Trabajo final de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas
Universidad de Barcelona

Sistema client-servidor d'ajuda a la focalització basat en el mètode Pomodoro

---

Alain Gonzalez Montesinos

Director:	Dr. J.Daniel Prades
Realizado en:	Departamento de Electrónica
Barcelona,	27 de juny de 2015

Abstract

Summary

The purpose of this document is to analise interruptions as one of the most usual reasons of a decrease in the productivity in open offices and to show an alternative methodology of work with an introduction software. This alternative methodology is intended to help any worker to focus into a task during a configurable time frame with the help of a software, which will allow the user to organise his time on his own way and also will help to get better every day with a score system.

On top of that, workers will see how their productivity increases over time and they will learn the importance of working without interruptions while allowing the rest of the team members to work properly.

Resumen

El siguiente trabajo de fin de grado tiene como objetivo analizar las interrupciones como una de las principales causas en el descenso de la productividad de una persona en las oficinas abiertas, y aportar una metodología de trabajo alternativa junto a un software introductorio. Con ésta metodología alternativa se pretende ayudar a cualquier persona a focalizar su atención en una tarea durante un periodo configurable con ayuda de un software que le permitirá configurar el tiempo a su gusto y poder superarse día a día con un sistema de puntuaciones. Además, verá como poco a poco va en aumento su productividad y comprenderá la importancia que tiene el trabajar sin interrupciones y dejar trabajar a los demás miembros de su equipo de trabajo.

Agradecimientos

Quiero agradecer a mi profesor J.Daniel Prades toda la ayuda y consejos que me ha dado durante estos meses de proyecto. Sin él no habríamos llegado hasta donde hemos llegado.

También a mis amigos y familiares que me han apoyado durante todos estos años de carrera, tanto en los buenos como en los malos momentos.

Gracias a todos.

Índex

1 Introducción **1**

1.1 Oficinas Abiertas 1

1.2 Metodología Pomodoro 4

1.2.1 Objetivos 5

1.2.2 Relación con Metodologías Ágiles y la Ley de Parkinson 5

1.2.3 Desventajas 6

1.3 Motivación del problema 7

2 Proyecto **8**

2.1 Descripción y Objetivos 8

2.2 Variaciones de la metodología Pomodoro 9

2.3 Software no intrusivo. Colores y estados. 10

2.4 Pomodoros correctos e incorrectos 11

2.5 Funcionamiento USB Button 12

2.6 Grupos 13

2.7 Metodología 13

2.7.1 ¿Por qué SCRUM? 13

2.7.2 Implementando Scrum en nuestro TFG 14

2.7.3 Planificación Inicial 15

2.7.4 Planificación Real 17

3 Implementación **20**

3.1 Tecnologías utilizadas 20

3.1.1 Java / RMI 20

3.1.2 USB Button y JNA 20

3.1.3 JSON 22

3.1.4 JNativeHook 23

3.1.5 JavaFX 24

3.2 Funcionalidad 25

3.2.1 Bandeja de Windows 25

3.2.2 Archivo de Configuración 27

3.2.3 Pantalla de Stats 27

3.2.4 Pomodoros 28

3.2.5 Administrador 29

3.3 Casos de Uso 30

3.3.1 Usuario 30
3.3.2 Administrador 31
3.4 Arquitectura 32
3.4.1 Core 34
3.4.2 CorePomodoro 34
3.4.3 CommonHandlers 35
3.4.4 Controllers 36
3.4.5 Timers 37
3.5 Pruebas Realizadas 37

4 Manual de Usuario **39**
4.1 Requisitos e Instalación 39
4.2 Interfaz 40
4.2.1 Iniciando la Interfaz 40
4.2.2 Pantalla Principal y Menú 41
4.2.3 Settings 43
4.2.4 Stats 44
4.2.5 Cerrar la aplicación 46
4.2.6 Administrador 46

5 Futuras Mejoras **49**

6 Conclusiones **51**

1 Introducción

1.1 Oficinas Abiertas

Las empresas se encuentran en un periodo de cambio en su entorno laboral. Las clásicas oficinas con despachos y cubículos con paredes para el resto de empleados están siendo sustituidas por oficinas abiertas. Los empleados pasan de tener su propia mesa con su ordenador a compartir espacio con más gente. Las empresas pasan a tener más espacios comunes que privados.

La idea original data de los años 50 en Hamburgo, pero hasta la década de los 80 y 90 no empezaron a verse unos entornos laborales más abiertos, ya que eran pocas las empresas que se atrevían a cambiar su modelo de trabajo por uno que apenas estaba empezando y del que todavía no había mucha información. No ha sido hasta estos últimos años cuando se ha visto un incremento radical de empresas que han apostado por este cambio. Según los últimos datos que se han podido recoger de la *International Facility Management Association*, un 68% de empresas en Estados Unidos ya trabajan en un entorno abierto, mientras que aquí en España la situación es muy distinta. Importamos la idea desde Estados Unidos, pero apenas un 20% de empresas han adoptado esta forma de trabajo en sus oficinas, aunque cada vez este porcentaje va en aumento y se espera que durante los próximos 5-10 años se incremente considerablemente. Cada vez son más las empresas que están reevaluando sus instalaciones y aprovechando mudanzas para adaptarse al cambio.

Cambiar la oficina tradicional por una Oficina Abierta no es un cambio pequeño que sea sencillo de asumir. Estamos hablando de que una empresa debe replantear su organigrama jerárquico, pasar de ser una empresa vertical a una empresa horizontal, donde jefes y empleados comparten espacio y trabajan codo con codo. De hecho, este es el principal problema que se encuentran las empresas cuando deciden implementar esta nueva forma de trabajar. Las personas con despacho asignado son reticentes a perderlo y pasar a un espacio común. Hasta ahora el despacho otorgaba un estatus y era una clara diferencia jerárquica para una empresa. Este concepto cambia y desaparece.

No hay duda de que grandes empresas como Google, Facebook, Youtube o Microsoft han influido en esta nueva moda. Empresas punteras que optaron por este método de trabajo y les ha llevado a liderar sus mercados y ser referencia para muchas otras. También la crisis y el continuo empeño de las empresas en reducir gastos y aumentar la productividad ha precipitado esta situación. Desde 1985 se ha reducido en más de un tercio el espacio que ocupa un empleado, desde 121m^2 a los 75m^2 actuales. En españa se calcula que mientras un entorno de trabajo clásico cada empleado necesita un ratio de 15 metros, ahora se pasa a un ratio de 10 metros, mientras que en Reino Unido han pasado desde 1997 de 16,6m^2 por persona a 11,8m^2. Esto implica un importante ahorro para la empresa en cuanto a espacio en la oficina, ya que consiguen tener a los mismos empleados en un entorno más pequeño y barato, y una optimización de la misma más grande de lo que parece a simple vista, ya que consiguen deshacerse del principal elemento que ocupa más espacio en una oficina: los despachos. Sobre todo si tenemos en cuenta el deshuso que tienen la mayoría de ellos estando vacíos gran parte de la jornada laboral.

