la fase de documentación, se ha llevado acabo el análisis de los requisitos para la implementación de la herramienta. También se realiza el diseño de la aplicación, basándose en los algoritmos previamente estudiados.

- **Desarrollo**

Una vez llevada a cabo la fase de análisis y diseño de la aplicación, se inicia la fase de desarrollo del sistema informático, bajo los requisitos definidos anteriormente.

- **Validación**

Una vez desarrollada la herramienta, se realizan las comprobaciones oportunas para dejarla lista para su posterior prueba con datos reales.

- **Pruebas**

Tras la fase de validación se procede a comprobar su correcto funcionamiento mediante un conjunto de datos de prueba reales.

- **Redacción de la memoria**

Finalmente se redacta esta memoria, con el fin de recoger detalladamente todo el proceso para cumplir los objetivos marcados en el TFG.

A continuación se muestra el diagrama de Gantt con las fases del proceso, desde su inicio hasta la propia redacción de esta memoria:

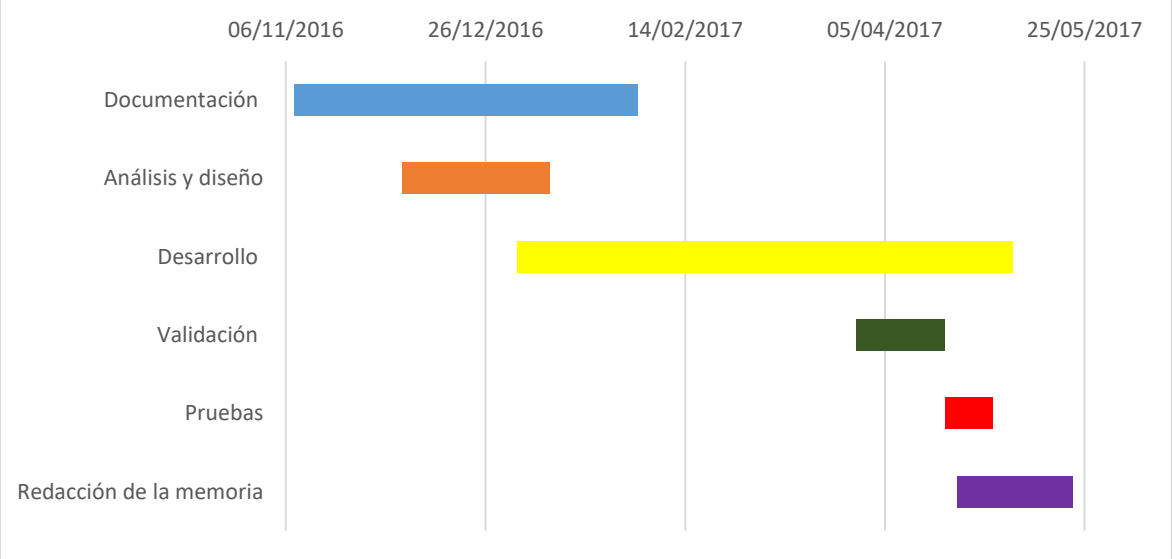


Ilustración 1. Diagrama de Gantt

1.4 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** en este apartado hablaremos de los sistemas actuales relacionados con el proyecto, los métodos y dispositivos para detectar el movimiento, los acelerómetros y los diferentes algoritmos existentes para la detección del tiempo de sueño.

- **Análisis:** en este apartado trataremos del objetivo del programa, así como se describirán los requisitos funcionales, de datos, de interfaz y encontramos el diagrama de Casos de Uso.

- **Diseño:** en este punto se explicará la arquitectura del sistema informático a implementar, los elementos de su arquitectura, podremos encontrar el diagrama general del procesamiento, el de clases, el de entidad relación, se detallará la carga inicial de datos, la salida de datos y el entregable final para el usuario.

- **Desarrollo:** en esta parte trataremos las decisiones tomadas durante la implementación de la aplicación, el procesado de datos (detallando los diferentes métodos implementados y los algoritmos de IA empleados).

- **Integración, pruebas y resultados:** en esta sección se detallarán los datos de entrada y de entrenamiento (en el caso del Método 3), el entorno de pruebas empleado, las pruebas realizadas y las conclusiones alcanzadas.

- **Conclusiones y trabajo futuro:** en este último apartado se explicarán las conclusiones generales del TFG y el trabajo futuro de este.

2 Estado del arte

2.1 Introducción

En este capítulo trataremos el estado actual de los sistemas relacionados con el proyecto, como son los dispositivos de detección del movimiento, los acelerómetros y los diferentes algoritmos para la detección del tiempo de sueño. Es fundamental conocer el estado de arte para saber en qué contexto nos encontramos y en qué basarnos para construir la herramienta.

2.2 Sistemas relacionados

La medición de la actividad física y el sedentarismo es un reto en nuestra sociedad, siendo su conocimiento algo imprescindible para definir políticas públicas en el ámbito de las ciencias de la salud. Se considera sedentarismo a cualquier actividad realizada por el individuo en posición sentada o inclinada con un gasto energético ≤1.5 METs, mientras está despierto [9].

Actualmente, los acelerómetros existentes, por ejemplo, en teléfonos móviles o en pulseras inteligentes, permiten valorar estas conductas de forma precisa. Sin embargo, al utilizar un acelerómetro durante 24 horas, se deberá distinguir el tiempo de sueño y sedentarismo, en los cuales apenas hay movimiento corporal, para una mejor valoración. Para la identificación del sueño se necesitan desarrollar algoritmos que detecten de forma automática cada patrón de comportamiento sobre medidas que abarquen 24 horas.

En la actualidad existen estudios relacionados con la obtención del tiempo de sueño o "luces fuera" y la diferenciación de este respecto al tiempo de sedentarismo, adaptados a los datos procedentes de otros tipos de acelerómetros distintos al que se usará en este estudio.

2.2.1 Métodos y dispositivos para la detección el movimiento y su monitorización

Un dispositivo de detección del movimiento, o sensor de presencia, es un dispositivo electrónico que contiene una serie de sensores capaces de dar una respuesta determinada ante un movimiento físico. Generalmente se encuentran en sistemas de seguridad o circuitos cerrados de televisión.

Para esta detección del movimiento pueden existir numerosas motivaciones, ya sean para seguridad o iluminación, como para ocio o simple comodidad. Un ejemplo sería un sistema de detección en el rellano de una vivienda, que ilumina la zona cuando detecta personas.

Existen en el mercado diversos tipos de sensores, entre los que destacan:

- **Sensores activos:** Estos detectores inyectan luz, sonido o microondas en la zona y detectan los cambios alrededor.

- **Sensores pasivos**: Muchos de estos elementos usan ondas infrarrojas, por lo que son conocidos como PIR (Pasivos Infrarrojos), que deben ser calibrados para detectar la temperatura del ser humano.

- **Acelerómetro:** Se trata de uno de los sensores más versátiles, siendo el más común el llamado "piezoeléctrico por compresión". Trataremos de ellos más adelante, ya que son el objetivo de nuestro trabajo.

- **Giroscopio mecánico:** Este tipo de dispositivos miden la orientación, consistiendo en un disco giratorio, que cambia de orientación a causa de las fuerzas externas provocadas por el movimiento.

2.2.2 Acelerómetros: ¿qué son y cómo funcionan?

Los acelerómetros (también llamados sensores de aceleración) son dispositivos pensados para llevar a cabo mediciones de la aceleración o de la vibración, proporcionando una señal de tipo eléctrico según la variación física, siendo la aceleración o la vibración en este caso esa variación física. Existen diferentes tipos combinando las diferentes tecnologías existentes, principalmente los acelerómetros piezoresistivos, acelerómetros piezoeléctricos y acelerómetros capacitivos.

Los rangos de medida pueden ser diversos: en cuanto al peso desde 1 gramo hasta varios kilogramos; en cuanto a las frecuencias de vibración, desde los 0 Hz, para los acelerómetros de alta sensibilidad, hasta los miles de Hz.

Algo similar ocurre con los formatos, que son variados dependiendo de la aplicación del acelerómetro: aluminio, acero inoxidable, titanio, con tornillos, magnético, pegado…

Combinando las diferentes tecnologías, funcionalidades, formatos, etc. Se pueden clasificar en seis grandes familias:

Acelerómetros piezoresistivos

Esta familia está compuesta por los acelerómetros DC o acelerómetros MEMS. Su principal característica es que miden desde los 0 Hz hasta varios de cientos de Hz, yendo desde los 2 gramos hasta los 6 Kg.

Acelerómetros piezoeléctricos

Básicamente son los acelerómetros de METRA o acelerómetros IEPE, también conocidos comercialmente como acelerómetros ICP. Se trata de dispositivos para la medida de medias y altas frecuencias. Están compuestos de un material piezoeléctrico que, al ser accionado mecánicamente, proporciona una tensión pequeña, proporcional a ese movimiento que lo ha provocado.

Acelerómetros capacitivos

Este tipo de acelerómetros son capaces de medir desde 0 Hz hasta varios cientos de Hz, por lo que se suelen utilizar para bajas o muy bajas frecuencias, aunque poseen una gran resistencia, lo que permite salir airosos de posibles picos de aceleración.

Acelerómetros triaxiales

Los acelerómetros triaxiales, o de tres ejes, se emplean para medir la aceleración y vibración en diferentes sectores. Lo que les diferencia de los monoaxiales es que se puede medir, con el mismo elemento, la aceleración o vibración en 3 coordenadas, mientras que los monoaxiales solo permiten medir en una coordenada.

Acelerómetros ATEX

Este tipo de acelerómetros, se instalan en zonas donde se aplica la normativa EX, como la industria del carbón o la petroquímica. Ofrecen una salida 4 a 20 mA proporcional al nivel de vibración medida.

Acelerómetros OEM

Los acelerómetros OEM o acelerómetros de bajo coste, están orientados a la integración de diseños propios, gracias a sus formatos en miniatura para soldar mediante terminales.

En el estudio del cual se procesarán los datos en este proyecto se ha seleccionado un dispositivo inteligente para la estimación de gasto de energía y actividad, llamado *Intelligence Device for Energy Expenditure and Activity* (IDEEA).

Se trata de un nuevo microprocesador portátil de actividad física, diseñado con el fin registrar los movimientos complejos de la actividad física diaria empleando sensores piezoeléctricos en el pecho, muslos y pies.

Esta herramienta emplea sensores de movimiento de pequeño tamaño, con un peso inferior a un gramo, que se ponen con cinta hipoalérgica en cinco localizaciones: el pecho (esternón superior), la mitad del muslo en ambas piernas y los dos pies. La escala de los sensores es de ± 2 G.

Además lleva una minicomputadora (de 59 gramos y de peso) en la cintura donde a la que se conectan tres cables delgados y flexibles. De esta manera es capaz de diferenciar entre posturas sedentarias y movimientos activos.

La validación clínica de IDEEA ha sido publicada en 2010 por el New York Obesity Research Center, Universidad de Columbia sobre Investigación en obesidad [10]. El IDEEA es capaz de grabar y reproducir los cambios de movimiento y de postura del cuerpo sobre una base de 24h, almacenando decenas de millones de datos para futuros análisis. También identifica más de 40 tipos de actividad física, incluyendo estar acostado, estar sentado, caminar, subir escaleras, correr y saltar.

Ilustración 2. Acelerómetro IDEEA

2.2.3 Algoritmos existentes en la actualidad para la detección del tiempo de sueño

A continuación se describirán los dos algoritmos más utilizados en la detección del tiempo de sueño.

2.2.3.1 Winkler

Este algoritmo, "Identifying adults' valid waking wear time by automated estimation in activPAL data collected with a 24 h wear protocol", está diseñado para una colección de datos de 24 horas, obtenidos del acelerómetro activPAL, y ha sido diseñado por diversos expertos de Australia.

Las principales características de este algoritmo automático son:

- Los participantes llevaron el acelerómetro activPAL 24 seguidas durante 10 días.
- Los participantes rellenaron un diario en el que apuntaban las horas en las que se acostaban, se dormían, se despertaban, se levantaban de la cama, y también cuándo se quitaban el acelerómetro.
- El algoritmo trata de identificar los tiempos en determinadas posiciones para suprimirlo del tiempo activo. Primero detecta el tiempo de "sueño o no-llevado" y luego dentro de estos periodos diferencia entre el tiempo no-llevado y el tiempo de sueño.

2.2.3.2 Berg

Este método, "Identifying walking time in 24-h accelerometry data in adults using an automated algorithm", ha sido desarrollado en la Universidad de Maastricht (Paises Bajos).

Las principales características de este algoritmo automático son:

- Identificación de los tiempos de sueño y despierto.

- Algoritmo con criterios de búsqueda o identificación de patrones (standing/stepping o sitting/lying) dentro de un periodo de tiempo determinado. Una vez ha encontrado estos patrones, el algoritmo suma diversos tiempos permanecidos en estas posiciones.

- Creación y establecimiento de unos puntos de corte para la comparación con los tiempos obtenidos en la identificación de los patrones activos y sedentarios y establecimiento de la hora de inicio de sueño y la hora de inicio despierto.

2.2.4 Datos del estudio

La aplicación informática que nos ocupa se encuentra en el marco de un estudio general sobre el sedentarismo de las personas mayores. Para ello, se ha asignado a cada uno de los participantes del estudio (personas de avanzada edad, de más de 65 años) un acelerómetro de tipo IDEEA, que deben llevar puesto durante un total de 48 horas.