Esto no implica que en las Oficinas Abiertas todo se haga de manera abierta y que desaparezcan las zonas más privadas e íntimas de los entornos laborales. Se mantienen salas habilitadas para reuniones internas que requieran privacidad o conferencias, se mantienen zonas de mesas y sofás para reuniones improvisadas o no tan formales, y se crean nuevas zonas comunes para aumentar la satisfacción de los empleados, como puede ser

una zona recreativa (con billares, ping pong, etc...) donde puedan desconectar durante unos minutos, o zonas de relax donde el trabajador pueda descansar.

Aunque uno de los motivos por los cuales se pensó en espacios abiertos y compartidos era para que el jefe pudiera de un simple vistazo comprobar que todos sus empleados estaban en su puesto realizando su trabajo, cambiar a este tipo de oficinas tiene una serie de ventajas que se explican a continuación.

Ventajas

Como ya hemos adelantado en el apartado anterior, una de las ventajas que tiene este método de trabajo y que más atrae a las empresas son sus ventajas económicas. La disminución de la densidad ocupada por los trabajadores se convierte en un ahorro para la empresa, ya sea en alquiler del terreno o mantenimiento de las instalaciones. Cada vez son más empresas las que tratan de maximizar el espacio y reducir los costes al mínimo. Menos paredes producen un espacio más aprovechable lo que nos lleva a una optimización de la distribución de la oficina. El ahorro económico no solo puede venir de este punto, si no también de cambios que se puedan producir. Es mucho más fácil realizar un cambio de distribución de empleados en un entorno grande y abierto que en cualquier otro lugar. Este caso se puede dar cuando se hacen cambios en los equipos de trabajo y varios empleados deben cambiar su sitio asignado. También facilita futuras implementaciones de nuevas tecnologías y la inversión inicial requiere entre un 15% y un 30% menos que para las oficinas tradicionales.

La principal ventaja y la más importante es el aumento de la productividad. Esto viene debido a que, normalmente, la distribución de los empleados en la oficina se realiza por grupos de trabajo, lo que provoca un aumento de la comunicación interna del grupo, una mejor retroalimentación de las tareas, flujo de ideas y la colaboración entre sus miembros cuando surgen problemas o tienen información relevante del proyecto. A su vez, esto provoca un mayor provecho del tiempo por parte de los empleados, puesto que al tener una comunicación más directa entre ellos y aumentar la comunicación informal evitan tener que realizar reuniones formales con el tiempo que conllevan. Al trabajar en la misma mesa que el equipo y mejorar la comunicación en momentos puntuales donde es vital para el correcto desarrollo de un proyecto, se da un aumento de rendimiento, velocidad y precisión de trabajo.

Pero no solo son las comunicaciones internas las que se ven beneficiadas en las Oficinas Abiertas, si no que también mejoran las externas con otros grupos de trabajo. Al ser el entorno de trabajo un espacio común y estar repleto de zonas comunes para los empleados hace que puedan socializar mucho más con el resto de la empresa. Esto puede tener beneficios cuando dos equipos deben mantener una estrecha comunicación porque sus proyectos están relacionados.

Para que estos cambios en los diseños de oficina tengan éxito se debe planificar con mucho cuidado. Es de vital importancia que la empresa tenga un buen mantenimiento de sus redes y tecnología en general, puesto que elementos como el wifi cobran una gran importancia para poder moverse por la empresa sin tener que perder el tiempo en elementos externos a su trabajo.

Desventajas

Las Oficinas Abiertas también suponen un reto para conseguir minimizar sus posibles desventajas y así poder disfrutar de todas las ventajas explicadas en el apartado anterior. Una mala aplicación de este diseño de oficina o su implementación por los incorrectos motivos puede provocar que consigamos todo lo contrario a lo que queríamos conseguir.

El primer problema que puede surgir es la falta de privacidad que puede sentir un empleado trabajando en un entorno abierto. Ya sea porque tiene una reunión con algún cliente de manera física o por teléfono, porque se sienta observado por la cantidad de gente que tenga alrededor o porque se sienta más controlado. Es por ello que es muy importante tener suficientes zonas habilitadas para reuniones o para momentos en los que una persona necesita una mayor privacidad.

Otro de los problemas más comunes son las distracciones. Pueden ser de distinta índole: por ruido, porque una persona pasa por delante tuyo, por motivos de temperatura o luz del entorno, etc. Infravalorar este problema podría terminar en empleados frustrados por la situación y su consecuente pérdida de productividad y motivación. Al trabajar en un entorno grande, con su reducción de muros y el respectivo aumento de empleados que ocupan un espacio único donde la comunicación es uno de sus puntos fuertes, es inevitable que los empleados hablen entre ellos, suenen teléfonos y termines trabajando en un ambiente demasiado ruidoso. Una reunión improvisada de un equipo que se esté realizando una fila de mesas por detrás de la tuya, otra reunión de otro equipo que termina con aplausos porque han conseguido llegar al objetivo que se habían propuesto o dos personas que van hacia la cafetería hablando entre ellos y pasan por delante de tu mesa son algunas de las situaciones que se pueden dar normalmente en un ambiente de trabajo de este índole. Es por ello muy importante que la empresa ponga todas las herramientas posibles para evitar este tipo de problemas y ayudar a crear un espacio de trabajo agradable para todos sus empleados y que fomente la concentración.

Por último hablaremos del principal problema que nos podemos encontrar en una Oficina Abierta y en el que se centra todo el proyecto que explicamos en esta memoria: las interrupciones. Es cierto que este diseño de oficina pretende promover la frecuencia de interacciones no controladas para reducir el tiempo de las tareas y aumentar la productividad, pero si no se controla y no se realiza correctamente puede conseguir todo lo contrario. Los mejores resultados se obtienen en periodos relativamente altos de concentración. Suponiendo que una persona consigue realizar dos horas útiles cada tres horas de trabajo, no es lo mismo realizar una hora completa concentrado en su tarea, tener una interrupción de treinta minutos y repetir el mismo proceso, que estar con su tarea diez minutos, que lo interrumpan los cinco minutos siguientes, luego conseguir estar con su tarea otros veinte minutos, otra interrupción de diez minutos, y así sucesivamente. Las interrupciones mal gestionadas son el mayor peligro que se puede encontrar una empresa o un equipo de trabajo en su objetivo de ser ágiles y eficientes. Un abuso de las mismas puede llevar a retrasos en los proyectos, en las tareas e incluso en la desmotivación del empleado al ver que no avanza.

Para evitar esto se dan una serie de consejos, pero al fin y al cabo cada empresa escoge lo que cree que se adapta mejor a su entorno laboral para solucionar este tema. Con este trabajo de fin de grado nosotros proponemos una solución alternativa: Pomodoros.

1.2 Metodología Pomodoro

El cerebro humano es incapaz de estar en un estado de concentración permanente, por lo que gestionarse el tiempo es la mejor forma de sacar nuestro máximo rendimiento. Básicamente en esto consiste la técnica pomodoro, un método de gestión de tiempo que nos permite focalizar toda nuestra concentración en periodos cortos de tiempo y realizando breves descansos para poder recuperar energías y descansar la mente. Esta metodología es fácil de aprender y muy simple de utilizar, además de que apenas se necesita nada fuera de lo común para poder usarlo, simplemente un reloj. Aunque se recomienda un reloj de cocina en forma de tomate (así es como comenzó a usarse esta metodología) sirve cualquier aparato capaz de hacer una cuenta atrás de 25 minutos.

Tal y como vemos en la tabla 1.2.1 adjunta, lo primero que debemos hacer cada día es dividir en tareas pequeñas todo el trabajo que queramos hacer durante el día. Una vez tengamos esa lista la podemos priorizar para seguir un orden y comenzar con el primer pomodoro. Los pomodoros están formados por bloques de 30 minutos, de los cuales 25 se dedican a la realización de la tarea y los otros 5 para descansar. Cuando se termina el cuarto pomodoro se recomienda hacer un breve paréntesis más largo de unos 15-30 minutos.

Algoritmo Pomodoro	
1	Realizar lista de tareas
2	Escoger tarea a realizar
3	Realizar pomodoro de 25 minutos
4	Actualizar lista de tareas y añadir posibles observaciones
5	Descansar 5 minutos (15-20 minutos si es el cuarto pomodoro)
6	Al final del día repasar las observaciones de los pomodoros con el objetivo de aprender y mejorar

Tabla 1.2.1. Enumeración de las etapas de la metodología

Una vez tengamos la tarea a realizar seleccionada ponemos la cuenta atrás de 25 minutos en nuestro dispositivo escogido. Nuestro principal objetivo durante este tiempo será dedicarle todo el tiempo única y exclusivamente a nuestra tarea. Es importante evitar distracciones o interrupciones. Los pomodoros no se pueden dejar a mitad ni interrumpir, si durante el transcurso del mismo por algún motivo no puede continuarlo no se puede pausar hasta que vuelva a reanudar la tarea. En este caso el pomodoro debe quedar cancelado, apuntar el motivo de la interrupción para intentar que no vuelva a suceder en los siguientes, y volver a comenzar uno nuevo cuando vaya a reanudar la tarea.

Los tiempos no son negociables ni extensibles. Una vez se terminen los 25 minutos no te puedes quedar unos minutos de más porque estés a punto de terminar la tarea o porque estés completamente concentrado y no veas conveniente un descanso. Tampoco se puede tomar el descanso antes de tiempo si la tarea dura menos del tiempo programado. En este caso se pasa a comenzar la siguiente tarea de la lista aunque se quede a medias al terminar el periodo de concentración o aprovechar los minutos restantes para hacer un repaso de la lista de tareas y los comentarios que hayas podido ir anotando en el transcurso del día.

Es de vital importancia aprovechar los 5 minutos de descanso para “desconectar” del trabajo y recuperar energías. En caso contrario podría darse el caso que dentro de 2-3 pomodoros el nivel de concentración descienda y empieces a perder tiempo útil de trabajo, que es exactamente lo que intenta evitar esta metodología.

1.2.1 Objetivos

Esta metodología tiene dos claros objetivos: conseguir que una persona sea más eficiente en su tiempo de trabajo y evitar las interrupciones. Esto se consigue intentando aumentar día a día el número de pomodoros realizados con éxito. Siendo más eficientes conseguiremos aumentar nuestra productividad y reducir los momentos en los que en plena tarea nos desconcentramos porque suena nuestro móvil, porque se nos viene a la cabeza algo que teníamos que consultar por internet (ya sea relacionado con la tarea o no) y pensamientos que provocan una distracción de la tarea y que al fin y al cabo también pueden considerarse interrupciones internas. En cambio, evitar las interrupciones (o interrupciones externas) en un entorno de trabajo abierto y social simplemente tenemos que avisar a la persona que nos interrumpa de que estamos a mitad de un pomodoro y que venga dentro de 25 minutos. También se pueden realizar pomodoros destinados a ayudar a un compañero. Por ejemplo, si estamos en medio de un pomodoro y nuestro compañero nos interrumpe porque necesita ayuda para terminar un documento, podemos incluir una tarea en nuestra lista que consista en ayudarle. Así conseguimos terminar nuestra tarea sin interrupciones y le podremos dedicar el tiempo necesario con el siguiente pomodoro. En caso de que sea una tarea urgente y que no pueda esperar, no habrá más remedio que dar por mal realizado el pomodoro, apuntar el motivo por el cual no se ha podido completar, resignarse e intentar que no vuelva a suceder.

1.2.2 Relación con Metodologías Ágiles y la Ley de Parkinson

A estas alturas ya nos habremos dado cuenta de que esta metodología tiene una cierta relación con la Ley de Parkinson. En concreto con la que se formula en la Tabla 1.2.2.

Ley de Parkinson	
1	El trabajo se expande hasta llenar el tiempo de que se dispone para su realización

Tabla 1.2.2. Ley de Parkinson, Cyril Northcote Parkinson, 1957

Ésta nos indica que las tareas nos llevan todo el tiempo que le asignemos. Si una tarea que lleva una tarde completarla, para ir sobre seguro le asignamos dos días, terminaremos tardando los dos días en terminarla. ¿En qué influye aquí la metodología Pomodoro? En que nos ayuda a fraccionar las tareas en periodos de 25 minutos enseñándonos el valor del tiempo. Hace que aumente nuestra productividad y veamos como nuestras tareas avanzan. No nos sirve de nada escoger una tarea y no asignarle una estimación y ver como ésta se nos alarga entre varios pomodoros sin que podamos controlarla. Con esta metodología la unidad de tiempo es un "pomodoro", por lo que debemos dividir nuestras tareas de manera que se puedan realizar en un pomodoro. Si una tarea creemos que puede abarcar varios pomodoros, se puede dividir en subtareas. Siempre debemos ser conscientes de nuestro objetivo durante el pomodoro y no dedicarnos únicamente a trabajar durante los 25 minutos en algo que vayamos viendo sobre la marcha o sin tener muy claro cuál es nuestro objetivo. Así no aumentaríamos nuestra eficiencia.