Estas personas informarán a los responsables del estudio en el momento de la recogida sobre la hora a la que se fueron a dormir y la hora a la que se despertaron durante los dos días que llevaron puesto el acelerómetro.

Una vez se tengan estos datos, se descargarán del acelerómetro y se pre-procesarán con el programa ActView de IDEEA, para convertirlos en datos legibles para el estudio. Esto da como resultado unos ficheros de texto (uno por sujeto), en el que cada línea corresponde a una fracción de segundo (1/32 s^{-1} en nuestro caso) de la actividad del sujeto a analizar: el tick (32 por cada segundo), el tipo de actividad (Ver *Anexo C: Tipos de actividad*), la velocidad, la potencia y el EE.

Además, estos ficheros contienen una cabecera. En esta parte, se pueden distinguir la dirección del fichero note.txt (información adicional) y dos números (el segundo indica la hora de inicio, en el caso del ejemplo 112432 – 11:24:32 h).

```
=======================Data Output From ActView(TM),MiniSun LLC ========================

For info see: C:\prueba\1233M32_1dia1._note.txt

160428  112432
Ticks   Activity Type   Speed   Power   EE

       0        120     0.019   0.193   1.236
       2        120     0.019   0.193   1.236
       4        120     0.019   0.193   1.236
       6        120     0.019   0.193   1.236
       8        120     0.019   0.193   1.236
      10        120     0.019   0.193   1.236
      12        120     0.019   0.193   1.236
      14        120     0.019   0.193   1.236
      16        120     0.019   0.193   1.236
      18        120     0.019   0.193   1.236
      20        120     0.019   0.193   1.236
      22        120     0.019   0.193   1.236
      24        120     0.019   0.193   1.236
      26        120     0.019   0.193   1.236
      28        120     0.019   0.193   1.236
      30        120     0.019   0.193   1.236
      32        120     0.019   0.193   1.236
      34        120     0.019   0.193   1.236
      36        120     0.019   0.193   1.236
      38        120     0.019   0.193   1.236
```

Ilustración 3. Fichero de entrada

Posteriormente, estos datos se emplearán como entrada del sistema a implementar como parte de este TFG.

2.3 Conclusiones

Esta sección de la memoria nos ha permitido conocer la realidad actual de los acelerómetros, concretamente el sistema IDEEA que emplearemos en el estudio, además de analizar los datos que emplearemos y los algoritmos más importantes para el cálculo del tiempo del sueño.

Estos dos algoritmos van a ser parte muy importante de nuestra aplicación, ya que para desarrollar nuestros primeros métodos nos basaremos en ellos.

3 Análisis

3.1 Introducción

En esta parte de la memoria trataremos sobre los requisitos solicitados para el sistema a implementar (funcionales, de datos y de interfaz) y el diagrama de casos de uso. Un análisis claro y detallado es fundamental para un buen desarrollo posterior de la aplicación.

3.2 Objetivo del programa

Esta aplicación analizará los datos obtenidos a través de un acelerómetro portado por los participantes del estudio, que estarán almacenados en formato de archivo de texto en una carpeta accesible por el programa, calculando a partir de ellos el tiempo de sueño o "luces fuera" mediante la utilización de distintos algoritmos (incluidos de Inteligencia Artificial).

También se extraerán los períodos correspondientes al tiempo de sueño o "luces fuera" previamente calculado, devolviéndose los demás periodos (tiempo de "no sueño") en el mismo formato del fichero de origen de los datos, a una carpeta previamente indicada por el usuario del programa.

Como objetivo adicional, se analizará el funcionamiento y precisión de estos algoritmos con el fin de detectar e incluir en el programa aquellos que sean más eficientes.

3.3 Definición de requisitos

En este apartado trataremos el análisis de los requisitos solicitados para el sistema a implementar, obtenidos en las reuniones iniciales del proyecto.

3.3.1 Requisitos funcionales

3.3.1.1 Introducir carpeta de origen

Introducir aquella carpeta de origen de los datos que se querrán procesar.

3.3.1.2 Aviso por campo vacío no válido al introducir carpeta origen

Aviso si el campo destinado para que el usuario introduzca la carpeta de origen de datos está vacía o si el nombre de esta no es válido.

3.3.1.3 Introducir carpeta de destino

Introducir aquella carpeta de destino de los datos procesados

3.3.1.4 Aviso por campo vacío o no válido al introducir carpeta destino

Aviso de si el campo destinado para que el usuario introduzca la carpeta de destino de datos está vacía o si el nombre de esta no es válido.

3.3.1.5 Selección de método a emplear

Seleccionar cuál de los 3 métodos emplear para procesar los datos: Método 1 (Winkler), Método 2 (Winkler + Berg) y Método 3 (Inteligencia Artificial).

3.3.1.6 Aviso de programa procesando

Aviso para que el usuario pueda ver por pantalla que los datos se están procesando.

3.3.1.7 Mostrar los resultados por pantalla

Se mostrarán los resultados finales por usuario en la pantalla, para que el usuario de la herramienta pueda analizarlos.

3.3.2 Requisitos de datos

3.3.2.1 Origen de datos

Los datos se obtendrán de una carpeta en la que previamente se habrán almacenado los ficheros de tipo ".txt" con los datos de cada sujeto a analizar en el estudio. Cada uno de estos ficheros tendrá el formato proporcionado por el programa ActView de IDEEA en su salida tras pre-procesar los datos.

3.3.2.2 Destino de datos

Los datos ya procesados se devolverán a una carpeta destino introducida previamente por el usuario. En dicha carpeta se volcará un archivo tipo ".txt" por cada sujeto del estudio, que contendrá los periodos de este que correspondan a "no sueño", con el mismo formato que el fichero de origen.

3.3.2.3 Tipo de actividad

La aplicación a desarrollar reconocerá un total de 51 tipos de actividad diferentes, a los que se asignará un código entre el 99 y el 183, que será el mismo que asigna el acelerómetro a los datos y que se obtiene tras pre-procesar los mismos a través del ActView de IDEEA (Ver *Anexo C: Tipos de actividad*).

3.3.3 Requisitos de interfaz

3.3.3.1 Introducción de datos y elección de método

Interfaz para que los usuarios puedan introducir las carpetas de origen y destino de los datos, además de elegir el método a usar para el procesado de los mismos: *Winkler*, *Berg* e *Inteligencia Artificial*.

3.3.3.2 Usabilidad

La aplicación ha de ser suficientemente intuitiva para que pueda ser utilizada por usuarios sin conocimientos informáticos elevados. Para ello, se ha optado por una interfaz gráfica

simple, con botones no demasiado pequeños y lo suficientemente descriptivos como para que el usuario pueda saber intuitivamente para qué sirven.

3.3.3.3 Uso del ratón

A la hora de manejar el sistema se hará con el ratón para seleccionar las opciones requeridas, usando el teclado para introducir los datos.

3.4 Casos de Uso

A continuación se mostrará el diagrama de casos de uso, pieza clave en esta etapa de diseño.

3.4.1 Diagrama de Casos de Uso

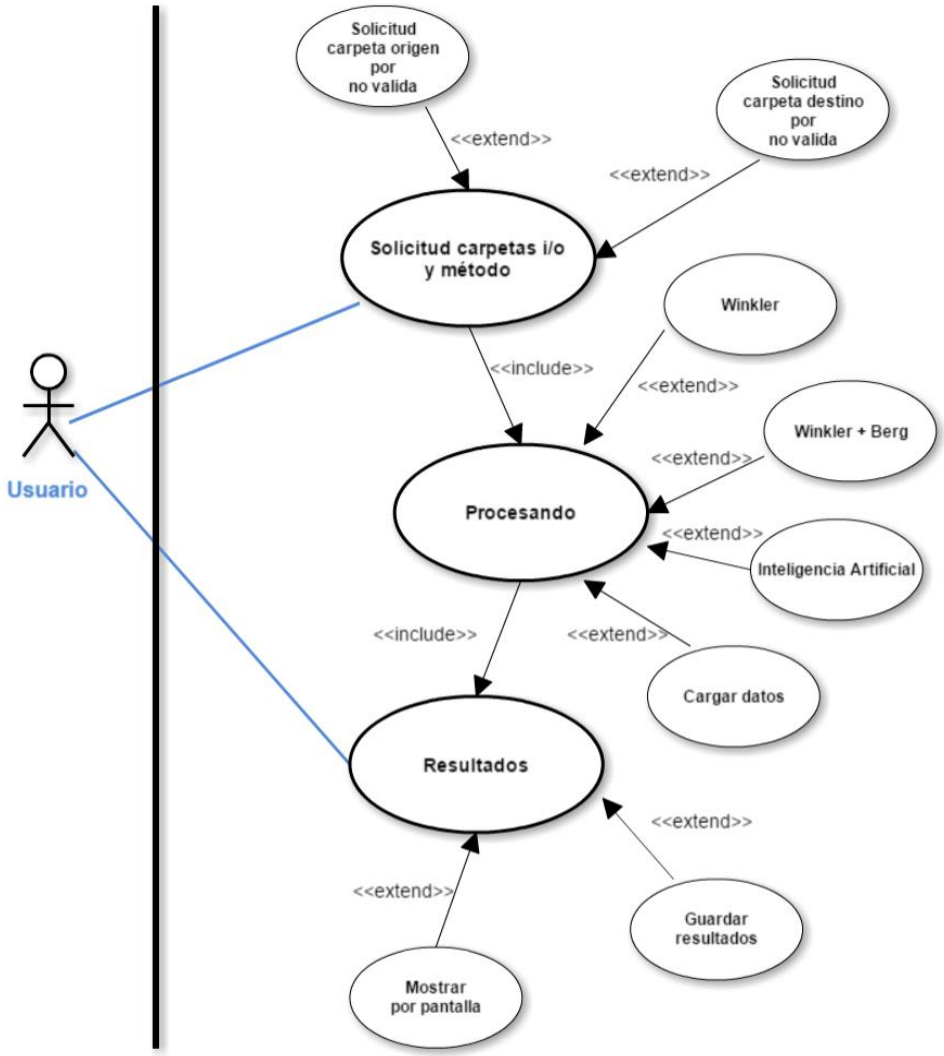


Ilustración 4. Diagrama de Casos de Uso

3.5 Conclusiones

En este capítulo se han recogido los requisitos del sistema informático que se diseñará e implementará a continuación. La explicación correcta de estos requisitos es algo fundamental para cualquier proyecto, ya que sentará las bases de la herramienta.

Por otro lado, el diagrama de Casos de Uso expuesto en este capítulo nos permite conocer una primera aproximación al sistema informático, representando la forma en como nuestro actor (el usuario) opera con el sistema en desarrollo, concretamente interactuando con la parte inicial y final del proceso.

En la parte intermedia, el núcleo propiamente dicho del programa, el usuario no interactúa con él, mientras se obtienen los datos finales

4 Diseño

4.1 Introducción

En este capítulo trataremos sobre el diseño de la aplicación, concretamente la arquitectura del sistema informático a desarrollar, los elementos de la arquitectura (Spring MVC, JPA/Hibernate) y los diagramas de procesamiento, de clases y de entidad – relación. También se explicará la carga inicial de datos, la salida de estos y el entregable final.

4.2 Arquitectura de la aplicación

Tras definir los requisitos para el desarrollo de esta aplicación informática, es el momento de dar comienzo a la creación del programa solicitado, llevando a cabo para ello el diseño.

La aplicación se desarrollará bajo un entorno Web con los siguientes componentes:

- Java JDK 1.8
- Eclipse java EE for web development (Neon 3), con los siguientes plugins:
    - Spring Boot
    - Maven m2e
    - Hibernate
- Base de datos HSQLDB embebida

El sistema tendrá las siguientes características:

- Capa Web. Contendrá la capa de usuario, es decir, lo que los clientes podrán ver al ejecutarla desde su navegador web. Se utilizará para su desarrollo el Framework Java Spring MVC que, utilizando el patrón de diseño de Software: Modelo – Vista – Controlador proporciona una excelente organización del código y escalabilidad.
- Capa Servicios. Contendrá la capa de obtención de información. Su objetivo será solicitar todos los datos necesarios para, posteriormente, entregárselos a la Aplicación Web que será la encargada de su representación final. Se utilizará también el Framework Java Spring MVC para su desarrollo, y para la persistencia de datos se utilizará JPA2 e Hibernate [3].

4.3 Elementos de la arquitectura

La arquitectura base sigue el patrón de diseño MVC, que proporciona una arquitectura robusta y en la que la separación de funcionalidades es la prioridad máxima [6]:

- Modelo: gestiona el acceso a la información del sistema, generalmente almacenada en Base de Datos o algún sistema de persistencia específico.
- Controlador: gestiona las peticiones del usuario, accediendo al modelo para la

consulta de datos y ofreciéndolos a través de la vista asociada.

- Vista: representa la información obtenida al usuario.

Esta capa se implementa en el sistema mediante el uso del framework Spring.