Y aquí es donde ya comenzamos a entrar también en las metodologías ágiles, en concreto, en Scrum. El objetivo de este trabajo y de este informe no es el de entrar a explicar en qué consiste Scrum ni cómo se trabaja bajo esta metodología, así que no entraremos a explicarla a fondo. Únicamente entraremos a valorar que también consiste en medir las tareas a realizar en esfuerzos y, si una tarea requiere de un esfuerzo muy grande, nos

enseña a cómo dividirla en subtareas de esfuerzos más pequeños. Por tanto, podemos ver como la metodología Pomodoro es completamente compatible con una metodología ágil como puede ser Scrum.

1.2.3 Desventajas

Un problema común es que esta metodología es incapaz de predecir el nivel de cansancio psicológico que requiere cada tarea (obviamente). Por tanto, trabajando de esta manera partes de la base de que todas son iguales cuando esto no tiene por qué ser cierto. Hay tareas que requieren de una mayor concentración y que cuando terminas con ellas necesitas unos minutos más para recuperarte, y otras que con los 5 minutos de descanso es más que suficiente para poder ponerte con la siguiente. También puede resultar frustrante al principio si nunca has hecho estimaciones de tareas, ya que raramente conseguirás dividir tus tareas en periodos de 25 minutos y verás como pomodoro tras pomodoro no consigues terminarlas.

Por último, vemos como los pomodoros son inflexibles en cuanto a su duración. Se debe trabajar estrictamente durante 25 minutos y descansar otros 5. ¿Pero cuál es realmente el tiempo óptimo de trabajo de una persona? Pues depende de cada uno. Hay gente que estar parando cada 25 minutos les provoca una interrupción y un descenso de la productividad. Esta gente requiere de tiempos más altos de concentración, como pueden ser 40 minutos o incluso una hora. Otra gente puede tener el caso contrario, que por las circunstancias que sean (como puede ser por su entorno laboral o su puesto de trabajo) no pueda estar 25 minutos con una tarea. ¿Qué sucede en estos casos? Que realizar pomodoro no es nada recomendable para este tipo de personas, porque conseguirán lo contrario a lo que buscan gestionando su tiempo de esta manera.

1.3 Motivación del problema

Bajo este conexto y ésta problemática se crea el proyecto que se explica en este informe. Lo que me llevó a interesarme por él fue que he podido comprobar personalmente todo lo que explico en apartados anteriores sobre las Oficinas Abiertas. Y sobre todo el gran problema que suponen las interrupciones junto a su consecuente pérdida de tiempo y contexto de la tarea que estabas realizando. He visto retrasarse proyectos por una mala planificación, tareas por una mala gestión que hace una persona de su tiempo y cómo las interrupciones improvisadas unidas a reuniones hacen que hayan días que apenas puedas avanzar nada en tus tareas.

Por estos motivos nos parece muy interesante cualquier método utilizado que trate de minimizar estos problemas y, por ello, este trabajo de fin de grado consiste en proponer un nuevo software que se pueda implementar en grupos de trabajo para ayudarles a mejorar su concentración, ser más eficientes y evitar lo máximo posible cualquier factor interno y/o externo que les distraiga de su tarea.

Sin intentar entrar en muchos detalles del proyecto (ya que está explicando a partir del siguiente apartado), también me gustó mucho la manera de plantearlo que tuvo mi profesor J.Daniel Prades al encararlo más hacia un software de introducción a una metodología de trabajo, cuyo objetivo tiene concienciar a la gente del problema y enseñarles a gestionar su tiempo a la vez que se divierten utilizándolo. Y descartar la opción más común en estos casos de diseñar un software que controle cómo trabaja la gente, que aplican correctamente la metodología, que realizan todos los pomodoros correctamente al cabo de un día y que se acaba convirtiendo para los usuarios una pesadez utilizarlo cada día.

Además del motivo funcional que puede tener esta aplicación también me gustaron las tecnologías a utilizar. Aunque las detallaremos mejor más adelante, el realizar el software en Java y tener la oportunidad de profundizar más en un lenguaje que está entre mis destacados me dio una motivación extra.

También me parece que éste es un tema que las empresas no se toman lo suficientemente en serio y no se esfuerzan en evitarlo hasta que no es demasiado tarde y el proyecto o las tareas están demasiado retrasadas. Al fin y al cabo, el éxito o el fracaso de un proyecto depende en gran medida de los empleados. Si ellos no están en un entorno de trabajo amigable y que les deje trabajar a su gusto, da igual la planificación que haya y las prisas que pueda tener la empresa. El proyecto se retrasará. Por tanto, creo que cualquier ayuda que se le pueda aportar a un trabajador en una Oficina Abierta es de utilidad y ésto intentamos en este trabajo de fin de grado.

Todos estos motivos me llevaron a pensar que de aquí se podría sacar un interesante proyecto y que, como mínimo, me ayudaría a mi mismo a informarme más sobre la problemática, posibles soluciones y a gestionarme mejor el tiempo en mi lugar de trabajo. Y si además consigo ayudar a alguien más podré darme por satisfecho.

2 Proyecto

Como ya se ha comentado muy por encima en la introducción anterior, este proyecto consiste en dar una solución alternativa a la gestión del tiempo de trabajo y a la problemática de las interrupciones. Una vez hemos conseguido situar contextualmente nuestro software, en los próximos apartados pasamos a explicar todos los detalles del mismo.

2.1 Descripción y Objetivos

El proyecto consiste en una aplicación cliente/servidor destinada a grupos de trabajo conectados por intranet. Es decir, no sólo se busca ayudar a la autogestión del tiempo de una persona en concreto, si no de todo un grupo de trabajo. Para evitar las interrupciones en el entorno de trabajo no es sólo la mentalidad del empleado la que debe cambiar, si no la de todo su entorno. Por ello, no sólo se crea una herramienta para ayudar a un usuario, si no a todo el equipo. ¿Cómo se consigue esto? Informando a cada usuario del estado de sus compañeros en todo momento. Es por ello que con nuestro software hay dos formas de comprobar dicha información. La primera es desde el propio cliente de la aplicación, donde hay un apartado de estadísticas en el cual aparece el estado actual de todos los miembros del grupo en el que el usuario esté asignado. La segunda es desde un pulsador que va conectado al pc de cada usuario mediante USB, que controla el cliente de la aplicación y del que hablaremos más adelante. Dicho pulsador va cambiando de color según el estado en el que se encuentre su propietario, por lo que de un simple vistazo una persona puede ver el color del pulsador de otra persona. Y en caso de que no lo tenga a la vista o no estén trabajando en la misma zona, tendrá la primera opción comentada anteriormente. Junto a este dato también aparece el tiempo restante que le queda en ese estado, por lo que puedes saber cuándo puedes interrumpir a una persona y cuando no.

Además del objetivo de reducir las interrupciones y, en consecuencia, aumento de la productividad, el principal objetivo que tiene este software es el de introducir a los empleados en la metodología que proponemos (basada en pomodoros) y que está explicada en el próximo subapartado. Es por ello que la idea original no es llevar un control de quién trabaja más y quién trabaja menos o quién consigue realizar pomodoros correctamente y quién no consigue realizar ninguno en todo el día. El objetivo es darle al empleado una alternativa para su gestión del tiempo, que vea las mejoras que ésto puede provocar en su día a día en el trabajo, que le motive y que se esfuerce por conseguir completar los pomodoros correctamente. Para ello introduciremos un pequeño factor competitivo, ya que de la misma manera que un usuario podrá ver el estado actual en el que se encuentra otro para poder interrumpirle o no, también podrán ver las puntuaciones de cada usuario y ver quién es el que más pomodoros consigue realizar hoy, esta semana, el que consiguió más la semana pasada y el que lleva más tanto este mes como el pasado.

Al introducir este pequeño factor competitivo para estimular a la gente tenemos que empezar a controlar los pomodoros correctos y que ningún usuario intente engañarnos diciendo que ha realizado uno siendo mentira. Por lo que el software controlará lo estrictamente necesario del ordenador de un usuario para comprobar que está trabajando. Y hacemos especial hincapié en "estrictamente necesario", puesto que uno de los objetivos que buscamos es que la aplicación sea lo menos invasiva posible. Que no distraiga al usuario durante sus intervalos de trabajo y que no sienta que la privacidad de su ordenador pueda verse comprometida. Es por ello que, como veremos más adelante, lo único que controlará el software son los eventos de teclado y ratón.

Por tanto, podemos resumir los objetivos en los siguientes puntos:

- Disminuir en la mayor medida posible las interrupciones que sufre un empleado.
- Aumentar la productividad.
- Introducción de una nueva metodología de trabajo para ayudar en la gestión del tiempo.
- Introducir un pequeño factor competitivo para fomentar su uso.

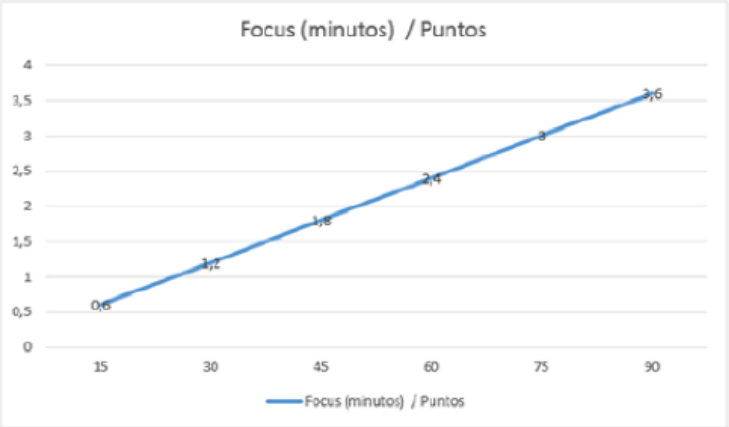
2.2 Variaciones de la metodología Pomodoro

La metodología propuesta es una variante de la de pomodoro explicada en el apartado 1.2. El objetivo de nuestro método de gestión de trabajo consiste en hacerlo más flexible que el pomodoro original y que se pueda adaptar a varios tipos de personas dependiendo de su capacidad de concentración. Es por ello que nuestros intervalos de trabajo no serán de 25 minutos, si no que será completamente configurable por el usuario en un intervalo entre los 15 y los 90 minutos, con incrementos y decrementos de 5 minutos. De esta manera una persona puede realizar un intervalo de trabajo acorde a unas variables que sólo el propio usuario puede medir:

- Su capacidad de concentración.
- El desgaste psicológico que conlleva la tarea a realizar.
- Duración de la tarea o de tiempo disponible del que dispone hasta algún evento.

Por lo que respecta al periodo de descanso si que nos parece apropiado mantenerlo, obviamente adaptado al cambio de tiempo que pueden sufrir los pomodoros, pero manteniendo el mismo factor trabajo/descanso. Si configuramos un pomodoro de 25 minutos mantenemos los 5 minutos de descanso, pero si configuramos un pomodoro de 50 minutos tendremos 10 minutos de descanso al finalizarlo correctamente.

Otro cambio importante es la medida de los periodos de trabajo completados. Mientras que la unidad de medida de la metodología original son "pomodoros", en nuestra propuesta otorgamos puntos. Cuanto más dure un pomodoro más puntos otorgamos al usuario, mientras que cuanto menos tiempo dure menos puntos recibirá. De esta manera conseguimos puntuar todos los pomodoros bajo el mismo baremo independientemente de su duración, y saber el tiempo que ha estado el usuario en un intervalo de trabajo al final del día. Éste dato nos es muy útil para poner un límite al uso de la metodología y no permitir a ningún usuario hacer más de 16 puntos al día, que vendrían a ser 8 horas de trabajo.



Focus	Relax	Puntos
15 min	3 min	0,6 ptos
20 min	4 min	0,8 ptos
25 min	5 min	1 punto
30 min	6 min	1,2 ptos
...	...	...
75 min	15 min	3 ptos
80 min	16 min	3,2 ptos
85 min	17 min	3,4 ptos
90 min	18 min	3,6 ptos

Tabla 2.1.1. Puntos según el tiempo del Pomodoro

Otro de los cambios que introducimos es la eliminación de los 15 minutos de descanso cada cuatro pomodoros. Esto no es porque creamos que no son necesarios, si no porque es una herramienta de introducción a ésta metodología que intenta enseñar y ayudar a grupos de usuarios a gestionarse mejor. Esto significa que, como hemos explicado anteriormente, no llevamos ningún control estricto de aplicación de la metodología y no se trata de que cada persona realizace obligatoriamente los 16 puntos al cabo de un día. Es más, al ser un método de enseñanza se trata de que poco a poco vayan consiguiendo completar más pomodoros correctamente y vayan aumentando sus estadísticas, suponiendo de antemano que los primeros días/semanas de uso y de intento de cambio en su forma de trabajar les resulte difícil realizar muchos pomodoros completos. Además, en un entorno laboral hay una serie de factores que ninguna metodología es capaz de predecir, como pueden ser reuniones de urgencia. Por tanto, como mucho podemos aconsejar tomarse un descanso mayor cada cierto tiempo, pero nunca obligar al usuario con nuestro software tomarse un descanso mayor cuando puede darse el caso de que entre su segundo y tercer pomodoro hayan pasado dos horas sin poder realizar ninguno.