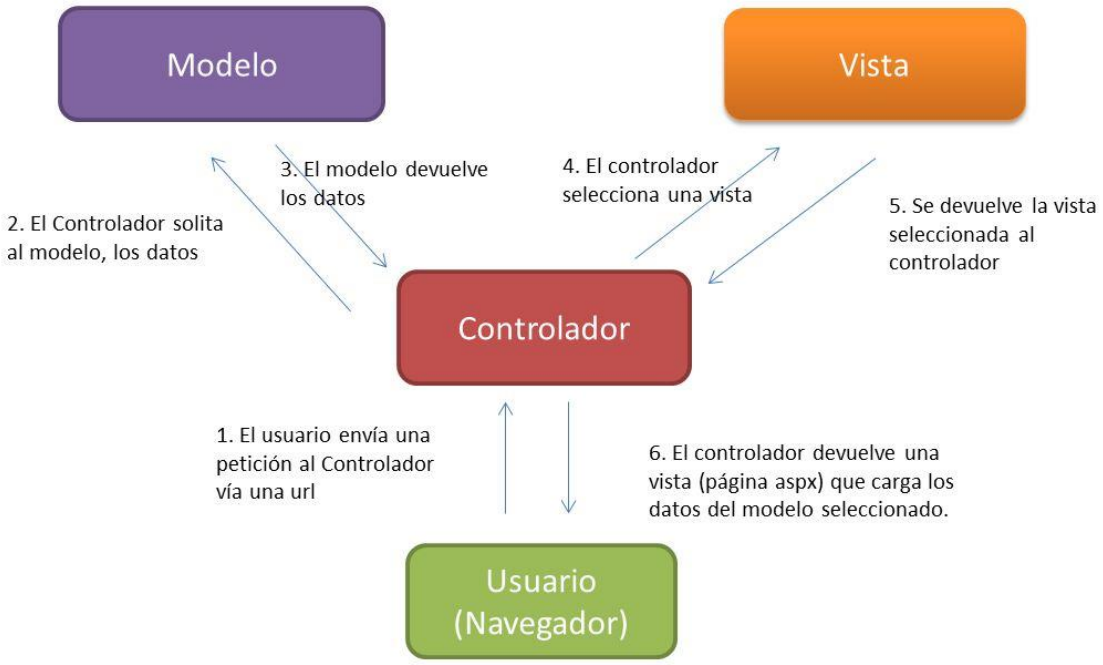


Ilustración 5. Diagrama MVC

4.3.1 Spring MVC

Se encarga de recibir las peticiones, gestionarlas y procesarlas y retornar una vista con la que pueda interactuar el cliente. Es básicamente el núcleo de la aplicación [5].

Cada controlador debe ir anotado con la anotación @Controller y se deberá mapear correctamente su url de acceso con @RequestMapping. De esta forma conseguiremos que cada petición al sistema se asocie a un determinado controlador desarrollado al efecto.

4.3.2 Persistencia con JPA/Hibernate y base de datos HSQLDB

El sistema se apoyará en JPA/Hibernate para la gestión de la persistencia de datos. Uno de los retos de la implementación de esta herramienta ha sido el almacenamiento de datos, ya que la cantidad de ellos era muy elevada. Cada sujeto a analizar (24 horas) contaba con un fichero .txt que contenía varias columnas con alrededor de 5,5 millones de filas.

Ante la necesidad de tener estos datos disponibles durante la ejecución, se decidió guardarlos en una base de datos. Tras probar con varias (HSQLDB, MySQL, H2, Oracle), se decidió emplear una de tipo HSQLDB embebida, con persistencia en disco, atendiendo a parámetros de robustez, tiempo de carga y facilidad de implementación.

HSQL se trata de una base de datos relacional programada íntegramente en Java, que se puede integrar completamente en cualquier aplicación Java y acceder a ella lanzando SQL directamente, sin necesidad de una conexión a través de sockets, como sí pasaba con MySQL, Postgress, Oracle…, aunque también es posible emplearla mediante conexiones externas JDBC.

Todo esto hace que podamos encontrarnos con una base de datos más rápida y con una mejor integración más sencilla.

4.4 Diagrama general del procesamiento

A través de este diagrama podremos visualizar el funcionamiento general del programa informático. Primeramente se crea la base de datos y se carga la información inicial.

Posteriormente se inicia un bucle en el que se lee el fichero de entrada del participante, con sus datos de actividad. A continuación se procesan estos datos y finalmente se guardan en un fichero de salida único para cada participante.

Una vez terminado el bucle, es decir, terminado el procesado de todos los participantes, se imprimen los resultados por pantalla, para que el usuario pueda ver el tiempo de sueño de cada uno.

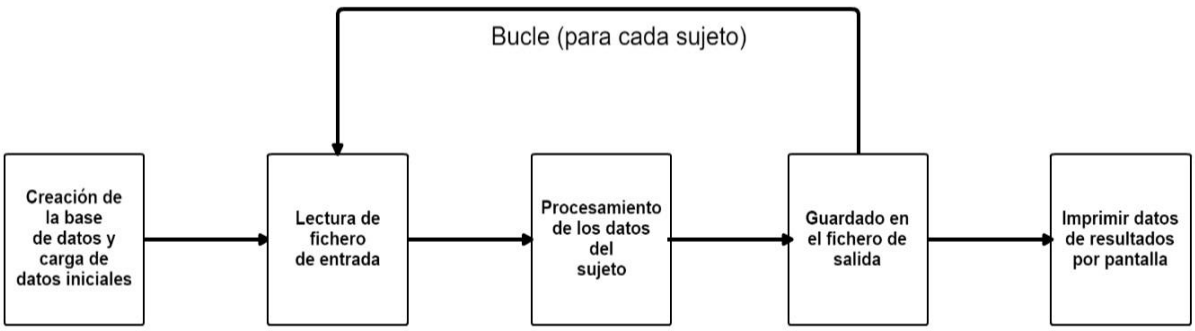


Ilustración 6. Diagrama general del procesamiento

4.5 Diagrama de clases

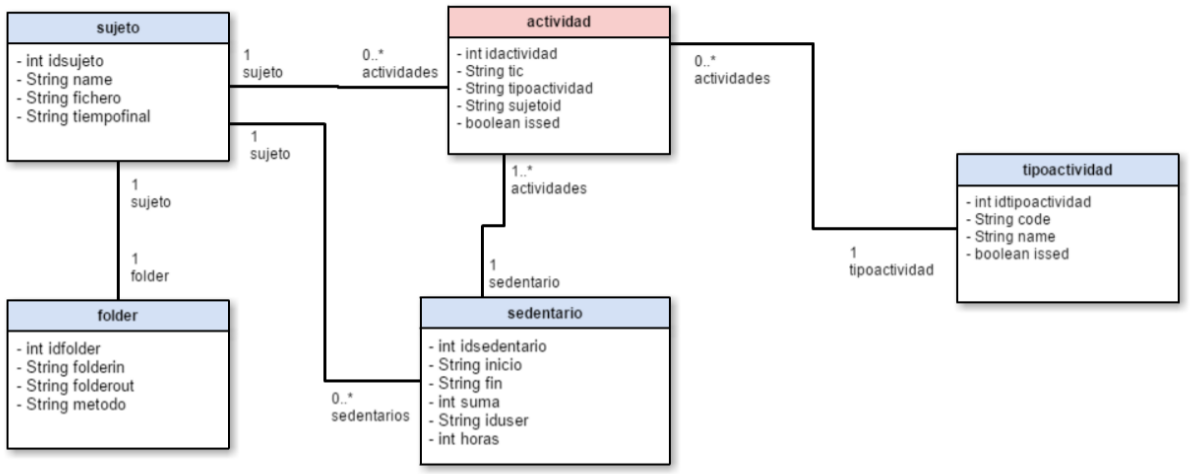


Ilustración 7. Diagrama de clases

4.6 Diagrama entidad – relación

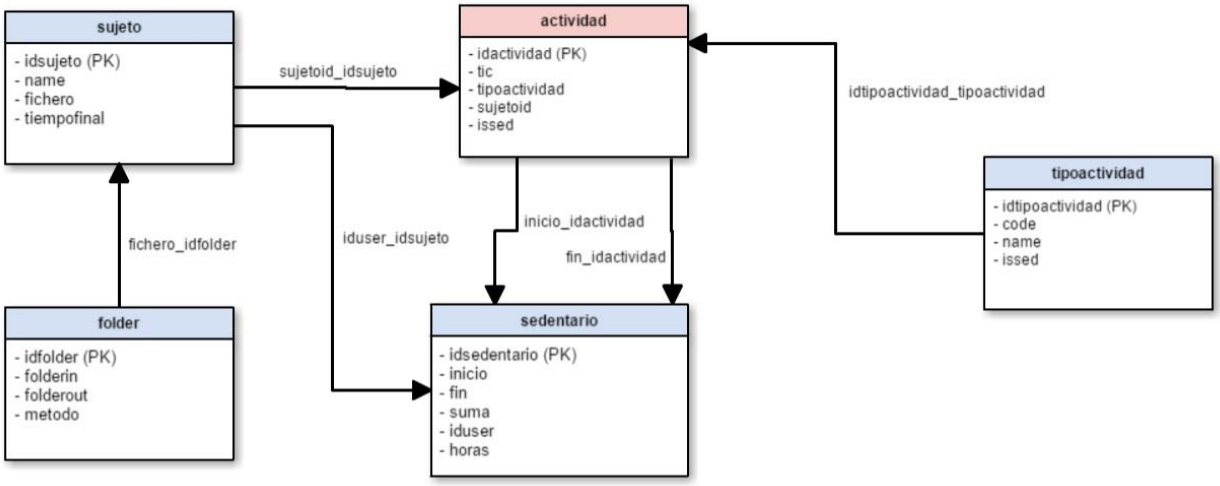


Ilustración 8. Diagrama entidad - relación

4.7 Carga inicial de datos

Para el correcto funcionamiento de la aplicación a desarrollar, será necesario realizar una carga previa de datos que contendrá los 51 tipos de actividad detallados en el *Anexo C: Tipos de actividad*.

Esta carga previa se llevará a cabo durante el inicio del programa, tras la creación de la base de datos que contendrá, entre otros, esos datos iniciales.

4.8 Salida de datos

Tras la finalización del procesamiento de los datos, la aplicación informática generará por cada sujeto analizado un fichero tipo ".txt" que será guardado en la carpeta destino especificada por el usuario. Este fichero contendrá todos los períodos de "no sueño" del sujeto correspondiente.

También se mostrará por pantalla al usuario de la herramienta una tabla con los resultados obtenidos en el procesamiento de los datos. Se podrá ver el nombre de cada participante en el estudio y las horas, minutos y segundos de tiempo de "sueño"

4.9 Entregable

Para facilitar la ejecución del programa por parte del usuario final y que este no necesite tener preinstalados más elementos de los necesarios, se entregará un fichero tipo ".war" ejecutable, que contendrá la aplicación. Solo se requerirá la instalación, en caso de ser necesario, de Java 8.

Además, con el fin de iniciar este programa y posteriormente poder detenerlo y salir, se entregarán también dos archivos llamados "start" y "stop". La finalidad es que el usuario ejecute el archivo "start" y el programa se lance automáticamente en el navegador del usuario.

4.10 Conclusiones

En este capítulo de Análisis hemos podido ver la arquitectura de la aplicación a desarrollar y sus elementos. Además, gracias a los diversos diagramas mostrados podemos ver de forma más detallada cómo será la herramienta.

La correcta descripción de los elementos en este capítulo es fundamental si queremos que el desarrollo se haga lo más sencillo posible, además de evitar con ello futuros cambios que hagan emplear más tiempo en rediseñar algunas partes de la aplicación.

5 Desarrollo

5.1 Introducción

En esta sección trataremos el proceso de implementación de nuestro sistema, describiéndose el lenguaje de programación empleado y el entorno de programación en el que hemos desarrollado la herramienta, analizando de forma detallada las decisiones que hemos tomado a la hora de implementar el código.

En cuanto al lenguaje de programación, usaremos Java: se trata de un lenguaje de alto nivel orientado a objetos.

Bajo el punto de vista de un desarrollador, para implementar un código en Java existen dos entornos de desarrollo principales: Eclipse y NetBeans. En nuestro caso, se ha seleccionado como entorno de desarrollo Eclipse, en su versión *Neon 3*.

5.2 Decisiones de implementación

Durante el desarrollo del código de la herramienta solicitada, se han tomado una serie de decisiones para cumplir con los requisitos solicitados. En este apartado se van a explicar las decisiones más importantes tomadas a lo largo de esta fase de implementación, que se han dividido según las características principales de la herramienta.

5.2.1 Interfaz

A la hora de implementar una herramienta que interactúe con el usuario, es fundamental tener en cuenta la interfaz y el sistema de ventanas.

Para este proyecto se ha optado por una serie de ventanas con botones y paneles para introducir datos, ya que el usuario solicitaba una aplicación simple, dividida en claros y sencillos pasos y bien explicada. Para comenzar, la primera ventana cuenta con una explicación del objetivo de la herramienta y el botón “Comenzar”.