2.3 Software no intrusivo. Colores y estados.

Recordemos que uno de los objetivos del software es molestar lo menos posible al usuario en su trabajo y ser lo menos invasivos posible en su ordenador. Es por ello que nada más ejecutarlo, automáticamente éste pasará a ejecutarse en la bandeja de Windows. Si abres la aplicación desde ahí aparecerá una interfaz desde la cual puedes manejar todos los aspectos del programa y que incluso trae una simulación del funcionamiento del pulsador USB. Es importante saber que todo lo importante de la aplicación puede manejarse desde el propio pulsador, de manera que el uso de la interfaz no es estrictamente obligatorio y que puedes realizar pomodoros y saber el estado del actual sin necesidad de abrirla en ningún momento. En el apartado 2.5 se explica el funcionamiento de éste USB Button.

También puedes configurar a tu gusto el nivel de avisos que quieres que te proporcione el software. Puedes configurarlo de manera que no te moleste en ningún momento, o pedirle que te vaya sacando mensajes cada vez que cambia tu estado en forma de tooltip y/o incluso pedir que suene un pequeño pitido para informarte de los cambios. También si pones el ratón encima de la bandeja de windows te informará de tu estado actual y del tiempo restante que te falta para terminar dicho estado.

Tal y como podemos observar en la Figura 2.3.1, un usuario puede pasar por seis estados distintos, cada uno con su respectivo color. También se adjunta una breve descripción del significado de cada uno de ellos.

Estado	Color	Descripción
Ready	Verde	El usuario está listo para comenzar un pomodoro
Focus	Azul	Se está realizando un pomodoro
Relax Time	Rojo	El usuario está en el tiempo de descanso
Warning	Amarillo	Estado inactivo. El pomodoro se prepara para cancelarse
Push Pomodoro	Amarillo	El pomodoro ha terminado y falta la confirmación
End Day	Rojo	El usuario ya ha realizado el máximo de pomodoros hoy

Tabla 2.3.1. Color y descripción de los distintos estados del pomodoro

Por último añadir que el color del icono del software en la bandeja de windows se mantiene actualizado con el estado en el que te encuentras.

2.4 Pomodoros correctos e incorrectos

Ya entrando más a nivel del software pasamos a explicar lo que se considera un pomodoro correctamente terminado y un pomodoro incorrecto.

Un pomodoro incorrecto o mal realizado es aquel en el que el usuario no consigue trabajar durante todo el transcurso del mismo debido a que ha tenido alguna interrupción, ya sea interna o externa. El software detecta estas interrupciones observando los eventos del ordenador, tanto del ratón como del teclado. Si durante más de dos minutos no detecta actividad comienza una cuenta atrás de diez segundos donde espera que el usuario vuelva a mostrarla. Durante este tiempo y con el fin de alertar al usuario, comienza a parpadear rápidamente alternando el color del estado Focus (azul) y el del estado Warning (amarillo). Si el usuario sigue sin mostrar actividad pasados esos diez segundos, el pomodoro se cancela y pasa nuevamente a estado Ready. En cambio, si el usuario vuelve a mostrar actividad se vuelve al estado Focus y se continúa con el pomodoro como si no hubiese pasado nada. También se considera incorrecto aquel pomodoro que no haya sido confirmado. Cuando se termina el tiempo, durante los siguientes 30 segundos, el software pide que confirmes la finalización del pomodoro. De esta manera controla que sigas atento a tu tarea y no te hayas ido un minuto antes o unos segundos antes. Para alertar al usuario de que ha finalizado el tiempo y que debe pasar al tiempo de relax comienza a parpadear en amarillo. Se confirma correctamente el pomodoro pulsando el USB Button o su equivalente en la interfaz del programa. En caso de que el tiempo expire sin confirmación, el pomodoro pasará a cancelarse y volverá al estado Ready, mientras que si se confirma con éxito pasará al estado Relax Time que durará lo que se haya establecido en la configuración previa del pomodoro. Este estado de confirmación se llama “Push Pomodoro”. Por último, si se cierra la aplicación en mitad de un pomodoro también se cancelará y se pasará al estado Ready cuando se vuelva a ejecutar.

Por tanto, podemos definir un pomodoro correcto como aquel en el que el usuario ha mostrado actividad laboral durante toda su duración y que ha confirmado el pomodoro una vez finalizado.

Durante el tiempo de Relax Time el usuario no podrá iniciar ningún otro pomodoro hasta que termine el tiempo y vuelva al estado Ready.

2.5 Funcionamiento USB Button

El software viene junto a un pulsador USB que se conecta al ordenador. Dicho hardware es controlado en todo momento por nuestra aplicación que lo mantiene sincronizado con el color del estado en el que se encuentre el usuario, de esta manera tus compañeros conocerán el estado en el que te encuentras y también te servirá a ti mismo para conocer de un simple vistazo si continúas en el mismo estado o ha cambiado.

Desde el USB Button se puede manejar todo lo esencial del software. Puedes conocer tu estado actual conociendo la tabla de colores 2.3.1. De esta manera sabes si puedes comenzar un pomodoro, si estás en medio de uno, si estás en el estado Warning, etc. Para iniciar un pomodoro únicamente debes pulsar el USB Button (obviamente estando en el estado Ready) y el software detectará dicha pulsación y comenzará el pomodoro con la configuración del último que hayas realizado. Dicha configuración únicamente se puede cambiar desde la interfaz que se puede encontrar en el modo tray.

Cuando lleves 1/4, 2/4 y 3/4 del Pomodoro completado, el USB Button te avisará con uno, dos o tres breves parpadeos, según el momento en el que estés. Si debes confirmar el Pomodoro o está a punto de cancelarse por inactividad, el parpadeo será mucho mayor y de color amarillo para llamar tu atención. En el caso de la confirmación deberás pulsar el USB Button para dar por finalizado el Pomodoro, mientras que en el estado Warning servirá con pulsarlo o provocar cualquier evento con el ordenador, ya sea pulsando alguna tecla o con el ratón.

En caso de estropearse el USB Button o de no tener ninguno, no pasa absolutamente nada. Desde la interfaz se puede manejar con total normalidad y, tal y como se ha explicado en el apartado 2.3, se puede configurar el nivel de mensajes que quieres que te envíe el software para avisarte de los estados.