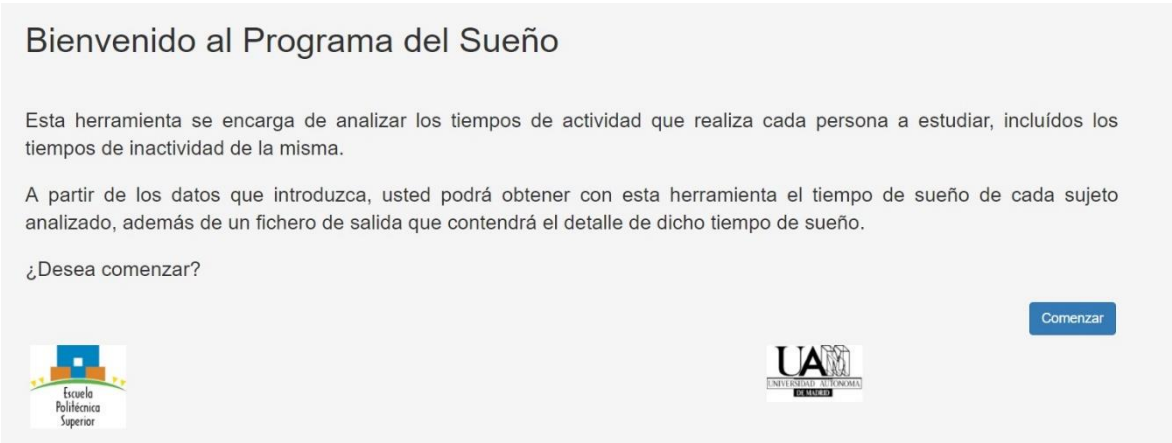


Ilustración 9. Interfaz 1

Posteriormente se accede a una ventana nueva donde se solicita al usuario que introduzca la carpeta de origen de los datos, la carpeta de destino de los datos y que seleccione el método a emplear para procesar los datos: *Winkler*, *Berg* o *IA*.

En esta última selección, y para evitar posibles errores, se selecciona por defecto el método *Winkler*. Tras presionar el botón "Continuar", pasamos a la siguiente ventana.

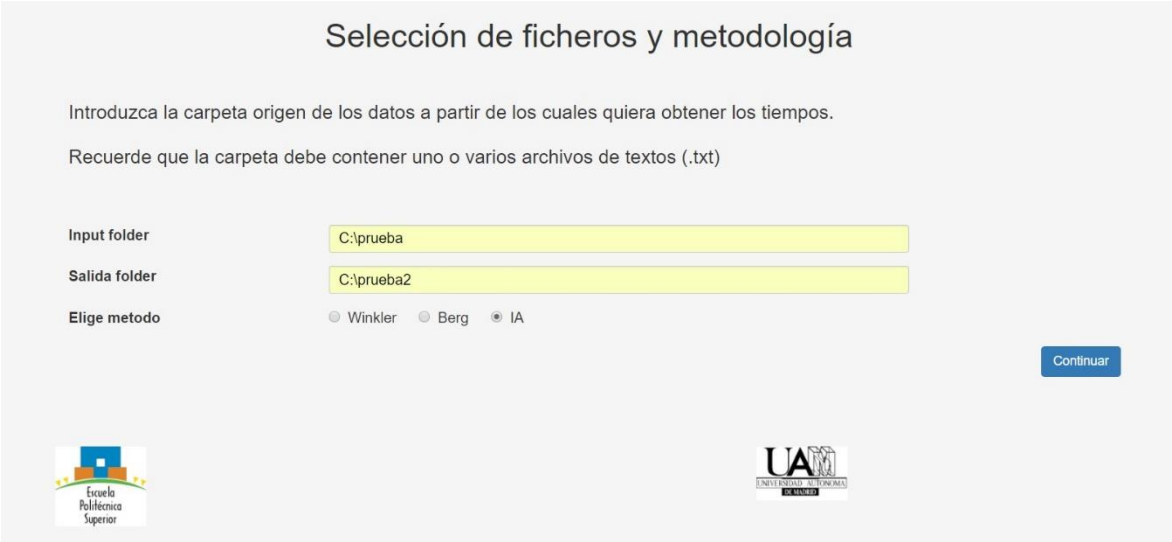


Ilustración 10. Interfaz 2

Si la ruta de la carpeta introducida no es válida o está vacía, aparece un mensaje de error.

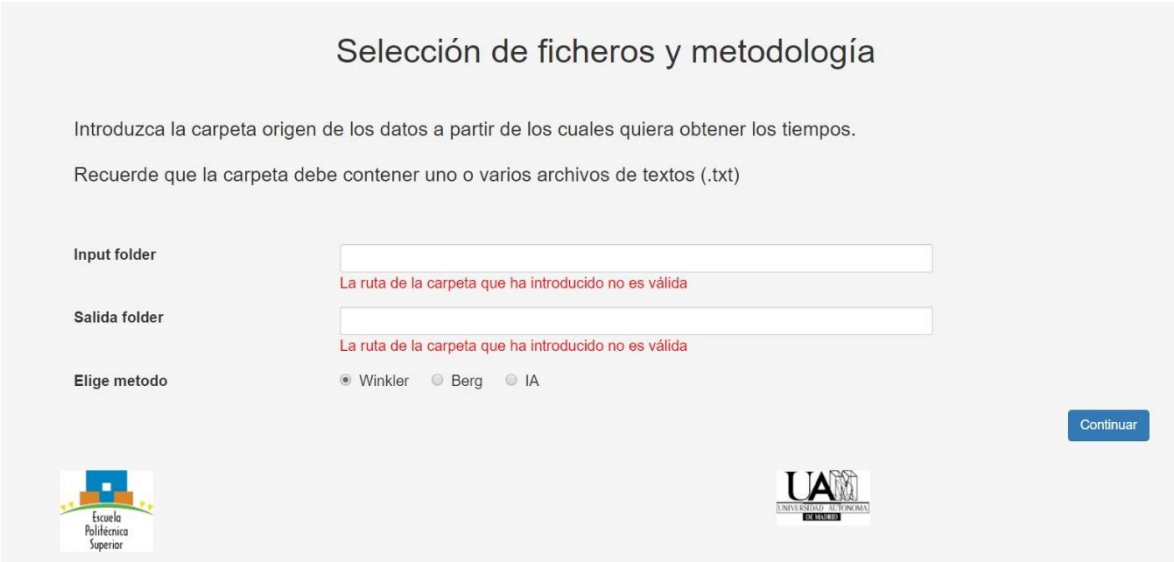


Ilustración 11. Interfaz 3

En esta nueva ventana, aparece una barra de progreso, en la que se irá rellenando de color azul y con un indicador del porcentaje (%) se podrá visualizar lo que ya se ha procesado.

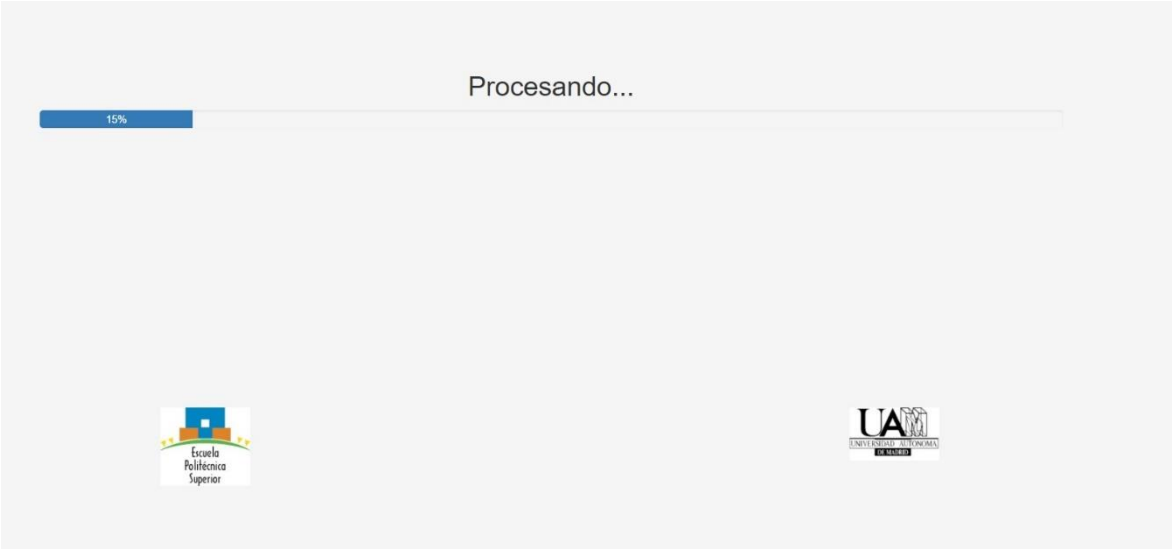


Ilustración 12. Interfaz 4

Una vez finalizado el proceso, la barra se rellena totalmente y aparecerá un botón de “Ver resultados”.

Ilustración 13. Interfaz 5

Tras pulsar ese botón, aparecerá una última pantalla con los resultados, ordenados por sujeto analizado. Se podrán ver su nombre y las horas, minutos y segundos de “sueño”. Aparecerá

además un botón de “Volver al inicio”, que si lo pulsamos nos devolverá a la primera pantalla.

Ilustración 14. Interfaz 6

5.2.2 Almacenamiento de datos

En el apartado de diseño se seleccionó para la herramienta el uso de una base de datos embebida HSQLDB debido a su rapidez. Puesto que la aplicación deberá almacenar una gran cantidad de datos (alrededor de 5,5 millones de registros por sujeto), una base de datos en memoria no sería capaz de gestionar tal volumen sin colapsar el sistema sobre el que se ejecuta, por lo que se tomó la decisión de realizar parte del almacenamiento en disco.

Además, para mantener la rapidez de acceso a los datos, se tomó la decisión de crear las tablas de la base de datos con el parámetro CACHED, lo que implica que solo la información de la transacción se almacena en memoria.

Se puede ver el script de creación de las tablas y la url de conexión a través de JDBC a la base de datos en el *Anexo D: Script de creación de tablas y url de conexión a la base de datos*.

5.2.3 Uso de AWK

AWK es un lenguaje de programación que está diseñado con el fin de poder procesar datos basados en texto, ya sean ficheros o flujos de datos. Se decidió estudiar su implantación en el código, con el fin de acelerar la salida de los datos al fichero. Sin embargo, se decidió no emplearlo varias razones:

- Prácticamente todo el tiempo de ejecución del programa se emplea en la carga inicial de datos en la base de datos y en su procesamiento, no en la lectura/escritura de ficheros.

- Se podría adaptar el programa con sentencias de AWK a la hora de escribir los resultados finales en los ficheros, ya que se copia del de entrada para guardar en el de salida, pero solo las líneas que sean de "no sueño", bajo unas condiciones. Sin embargo, introducir estas sentencias: modificación del código, cambio del algoritmo para sacar las condiciones a la hora de copiar archivos, mayor tiempo de ejecución… hace que se deseche esta opción, ya que complica la ejecución y la mejora en el tiempo de guardado final es casi inapreciable, mientras que el incremento en el tiempo de ejecución es considerable.

5.3 Procesado de los datos

Dado que uno de los objetivos de este TFG es analizar los distintos algoritmos existentes para extraer el tiempo de sueño de un sujeto y encontrar los óptimos, adaptado a nuestros datos de entrada, así como realizar una comparativa de los errores relativos obtenidos.

Para este fin han desarrollado tres métodos basados en los algoritmos ya existentes (Winkler y Berg) y en la Inteligencia Artificial mediante el uso de Weka, una librería Java que contiene una colección de algoritmos de aprendizaje automático para tareas de minería de datos.

5.3.1 Metodo 1: Winkler

Nuestro objetivo principal es dar solución al problema planteado de extraer el tiempo de sueño o "luces fuera" del tiempo de actividad física del sujeto, diferenciándolo del tiempo de sedentarismo. Para ello, se decidió primeramente basarse en un algoritmo diseñado para una colección de datos de 24 horas, obtenido del acelerómetro activPAL. Este algoritmo, "Identifying adults' valid waking wear time by automated estimation in activPAL data collected with a 24 h wear protocol", ha sido diseñado por diversos expertos de Australia.

Las principales características de este algoritmo automático son:

- Los participantes del estudio llevaron el acelerómetro activPAL 24 seguidas durante 10 días.

- Se solicitó que rellenaran un diario en el que apuntaban las horas en las que se acostaban, se dormían, se despertaban, se levantaban de la cama, y también cuándo se quitaban el acelerómetro.

- El algoritmo trata de identificar los tiempos en determinadas posiciones para suprimirlo del tiempo activo. Primero detecta el tiempo de "sueño o no-llevado" y luego dentro de estos periodos diferencia entre el tiempo no-llevado y el tiempo de sueño.

Tras analizar el algoritmo, se procedió a adaptarlo a nuestro estudio, empleando los mismos pasos pero utilizando los 51 tipos de actividad de nuestro proyecto (*Anexo C: Tipos de actividad*).

Serán considerados como “de sueño” aquellas que sean de tipo *lie, recline, sit*, todos los subtipos de estas, más los cambios de posturas entre ellas. Los pasos de este algoritmo serán:

Primer paso

Detectar los periodos en los que las variables lie/recline/sit (y todos los subtipos) sumen un periodo más de 5 horas, o un periodo más largo de 2 horas, siendo este el periodo el más largo dentro de las 24 horas de ese día, contando un período de 24 horas.

Segundo paso

Detectar si las actividades previas y posteriores al tiempo de sueño detectado en el primer paso pertenecen a una actividad o al tiempo de sueño:

- Si las variables que aparecen son lie/recline/sit durante más de 2 horas.
- Si existe un periodo de más de 30 minutos en lie/recline/sit en con menos de 20 pasos.
- Si únicamente son periodos en los que el participante cambia de postura lie/recline/sit (dentro de cada subtipo) pero no se detecta ningún paso.

Tercer paso

Los días no válidos se interpretarán de la misma manera. Aunque la previsión es de que todos tendrán datos de acelerometría válidos, puesto que esto se decide en un análisis previo, si se encuentra un periodo en el que:

- Cualquier actividad supone más del 95% del tiempo llevado.
- Hay menos de 500 pasos.
- El tiempo total de acelerometría es menor a 10 horas.

Cuarto paso

Comparación del algoritmo con el tiempo de sueño autoreportado por los participantes.

5.3.2 Método 2: Winkler + Berg

Tras implementar este primer algoritmo y comprobar que los resultados de error no eran del todo satisfactorios (ver más adelante en *Integración, pruebas y resultados*), se decidió mejorarlo a partir de otro algoritmo. Para ello, el Método 2 se basó en el Método 1, más unas mejoras extraídas de otro algoritmo (Berg), “Identifying walking time in 24-h accelerometry data in adults using an automated algorithm“, desarrollado en la Universidad de Maastricht (Paises Bajos).

Las principales características de este algoritmo automático son:

- Identificación de los tiempos de sueño y despierto.

- Algoritmo con criterios de búsqueda o identificación de patrones (standing/stepping o sitting/lying dentro de un periodo de tiempo determinado. Una vez ha encontrado estos patrones, el algoritmo suma diversos tiempos permanecidos en estas posiciones.

- Creación y establecimiento de unos puntos de corte para la comparación con los tiempos obtenidos en la identificación de los patrones activos y sedentarios y establecimiento de la hora de inicio de sueño y la hora de inicio despierto.

Uno de los aspectos a mejorar que encontramos en el Método 1 fue que los breves períodos de tiempo de "no sueño" entre períodos largos de sueño (como por ejemplo ir al baño en medio de la noche) no eran tenidos en cuenta, lo que hacía que, en muchos casos, partiese períodos de más de cinco horas y que esos períodos resultantes no llegasen a las cinco horas mínimas necesarias, por lo que no se tenían en cuenta en las mediciones, ya que el Método 1 tan sólo se queda con el mayor período de más de 2 horas (aparte del de 5 horas).

El resultado es que se perdía mucho tiempo de sueño, y el tiempo final tan solo era de 2 horas y algo más (porque se sumaba a períodos de antes o después).

A partir de este algoritmo usamos las siguientes mejoras, que se introdujeron en el anteriormente implementado, dando como resultado el "Método 2":

- Establecimiento de la hora de inicio de sueño y la hora de inicio despierto, que según el algoritmo de Berg es entre las 19:00h y las 12:00h del día siguiente.

- Los pequeños períodos de tiempo de "no sueño" durante la noche, entre períodos de sueño, eran contados como "sueño" si su duración no era mayor a los 6 minutos.

5.3.3 Método 3: Inteligencia Artificial (mediante librería Weka)

Dado que los resultados obtenidos no satisfacían totalmente (ver más adelante en *Integración, pruebas y resultados*), se decidió emplear aprendizaje automático, dando como resultado el "Método 3". Para ello se procedió a integrar Weka en nuestra herramienta en Java.

Weka (Waikato Environment for Knowledge Analysis) es un software libre orientado al aprendizaje automático y a la minería de datos. Está escrito en lenguaje Java y ha sido

desarrollado por la Universidad de Waikato, en Nueva Zelanda. El paquete de Weka dispone de una serie de algoritmos para el análisis de datos y el modelado predictivo.

En nuestro proyecto utilizaremos Weka para analizar los ficheros de datos de cada sujeto. No usaremos la versión de interfaz gráfica, sino el paquete de Weka para Java, que nos permitirá integrar sus funcionalidades en el código que estamos desarrollando.

Para comenzar, entrenaremos nuestra red neuronal proporcionada por Weka con una serie de datos de entrenamiento, contenidos en un fichero tipo ".arff" (ver más adelante en *Integración, pruebas y resultados*).

Tras este entrenamiento, procederemos a clasificar los datos de cada usuario. Para ello, usaremos diferentes clasificadores, con el fin de encontrar el que de unos resultados con el menor error posible.

5.3.3.1 Algoritmo J48

El algoritmo J48 de Weka es una implementación del algoritmo C4.5, uno de los algoritmos más empleados de minería de datos, basado en árboles de decisión. El parámetro más importante que debemos de tener en cuenta es el factor de confianza para la poda, que influye en gran medida en el tamaño y en la capacidad de predicción de este árbol construido. El 25% es el valor por defecto.

Este método emplea valores continuos y usa criterios estadísticos para impedir que el árbol se sobre-adapte, es decir, que "crezca demasiado": aprenda los datos en vez de generalizar.

Las fases del este algoritmo serían, de forma simplificada:

1- Parar la construcción del árbol si:

   a. Todos los ejemplos son de la misma clase.
   b. Si no quedan ejemplos o atributos.
   c. Si no se esperan mejoras continuando con el proceso de subdivisión.

2- Si no se cumplen a, b o c: elegir el mejor atributo para poner en ese nodo (el que haga que la entropía media sea la menor).

3- Construir de forma recursiva tantos subárboles como posibles valores tenga el atributo que está seleccionado.

A continuación se muestra un sencillo ejemplo de un árbol de decisión obtenido con J48:

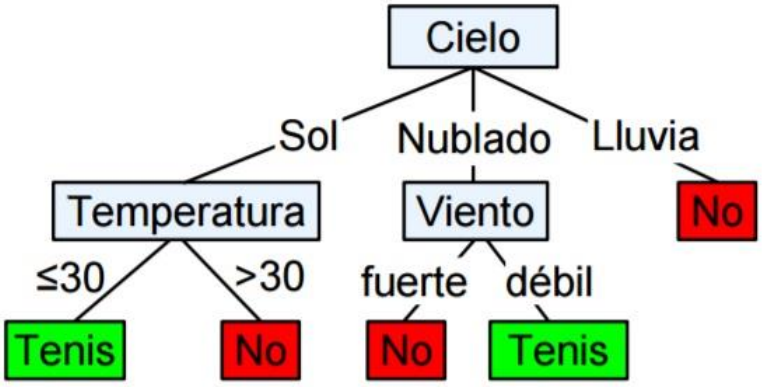


Ilustración 15. Algoritmo J48

5.3.3.2 Algoritmo IBk (k vecinos)

El algoritmo denominado K-Nearest Neighbors (K-vecinos más próximos), KNN, es un método supervisado que se utiliza principalmente para clasificación, aunque también se puede emplear para regresión. La idea principal de este algoritmo es que el objeto nuevo se clasificará en la clase más frecuente de sus K vecinos más cercanos.

En lugar de implementarse una estructura capaz de predecir, mediante un árbol de decisión o reglas, tan solo se guardan las instancias (en este caso los datos) o representantes de los mismos. Para clasificar una nueva instancia, se buscan los objetos más parecidos o cercanos.

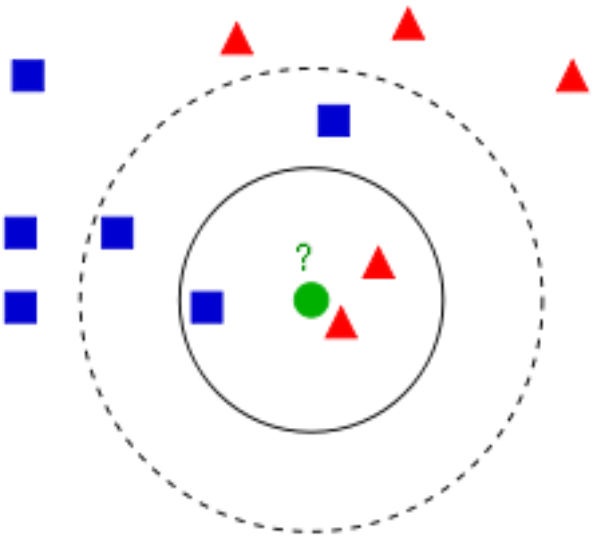


Ilustración 16. Algoritmo IBk

Este método es similar a la forma de clasificar de las personas, ya que para resolver un problema se intenta recordar el caso más parecido que ya se haya resuelto o se sea capaz de resolver.

En el caso de Weka, K va a ser igual a 1, es decir, que el nuevo objeto se clasificará en la clase más frecuente de su vecino más próximo.

5.3.3.3 Algoritmo Naive Bayes

El algoritmo Naive Bayes es uno de los clasificadores más usados, debido a su simplicidad y rapidez. Se basa en el Teorema de Bayes, también conocido como Teorema de la Probabilidad Condicionada [4].

Este algoritmo se divide en dos partes:

1- **Crear el modelo**
   - **a.** Se calculan las probabilidades "a priori" de cada clase.
   - **b.** Para cada una de las clases, se hace un recuento de los valores de atributos que toma cada una.
   - **c.** Aplicamos la Corrección de Laplace, con el fin de evitar los valores 0.
   - **d.** Se normalizan los resultados, para que estén entre 0 y 1.

2- **Clasificar un ejemplo**
   - **a.** Para cada clase, determinados los valores de probabilidad de cada valor de los atributos del ejemplo.
   - **b.** Aplicamos la fórmula de Naive Bayes. Esta fórmula se obtiene de desarrollar la del Teorema de Bayes.

$$P(A|B)=\frac{P(B|A)\cdot P(A)}{P(B)}$$

$$P(A|b_1,b_2,b_3,b_4)=P(A)\cdot(P(b_1|A)\cdot P(b_2|A)\cdot P(b_3|A)\cdot P(b_4|A))$$

$$P(A|b_1,b_2,\dots,b_n)=P(A)\cdot P(b_1,b_2,\dots,b_n|A)=P(A)\cdot\prod_{i=1}^{n}P(a_i|A)$$

$$Solucion=arg\max_{i=1}^{n}P(c_i)\cdot\prod_{j=1}^{m}P(a_j|c_i)$$

5.3.3.4 Algoritmo Random Forest

El algoritmo Random Forest (RF) es uno de los métodos más usados por los científicos de datos. Este algoritmo emplea la técnica llamada *Bagging*, que consiste en implementar diferentes modelos mediante muestras aleatorias con reemplazo, combinando los resultados. Mediante el *Bagging* se combinan diferentes árboles [8].

Este algoritmo se divide en cuatro pasos:

1- Se seleccionan al azar participantes, mediante muestreo con remplazo, para crear una serie de grupos de datos.

2- Posteriormente se crea un árbol de decisión con cada grupo de datos.

3- A la hora de implementar esos árboles, se eligen variables al azar para cada nodo de cada árbol, sin límite de profundidad, es decir, sin podar.

4- Se predicen los nuevos datos mediante el "voto mayoritario", clasificándose como *positivo* si la mayoría de los árboles implementados predicen la salida como *positiva*.

5.3.3.5 Algoritmo SVM (Máquinas de Vectores de Soporte)

SVM o Máquinas de Vectores de Soporte son un grupo de algoritmos de aprendizaje supervisado, que representan a los puntos de muestra en el espacio, separando las clases entre dos espacios lo más grandes posibles, mediante el uso de un hiperplano de separación.

Este hiperplano, llamado "vector soporte", se define como el vector entre los dos puntos más cercanos de las dos clases. Cuando los nuevos datos de entrada se ponen en correspondencia con este modelo, dependiendo de los espacios a los que pertenezcan, se les clasifica en una clase o en otra.

5.4 Conclusiones

En este capítulo se ha descrito el proceso de implementación de nuestro sistema informático y se han descrito los métodos y algoritmos a emplear para el procesado de los datos.

Como hemos visto, partiendo del Método 1 basado en Winkler, llegamos al desarrollo del Método 2 como mejora del anterior, eliminándose uno de los problemas principales que impedían encontrar períodos de más de 5 horas.

Con un cambio aparentemente sencillo, hemos producido una mejora considerable, que como veremos en el capítulo siguiente reducirá el error relativo a la cuarta parte.

En cuanto al Método 3, es fundamental la buena elección del algoritmo de clasificación entre los detallados, con el fin de conseguir los resultados más cercanos al error 0, para alcanzar los objetivos descritos al inicio del proyecto.

6 Integración, pruebas y resultados

6.1 Introducción

En este apartado se van a presentar las pruebas llevadas a cabo a la herramienta, con las cuales podamos asegurar el correcto cumplimiento de los requisitos definidos. Primeramente describiremos el entorno donde se han realizado las pruebas. Posteriormente se mostraran estas pruebas y finalmente los resultados obtenidos, destacándose los puntos más importantes de estas pruebas.

6.2 Descripción de los datos de entrada

Los datos de entrada para los tres métodos serán los obtenidos del acelerómetro, pre-procesados con el programa ActView de IDEEA y dando una serie de ficheros de texto, uno por sujeto. Esto ya se explicó en el apartado *2.1.4 Datos del estudio*.

En total se emplearán para la validación un total de 30 ficheros, que se corresponden a 30 períodos de 24 horas de 15 participantes. Cada sujeto, por tanto, contará con 2 días de estudio, cada uno de los cuales contará con 2,8 millones de líneas.

Código del Sujeto	Día 1 Despertar	Día 1 Acostar	Día 2 Despertar	Día 2 Acostar
1233	7:45	23:45	7:30	0:00
1234	7:15	0:00	7:30	0:15
1235	8:00	0:10	7:00	0:50
1236	9:00	22:00	9:00	22:00
1237	7:15	0:00	8:15	0:00
1239	7:00	23:45	9:00	23:50
1241	7:30	23:20	7:00	23:20
1242	9:00	23:30	8:00	23:50
1243	10:30	1:15	7:00	1:00
1245	8:15	0:00	7:30	0:00
1246	8:10	1:00	9:30	0:45
1250	8:30	0:00	7:00	1:00
1251	6:30	22:30	8:45	22:45
1252	8:00	23:00	8:00	23:00

Tabla 1. Participantes de prueba

6.2.1 Datos de entrenamiento para el Método 3 (Inteligencia Artificial con Weka)

En el caso del método 3, para el que se ha implementado un algoritmo con Inteligencia Artificial, se precisarán unos datos de entrenamiento para preparar el sistema proporcionado por Weka. Estos datos, contenidos en un archivo tipo .arff, serán un total de 125 tuplas de 5 minutos (9.600 tikcs), compuestos por 51 números (cada uno indica la presencia relativa de un tipo de actividad, de la 99 a la 183 – *Anexo C: Tipos de actividad*) y el último que indica si ese conjunto de datos es sueño o no.