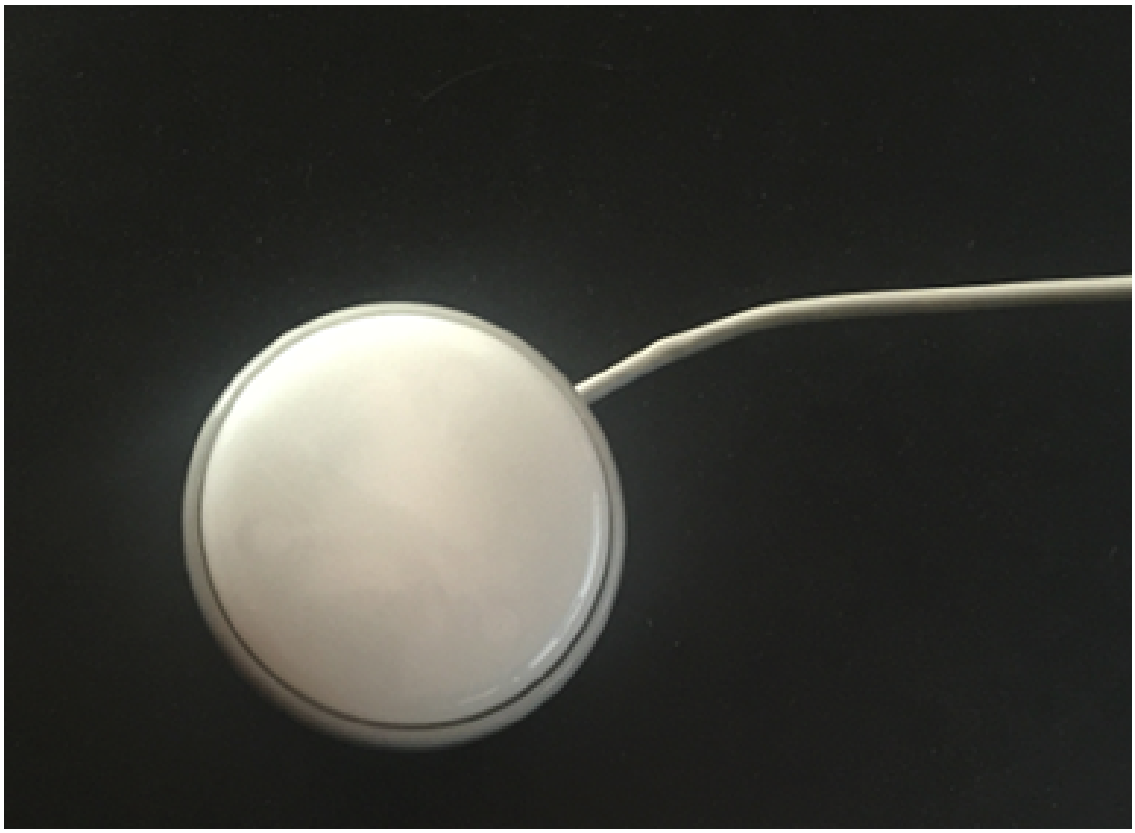


Tabla 2.5.1. Imagen del USB Button

2.6 Grupos

Para enfocar el software a la gestión de grupos de trabajo, los usuarios podrán pertenecer a uno o más grupos. En caso de pertenecer a dos grupos o más, dicho usuario podrá escoger en qué grupo realiza cada uno de sus pomodoros, seleccionándolo desde la configuración de la interfaz.

En cuanto a las estadísticas mencionadas en apartados anteriores, cada usuario sólo podrá ver los estados y los tiempos restantes de los usuarios asignados en su mismo grupo, pero en ningún momento podrá ver la información de los demás grupos.

2.7 Metodología

Para que un proyecto acabe con éxito, su planificación y metodología es vital. Un proyecto con una mala gestión puede llevarnos al más absoluto fracaso independientemente de las horas que se le dediquen.

Algunas de las características a tener en cuenta a la hora de realizar un trabajo de fin de grado son las siguientes:

- **Proactividad.** Se espera que el alumno aprenda a ser proactivo para adquirir los conocimientos necesarios para el correcto desarrollo del software.
- **Supervisión.** Un tutor supervisa cómo va el trabajo y aconseja. Debes ir mostrándole los avances.
- **Gestión de tiempo.** Sirve para aprender a gestionar y predecir el tiempo de trabajo en un proyecto de mayor envergadura.
- **Análisis y decisiones.** El alumno debe ser capaz de analizar la aplicación a construir y tomar decisiones, ya sea sobre las tecnologías a utilizar o sobre problemas que vayan surgiendo durante el transcurso del mismo.

Analizando estos puntos y las metodologías que cada vez se utilizan más en entornos laborales me pareció un buen momento para aprovechar e implementar una forma de trabajo que me sirviese también para aprender y adaptarme a la forma de trabajar que me puedo encontrar, siempre adaptado a las circunstancias y limitaciones que se puede uno encontrar. Mirando detenidamente punto a punto la lista anterior en seguida nos daremos cuenta de que encajan a la perfección con las metodologías ágiles. Es por ello que este trabajo se ha gestionado con SCRUM.

2.7.1 ¿Por qué SCRUM?

La metodología Scrum se adapta a la perfección a la forma de trabajar que considero óptima en un trabajo de este tipo. Dicha metodología promueve equipos auto-gestionados, comprometidos y autónomos, además de trabajar en iteraciones cortas de tiempo (sprints) para poder mostrar al cliente (Product Owner) el trabajo realizado para que opine de él. Cada sprint requiere de una planificación, análisis, creación y comprobación de lo que se

va a realizar en el mismo y de lo que se entregará al finalizar. De esta manera podemos adaptar esta metodología a lo que necesitamos para poder cumplir con las características mencionadas anteriormente y maximizar el beneficio que podemos sacar de las mismas.

También añade un punto muy significativo y es el hecho de tener que dividir las tareas en subtareas atómicas que aporten algo de valor por si mismas y que nos permitan una mayor organización y visibilidad de todo lo realizado y todo lo que queda por hacer. De esta manera se puede medir el ritmo de trabajo que lleva el grupo (en este caso yo) y ver hasta donde se puede llegar.

Obviamente no se puede aplicar Scrum al pie de la letra como se podría hacer en una empresa, pero sí que se ha orientado hacia esta forma de trabajar de una manera que se detalla en el siguiente apartado.