A continuación podemos ver un ejemplo del fichero *training.arff*, compuesto de los 51 atributos de tipo real con cada tipo de actividad, y otro atributo de tipo {yes, no}. Contiene posteriormente las tuplas que hablábamos anteriormente, con la presencia relativa entre 0 y 1.

```
@relation sedentario

@attribute      A100    real
@attribute      A101    real
@attribute      A183    real
.
.
.
@attribute      A99             real
@attribute      sedentario      {yes,no}


@data

100,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,yes
0,100,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,yes
.
.
.
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.08333333333333334,0.0,0.0,0.0,0.0,0.33333333333333337,0.0,0.0
,0.0,0.0,0.0,0.0,85.796,0.04166666666666667,0.0,0.0,0.0,0.0,0.0,0.08333333333333334,0.5,0.083333333
33333334,0.0,1.292,1.125,0.7916666666666667,0.0,0.0,0.0,3.458,3.17,2.292,0.9583333333333333,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,no
```

6.3 Entorno de pruebas

Para llevar a cabo las pruebas, se han realizado en un ordenador portátil con las siguientes especificaciones:

Sistema operativo: Windows 10 Home de 64 bits
Procesador: Intel Core i7 – 6700 HQ @ 2,60 Hz
Memoria instalada: 16,0 GB

6.4 Pruebas realizadas

Para el desarrollo hemos empleado los 30 ficheros, que se corresponden a 30 períodos de 24 horas de 15 participantes diferentes. Posteriormente, se han comparado con los datos de sueño proporcionados por los participantes. Los resultados son:

Método 1:

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	7:30:25	3:59:48	16,65%
1234	14:30:00	0:00:00	7:15:00	30,21%
1235	14:00:00	4:59:35	4:30:13	18,76%
1236	22:00:00	17:15:33	2:22:14	9,88%
1237	15:30:00	13:32:06	0:58:57	4,09%
1239	16:25:00	7:08:29	4:38:16	19,32%
1241	15:50:00	4:18:05	5:45:58	24,02%
1242	17:40:00	4:03:04	6:48:28	28,37%
1243	15:15:00	4:29:08	5:22:56	22,43%
1245	15:45:00	4:50:07	5:27:27	22,74%
1246	15:55:00	4:18:21	5:48:20	24,19%
1250	14:30:00	5:46:25	4:21:47	18,18%
1251	18:00:00	9:20:06	4:19:57	18,05%
1252	18:00:00	9:51:12	4:04:24	16,97%
			Error total:	**19,56%**

Tabla 2. Método 1

Método 2

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	15:44:43	0:07:21	1,02%
1234	14:30:00	4:06:39	5:11:40	21,64%
1235	14:00:00	10:56:14	1:31:53	6,38%
1236	22:00:00	20:26:52	0:46:34	3,23%
1237	15:30:00	18:05:21	1:17:40	5,39%
1239	16:25:00	10:33:50	2:55:35	12,19%
1241	15:50:00	15:42:27	0:03:46	0,26%
1242	17:40:00	16:36:33	0:31:44	2,20%
1243	15:15:00	14:05:35	0:34:42	2,41%
1245	15:45:00	15:21:14	0:11:53	0,83%
1246	15:55:00	16:12:40	0:08:50	0,61%
1250	14:30:00	10:52:55	3:37:05	7,54%
1251	18:00:00	20:48:56	2:48:56	5,87%
1252	18:00:00	9:51:12	8:08:48	16,97%
			Error total:	

	5,11%

Tabla 3. Método 2

Método 3 (J48):

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	17:40:00	1:05:00	4,51%
1234	14:30:00	11:20:00	1:35:00	6,60%
1235	14:00:00	17:50:00	1:55:00	7,99%
1236	22:00:00	21:20:00	0:20:00	1,39%
1237	15:30:00	19:00:00	1:45:00	7,29%
1239	16:25:00	18:55:00	1:15:00	5,21%
1241	15:50:00	17:15:00	0:42:30	2,95%
1242	17:40:00	19:00:00	0:40:00	2,78%
1243	15:15:00	16:00:00	0:22:30	1,56%
1245	15:45:00	20:00:00	2:07:30	8,85%
1246	15:55:00	18:30:00	1:17:30	5,38%
1250	14:30:00	18:05:00	1:47:30	7,47%
1251	18:00:00	18:30:00	0:15:00	1,04%
1252	18:00:00	16:45:00	0:37:30	2,60%
			Error total:	**4,69%**

Tabla 4. Método 3 (J48)

Método 3 (Naive Bayes):

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	19:05:00	1:47:30	7,47%
1234	14:30:00	9:45:00	2:22:30	9,90%
1235	14:00:00	16:40:00	1:20:00	5,56%
1236	22:00:00	21:20:00	0:20:00	1,39%
1237	15:30:00	19:00:00	1:45:00	7,29%
1239	16:25:00	18:40:00	1:07:30	4,69%
1241	15:50:00	18:35:00	1:22:30	5,73%
1242	17:40:00	20:20:00	1:20:00	5,56%
1243	15:15:00	14:05:00	0:35:00	2,43%
1245	15:45:00	15:50:00	0:02:30	0,17%
1246	15:55:00	19:55:00	2:00:00	8,33%
1250	14:30:00	18:40:00	2:05:00	8,68%
1251	18:00:00	19:40:00	0:50:00	3,47%
1252	18:00:00	18:10:00	0:05:00	0,35%
			Error total:	**5,07%**

Tabla 5. Método 3 (Naive Bayes)

Método 3 (IBk):

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	17:50:00	1:10:00	4,86%
1234	14:30:00	12:40:00	0:55:00	3,82%
1235	14:00:00	16:20:00	1:10:00	4,86%
1236	22:00:00	19:40:00	1:10:00	4,86%
1237	15:30:00	19:15:00	1:52:30	7,81%
1239	16:25:00	18:40:00	1:07:30	4,69%
1241	15:50:00	15:50:00	0:00:00	0,00%
1242	17:40:00	19:15:00	0:47:30	3,30%
1243	15:15:00	13:10:00	1:02:30	4,34%
1245	15:45:00	15:55:00	0:05:00	0,35%
1246	15:55:00	18:00:00	1:02:30	4,34%
1250	14:30:00	18:00:00	1:45:00	7,29%
1251	18:00:00	18:35:00	0:17:30	1,22%
1252	18:00:00	16:20:00	0:50:00	3,47%
			Error total:	**3,94%**

Tabla 6. Método 3 (IBk)

Método 3 (Random Forest):

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	18:30:00	1:30:00	6,25%
1234	14:30:00	11:05:00	1:42:30	7,12%
1235	14:00:00	16:25:00	1:12:30	5,03%
1236	22:00:00	21:20:00	0:20:00	1,39%
1237	15:30:00	19:50:00	2:10:00	9,03%
1239	16:25:00	18:30:00	1:02:30	4,34%
1241	15:50:00	18:05:00	1:07:30	4,69%
1242	17:40:00	20:05:00	1:12:30	5,03%
1243	15:15:00	13:45:00	0:45:00	3,12%
1245	15:45:00	17:35:00	0:55:00	3,82%
1246	15:55:00	19:40:00	1:52:30	7,81%
1250	14:30:00	18:00:00	1:45:00	7,29%
1251	18:00:00	19:15:00	0:37:30	2,60%
1252	18:00:00	17:45:00	0:07:30	0,52%
			Error total:	**4,86%**

Tabla 7. Método 3 (Random Forest)

Método 3 (SVM):

Sujeto	Tiempo de sueño comunicado (48 horas)	Tiempo de sueño estimado (48 horas)	Error absoluto (24 horas)	Error relativo (24 horas)
1233	15:30:00	13:30:00	1:00:00	4,17%
1234	14:30:00	5:55:00	4:17:30	17,88%
1235	14:00:00	11:50:00	1:05:00	4,51%
1236	22:00:00	15:20:00	3:20:00	13,89%
1237	15:30:00	14:35:00	0:27:30	1,91%
1239	16:25:00	14:55:00	0:45:00	3,13%
1241	15:50:00	10:40:00	2:35:00	10,76%
1242	17:40:00	12:05:00	2:47:30	11,63%
1243	15:15:00	10:00:00	2:37:30	10,94%
1245	15:45:00	9:00:00	3:22:30	14,06%
1246	15:55:00	13:25:00	1:15:00	5,21%
1250	14:30:00	10:55:00	1:47:30	7,47%
1251	18:00:00	10:45:00	3:37:30	15,10%
1252	18:00:00	13:05:00	2:27:30	10,24%
			Error total:	**9,35%**

Tabla 8. Método 3 (SVM)

A continuación podemos ver la gráfica de resultados comparados del error relativo de cada método:

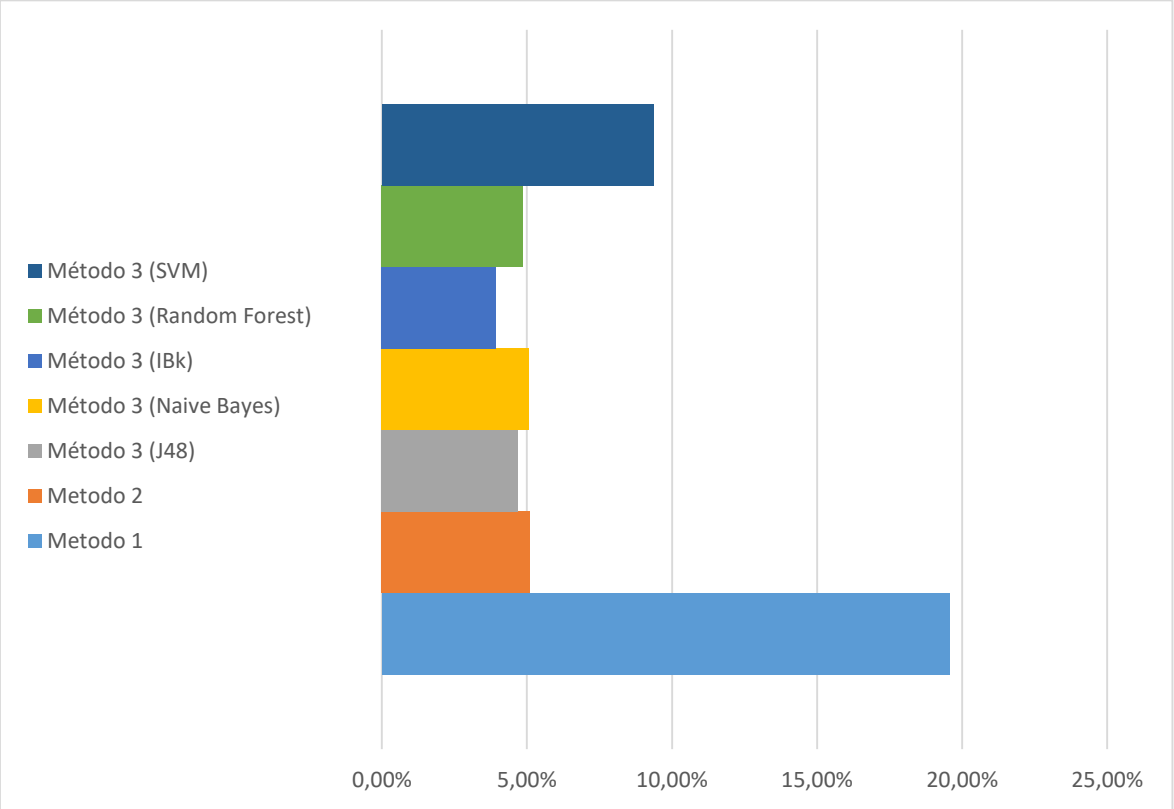


Ilustración 17. Error por método

A continuación podemos ver la gráfica de resultados comparados del error relativo de cada algoritmo empleado en el Método 3:

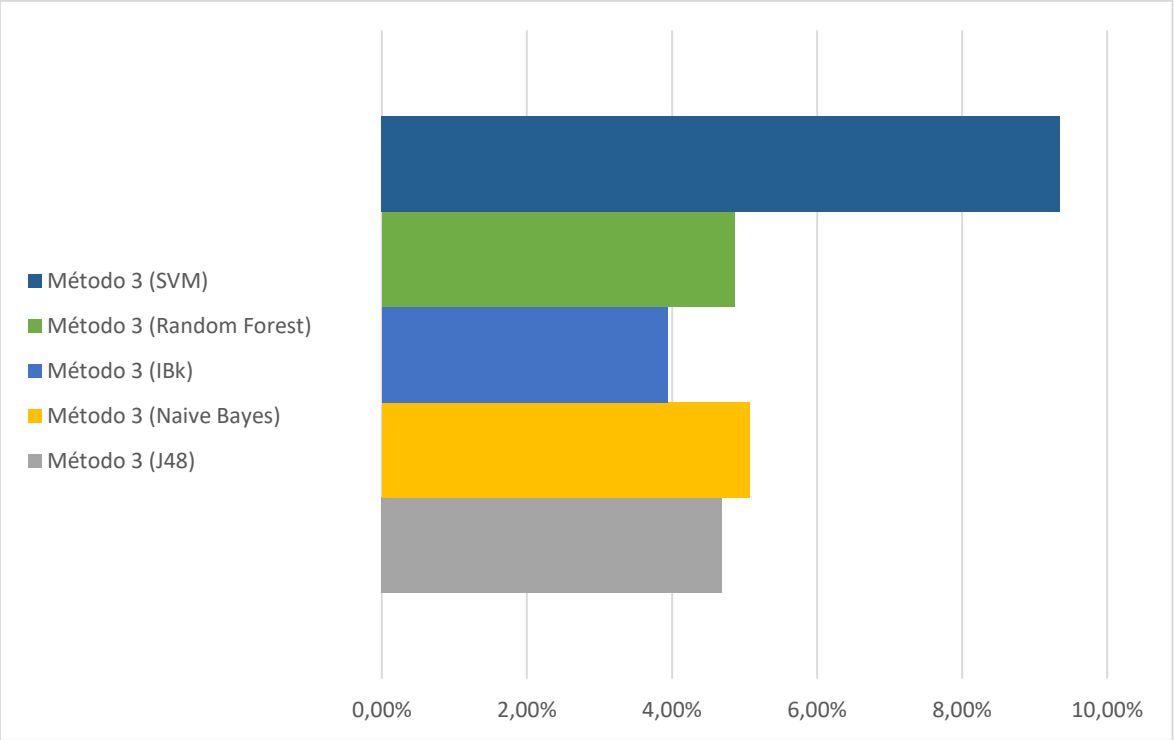


Ilustración 18. Error por algoritmo (Método 3)

Finalmente se muestran los resultados de error comparados entre métodos, por sujeto estudiado:

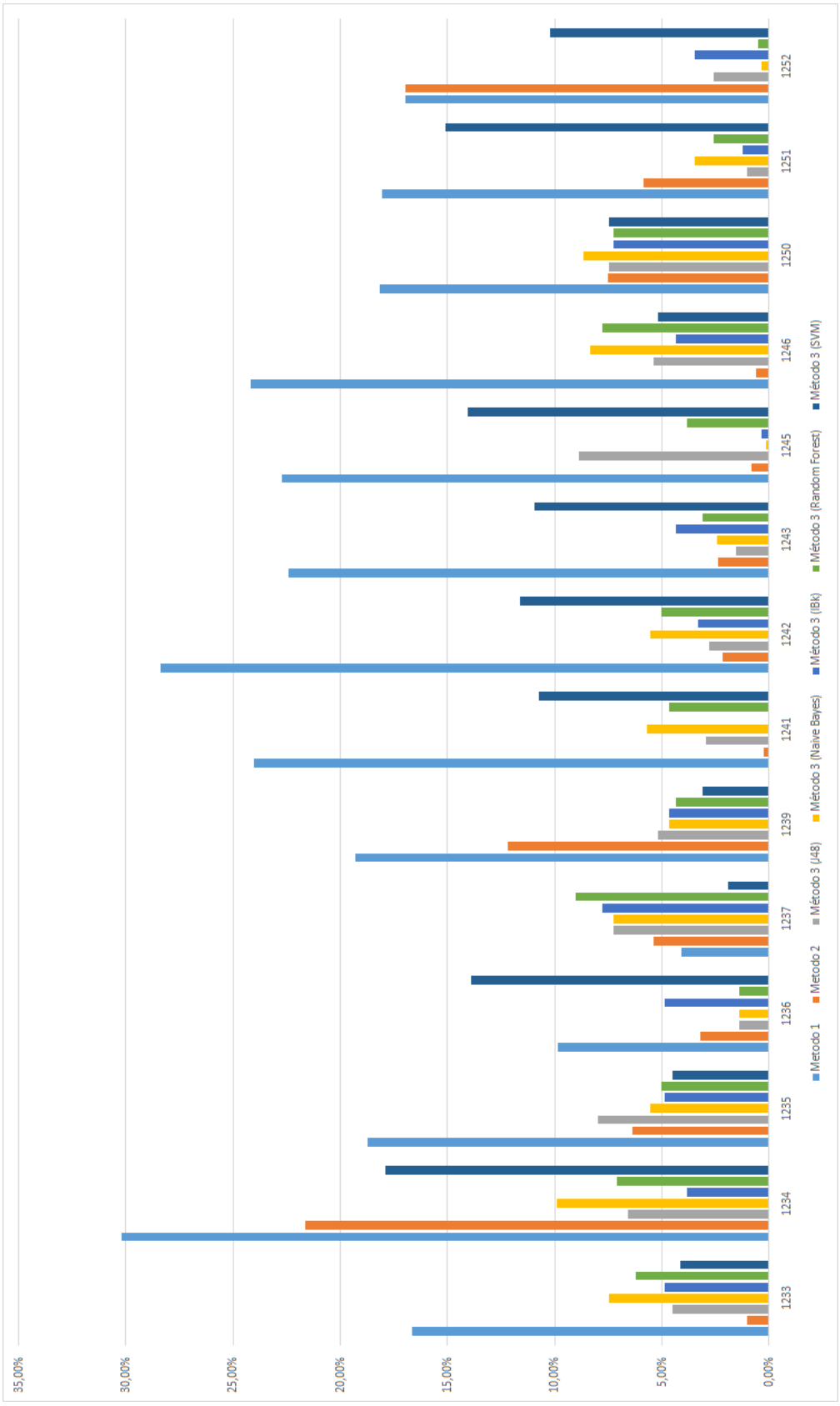


Ilustración 19. Error entre métodos por sujeto estudiado

6.5 Conclusiones

Como se puede ver en los apartados anteriores, el error relativo del Método 1 es de **19'56%**, que lo hace muy elevado para poder ser válido para el objetivo que se buscaba. El elevado error proviene, en gran medida, de que no se contabilizan en algunos casos grupos de 5 horas, y el algoritmo se queda tan solo con el mayor grupo de 2 horas (más sus alrededores).

Esto queda patente en los casos, por ejemplo, de los participantes 1241, 1242, 1243, 1245 o 1246, en los que el sueño estimado supera en poco las 4 horas (unas 2 horas estimadas de sueño cada 24 horas).

En cuanto al Método 2, se puede comprobar que las mejoras añadidas al Método 1 hacen que el error baje hasta **5'11%**, dentro del umbral aceptable. Como se explicó con anterioridad, al interpretar los períodos breves de "no sueño" como "sueño", se encuentran los períodos de más de 5 horas que antes no se encontraban (al estar partidos en grupos de más de 2 horas y menos de 5 horas).

Esto queda patente en el caso del sujeto 1241, que apenas produce 4 minutos de error, mientras que anteriormente solo alcanzaba las 2 horas de sueño en 24 horas.

En cuanto al Método 3, se debe destacar que con todos los clasificadores (a excepción del SVM) se ha bajado el error relativo. El que más desciende, y por tanto es el óptimo que buscábamos, es el que emplea IBk o k –vecinos, que rebaja el error al **3'94%**. Por su lado, SVM tiene un error relativo total de 9'35%, lo que no hace aconsejable su utilización.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El primer objetivo marcado para este Trabajo de Fin de Grado, consistente en la implementación de un sistema informático que sea capaz de devolver el tiempo de sueño o "luces fuera" de los participantes estudiados, ha sido cumplido satisfactoriamente.

La herramienta desarrollada permite extraer con éxito el tiempo de sueño del sujeto con un error menor del 4%, lo que hace que los resultados sean próximos a lo indicado por los participantes del estudio. Además, los datos de "no sueño" de cada participante del estudio se guardan en un fichero de texto, uno por cada persona. Este resultado permite que se pueda dar como cumplido el segundo objetivo que se marcó inicialmente.

El mayor reto con el que me he encontrado en este proyecto era el de conseguir almacenar una gran cantidad de datos para procesar sin colapsar el sistema sobre el que se ejecuta. Para ello, se encontró la necesidad de realizar parte del almacenamiento en disco. Además, para mantener la rapidez de acceso a los datos, se tomó la decisión de crear las tablas de la base de datos "cacheadas".

Llevar a cabo este TFG me ha hecho adquirir conocimientos nuevos así como ampliar otros aprendidos durante la carrera (en especial en las asignaturas de Análisis de Algoritmos, Sistemas Informáticos, Inteligencia Artificial e Ingeniería del Software). Uno de ellos fue la implementación desde cero de un sistema informático con Java, que aunque es mejorable, es satisfactorio.

La elección de las tecnologías a emplear han sido, en mi opinión, las correctas. Se trata de aspectos novedosos, como Spring Boot o el framework JPA, que a través de este TFG he podido conocer.

Se debe destacar que la curva de aprendizaje durante este proceso ha sido mayor de la esperada, adquiriendo nuevos conocimientos y mejorando mi programación en lenguajes de programación como Java.

7.2 Trabajo futuro

Aunque el resultado final de la herramienta implementada ha sido satisfactorio, existen aspectos a mejorar.

Por un lado, en el futuro se podría explorar la posibilidad de implementar el proceso en paralelo de varios participantes mediante el uso de hilos. De esta manera, se aceleraría la ejecución, aunque habría que tener en cuenta el aumento del uso de recursos.

Por otro lado, se podría añadir funcionalidades como la posibilidad de exportar los datos finales a un fichero de Excel, PDF, etc.

El uso de ventanas emergentes para la aplicación informática desarrollada podría ser otro de los objetivos a marcarse para el futuro, convirtiendo a la aplicación en una herramienta totalmente independiente.

También se podría habilitar en la ventana de selección de método la elección de los diferentes clasificadores del Método 3, al igual que se hizo entre el Método 1, el Método 2 y el Método 3. Además, se podría indicar por pantalla el error obtenido para ese sujeto y con ese método.

Referencias

[1] JPA Tutorial: Setting Up JPA in a Java SE Environment https://dzone.com/articles/jpa-tutorial-setting-jpa-java

[2] Spring ORM example – JPA, Hibernate, Transaction http://www.journaldev.com/7655/spring-orm-example-jpa-hibernate-transaction

[3] Java standalone app with JPA + Hibernate (or similar) and Apache Derby embedded DB http://stackoverflow.com/questions/12688162/java-standalone-app-with-jpa-hibernate-or-similar-and-apache-derby-embedded

[4] Clasificador Naïve Bayes http://naivebayes.blogspot.com.es/

[5] ¿Qué es Spring Boot? http://www.arquitecturajava.com/que-es-spring-boot/

[6] Desarrollando una aplicacion Spring Framework MVC paso a paso http://www.davidmarco.es/spring-mvc

[7] Data Mining con R http://apuntes-r.blogspot.com.es/2014/12/bagging-para-mejorar-un-modelo.html

[8] Algoritmo Random Forest http://www.bigdatahispano.org/noticias/algoritmo-random-forest/

[9] Normalización del uso de los conceptos “sedentario” y “sedentarismo (comportamiento sedentario)” http://www.sedentarybehaviour.org/wp-content/uploads/2011/06/Spanish-Final-Version.pdf

[10] Kwon S, Jamal M, Zamba GK, Stumbo P, Samuel I. Validation of a novel physical activity assessment device in morbidly obese females. J Obes. 2010

Anexos

A Manual de instalación

Se necesita tener previamente instalado Java SE Development Kit 8 (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)

El entregable que se dará al usuario será un archivo comprimido "AnalisisSueno.zip" que contendrá una carpeta con los siguientes ficheros:

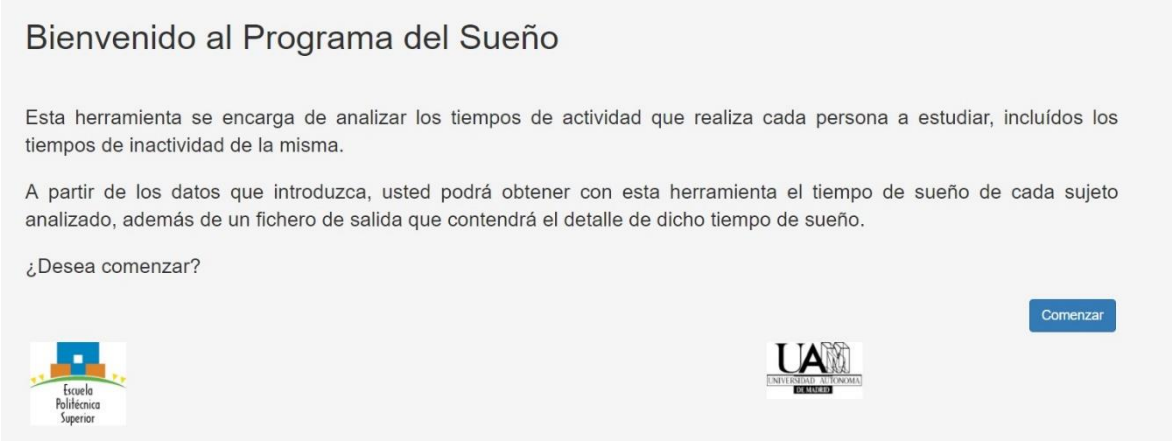
- Archivo .war ejecutable, que contiene el programa.
- Archivo start, que servirá para iniciar el programa, y que abrirá una ventana en el navegador web del usuario a través de la cual este podrá interactuar con la aplicación.
- Archivo stop, que servirá para finalizar el programa en caso de haber algún problema.

Para ejecutar el programa será necesario descomprimir el archivo "AnalisisSueno.zip" y hacer doble click en el archivo *start*, que abrirá una ventana en el navegador del usuario, en la dirección http://localhost:8080/.

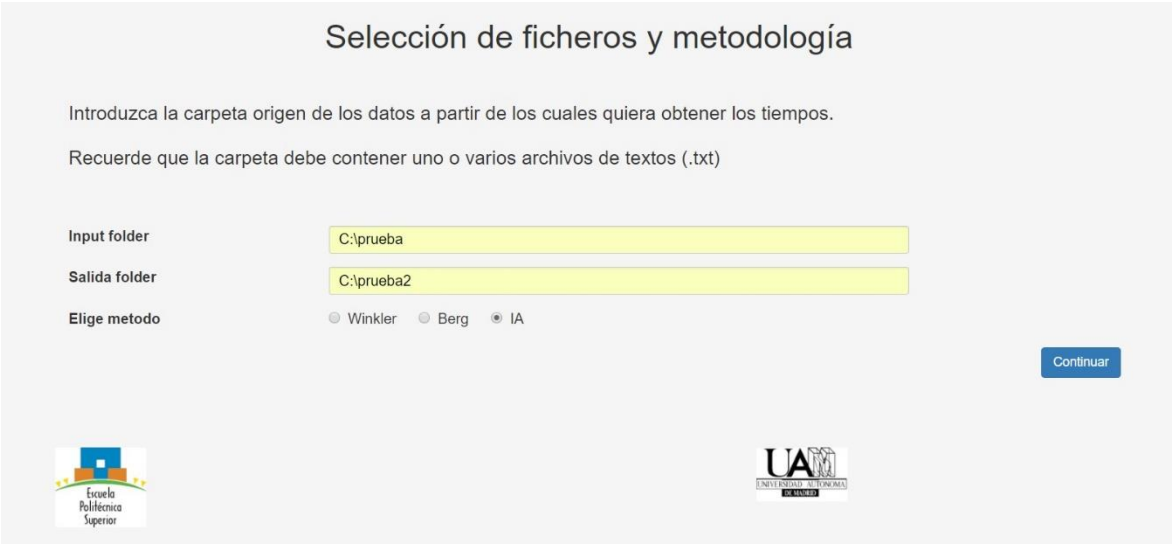
B Manual de usuario

Para ejecutar el programa será necesario descomprimir el archivo "AnalisisSueno.zip" y hacer doble click en el archivo *start*, que abrirá una ventana en el navegador del usuario, en la dirección http://localhost:8080/.

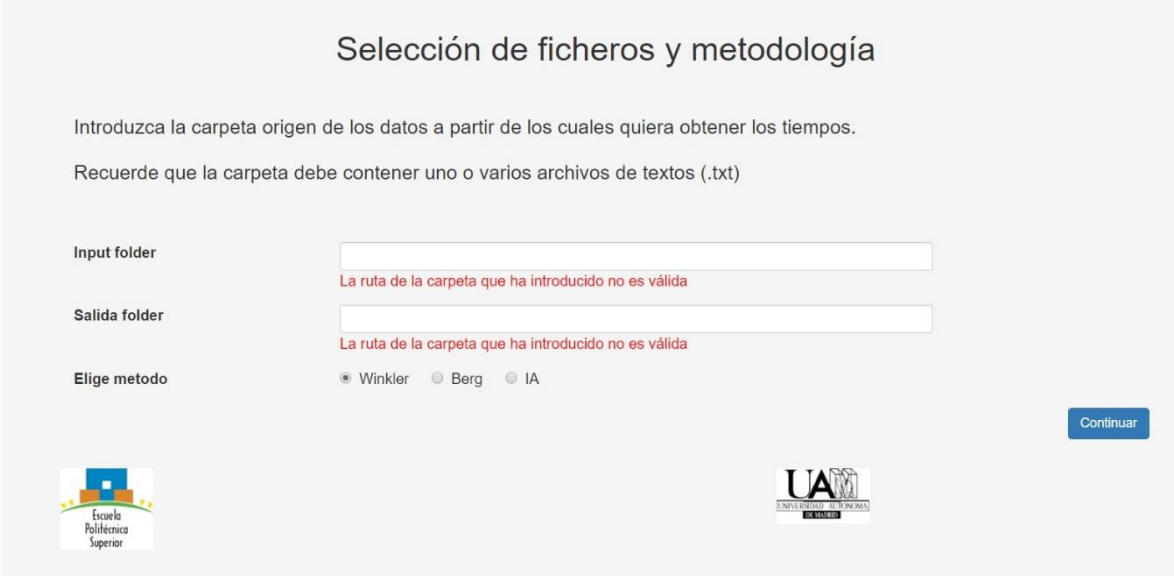
Una vez iniciada la aplicación, a través del navegador se visualizará la pantalla de bienvenida al programa que cuenta con una explicación del objetivo de la herramienta y el botón "Comenzar". Pincharemos en este botón para acceder a la siguiente pantalla.



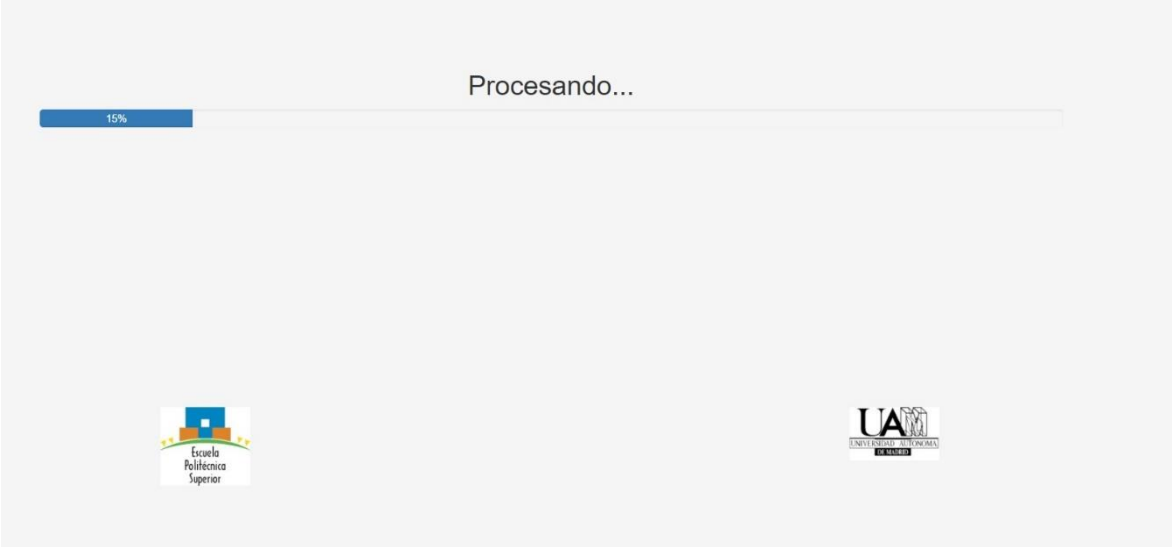
Posteriormente se accede a una ventana nueva donde se solicita al usuario que introduzca la carpeta de origen de los datos, la carpeta de destino de los datos y que seleccione el método a emplear para procesar los datos: *Winkler*, *Berg* o *IA*. En esta última selección, y para evitar posibles errores, se selecciona por defecto el método *Winkler*. Tras presionar el botón "Continuar", pasamos a la siguiente ventana.



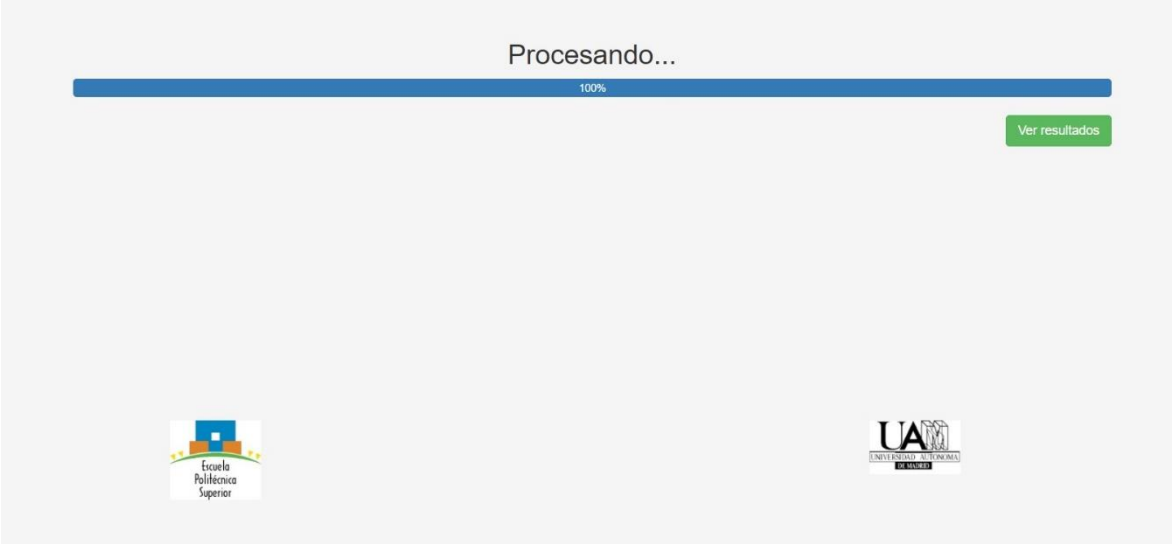
Si la ruta de la carpeta introducida no es válida o está vacía, aparece un mensaje de error.



En esta nueva ventana se muestra una barra de progreso que indicará al usuario el progreso del procesado de los datos mediante el color azul y un indicador del porcentaje.



Una vez finalizado el proceso, aparecerá un botón de “Ver resultados”.



Tras pulsar ese botón, aparecerá una última pantalla con los resultados, ordenados por sujeto analizado. Se podrán ver su nombre y las horas, minutos y segundos de "sueño". Aparecerá además un botón de "Volver al inicio", que si lo pulsamos nos devolverá a la primera pantalla.

Los ficheros que contienen los períodos detallados de tiempo de "no sueño" de cada sujeto estarán almacenados en la carpeta de salida indicada por el usuario en la segunda pantalla de la aplicación.

C Tipos de actividad

Los tipos de actividad, a los que se asigna un código, son:

100 lie down, facing up
101 recline, heel touch ground
102 recline, left leg on right
103 recline, right leg on left
104 changes within lie-recline
105 sit to lie
106 sit to recline
107 stand to lie
108 stand to recline
109 stand to sit
110 lie to sit
111 recline to sit
112 lie to stand
113 recline to stand
114 sit to stand
115 lie down, facing right
116 lie down, facing down
117 lie down, facing left
119 sit with feet up (e.g, on floor)
120 sit
121 sit, left on right
122 sit, right leg on left
123 Sitting with left leg move
124 Sitting with right leg move
125 Sitting, both leg move
126 sit with elbows on legs
127 sit with left toe
128 sit with right toe
129 sit with both toe
130 changes within sit
140 stand
141 standing with weight supported (e.g. forwar on a bar)
142 Standing with left leg move
143 Standing with right leg move
144 Standing carrying weight
145 standing with left leg up (on a stair, e.g.)
146 standing with right leg up
147 standing with left leg, leaning
148 standing with right leg, leaning
149 changes within stand
160 walk, left leg gait
161 walk, right leg gait
162 step up, left leg gait
163 step up, right leg gait
164 step down, left leg gait

165 step down, right leg gait
180 run, left leg gait
181 run, right leg gait
182 jump up with both feet
183 jump down (landing) with both feet
99 unknow posture

D Script de creación de tablas y url de conexión a la base de datos

Script de creación de las tablas y la url de conexión a través de JDBC a la base de datos:

```
create CACHED table IF NOT EXISTS SUJETO (
   id INT AUTO_INCREMENT,
   NAME VARCHAR(250) NOT NULL,
   FICHERO  VARCHAR(250) NOT NULL,
   TIEMPOFINAL VARCHAR(250) NOT NULL,
   PRIMARY KEY (id)
);

create CACHED table IF NOT EXISTS ACTIVIDAD (
   id INT AUTO_INCREMENT,
   TIC VARCHAR(30) NOT NULL,
   TIPOACTIVIDAD VARCHAR(30) NOT NULL,
   SUJETOID VARCHAR(30) NOT NULL,
   ISSEDENTARY VARCHAR(30) NOT NULL,
   PRIMARY KEY (id)
);

create CACHED table IF NOT EXISTS TIPOACTIVIDAD (
   id INT AUTO_INCREMENT,
   CODE INTEGER NOT NULL,
   NAME VARCHAR(250) NOT NULL,
   ISSED  BOOLEAN NOT NULL,

   PRIMARY KEY (id)
);

create CACHED table IF NOT EXISTS FOLDER (
   id INT AUTO_INCREMENT,
   FOLDERIN VARCHAR(250) NOT NULL,
   FOLDEROUT VARCHAR(250) NOT NULL,
   METODO VARCHAR(250) NOT NULL,
   PRIMARY KEY (id)
);


create CACHED table IF NOT EXISTS SEDENTARIO (
   id INT AUTO_INCREMENT,
   INICIO VARCHAR(250) NOT NULL,
   FIN VARCHAR(250) NOT NULL,
   SUMA INTEGER NOT NULL,
   IDUSER VARCHAR(250) NOT NULL,
   HORAS INTEGER NOT NULL,
   PRIMARY KEY (id)
);
```

spring.datasource.url=jdbc:hsqldb:file:./db/AnalisisSuenoDB;