2.7.2 Implementando Scrum en nuestro TFG

Teniendo en cuenta varios consejos se decidió hacer sprints de dos semanas. De esta manera se puede ir supervisando el desarrollo del producto, tener nuevas funcionalidades que enseñar y detectar a tiempo posibles errores, mejoras y el estado global del proyecto. Cada dos semanas nos reuníamos J.Daniel Prades y yo para analizar el estado del trabajo realizado, buscar puntos de mejora, sugerencias, ver el estado global del proyecto y para detallar lo que se podría realizar en el siguiente sprint. En esta reunión ya agrupábamos los objetivos que tienen un sprint planning, el sprint review y la retrospective. Empezábamos por lo que sería la Review, donde mostraba los avances del software realizados durante ese sprint y le explicaba lo implementado. Continuábamos con la Restrospectiva, la cual analiza la forma en la que se hacen las cosas y cómo podemos mejorar. Analizar impedimentos que han surgido durante esas semanas y han influido en la no consecución de algún objetivo o si todo ha ido como la seda y se ha conseguido avanzar. Y por último lo que sería el sprint planning donde actualizábamos nuestro backlog con nuevas tareas que iban surgiendo y se analizaba el trabajo que se podía asumir durante las siguiente semanas. Como vemos, en este punto también asumíamos el Grooming, reunión generalmente previa al Sprint Planning que sirve para planificar y revisar el backlog con el fin de actualizarlo y priorizarlo.

1	Sprints de dos semanas
2	Profesor y tutor como Product Owner
3	Sprint Planning, Review y Retrospective el mismo día
4	Desaparición de los Daily Meeting
5	Product Backlog en formato excel compartido

Tabla 2.7.2.1. Resumen implementación Scrum en el TFG

Nuestro backlog consistía en un excel donde íbamos apuntando las tareas a realizar y que yo iba marcando como done una vez realizadas, testeadas por mi y que daba por finalizadas.

En cuanto a los Daily Meeting son imposibles de asumir en este tipo de proyectos universitarios, por lo que es importante dejarlo todo bien explicado en la reunión realizada cada dos semanas y utilizar el email únicamente para dudas que puedan surgir, pero no para realizar los Daily.

Por lo que respecta a la jerarquía de entidades de Scrum tampoco ha sido necesario utilizar la mayoría de términos para definir las unidades manejables. Recordemos que ésta

metodología define una serie de unidades manejables para dividir el trabajo. Normalmente se utilizan únicamente tareas, historias de usuario y épicas, pero a veces una épica no es suficiente y aún hay que hacer mayores agrupaciones. En este caso aparecen también los temas. En nuestro caso únicamente hemos necesitado utilizar tareas, puesto que ya éramos capaces de dividirlas y hacerlas lo suficientemente pequeñas y atómicas para que no requirieran de semanas y siempre se pudieran ver avances y nuevas funcionalidades al finalizar cada sprint.

1	**Proyecto**
2	**Tema:** Unidad de medida superior a las épicas. Puede englobar varias.
3	**Épica:** Agrupaciones de historias de usuario que definen bloques de trabajo dentro de un proyecto.
4	**Historia de Usuario:** Definición en lenguaje de negocio que hace el Product Owner de los requisitos de trabajo. Se puede descomponer en Tareas.
5	**Tarea:** Trabajo concreto normalmente realizado por una única persona del equipo. Estas tareas deben ser cortas y claras.

Tabla 2.7.2.2. Jerarquía de entidades Scrum

2.7.3 Planificación Inicial

La planificación inicial estuvo incluida en un periodo de 18 semanas, desde la primera semana de febrero hasta el final de la primera semana de junio. Era importante saber desglosar las tareas en unas más pequeñas para poder realizar una estimación del tiempo que creía que podría llevarme cada una de ellas y poder ver si era asumible todo lo planteado o no. Dicho desglose terminó siendo el siguiente:

- **Establecer Requerimentos:** Lo primero que se debe realizar en un proyecto es aclarar y definir las funcionalidades del proyecto para poder tener una visión general del mismo. Una vez aclarado punto a punto lo que debe tener el software se puede dar por concluida esta etapa. La previsión para su definición era la primera semana de proyecto.
- **Análisis Software y Tecnologías:** Una vez claro todo el proyecto y lo que hay que construir, es el momento de analizarlo más a fondo para poder escoger las tecnologías adecuadas para su desarrollo. Para ello se requiere una investigación de las mismas y sobre todo de las menos conocidas por mi. El tiempo inicial estimado para esta etapa es de una semana.
- **Investigación USB Button:** Aclarados los requisitos y escogidas las tecnologías a utilizar empezamos con el proyecto. El primer paso es centrarse en la clave de este proyecto, que es el pulsador USB. Para poder utilizarlo desde Java requiere una investigación previa donde se debe buscar documentación por internet y hacer diversas pruebas debido a la escasa información que se adjunta con el dispositivo. El periodo comprendido por esta etapa son las semanas 3 y la mitad de la 4 del proyecto.
- **Comunicación Cliente/Servidor:** Con la semana 5 del proyecto entramos en el mes de marzo y la idea a estas alturas es empezar a implementar lo que será la aplicación software. El primer paso de esta implementación consiste en crear la

comunicación entre el cliente y el servidor usando RMI, tecnología escogida en la etapa dos y que se explica en el apartado de Implementación más adelante. Esta etapa no debería requerir de demasiado tiempo al ser una tecnología ya conocida por mi, por lo que la estimación para hacerlo funcionar y poder hacer pruebas de comunicación es de 4 días escasos.

- **Interfaz Básica Software:** Realizada la comunicación es hora de comenzar con la interfaz. Esta etapa es esencial para poder continuar implementado lógica, por lo que es necesario tener aunque sea una versión inicial de la misma. No se espera tener una versión avanzada o final de la interfaz, únicamente hacer algo funcional para poder probar en las siguientes etapas que todo esté funcionando como debe ser.

- **Creación Login:** Uno de los campos realizados en la etapa anterior es el del login. Teniendo en cuenta para utilizar prácticamente todas las funcionalidades del software hay que estar registrado, es el primer paso a realizar. Se espera que esté terminado en la semana 6 de proyecto.

- **Implementación Lógica:** Con el login ya creado es hora de pasar a completar todas las funcionalidades requeridas en la primera etapa. Esta etapa es la más larga, ya que requiere implementar absolutamente toda la lógica del programa. Esto incluye crear la lógica de control de pomodoros, estadísticas, grupos... y unirlo todo a la interfaz. El tiempo estimado de desarrollo es de 4 semanas, hasta mediados de abril.

- **Desarrollo Interfaz:** Como última etapa del desarrollo se planea terminar la interfaz. Recordemos que hasta ahora tenemos una implementación básica, por lo que tenemos que dejarla realmente como queremos que sea en la versión final de la aplicación. Esta etapa debería durar un máximo de dos semanas.

- **Desarrollo Administrador:** Con el cliente y el servidor terminados pasamos a realizar la herramienta para administrador. Todo software debe tener su propia herramienta con la que poder gestionar una serie de opciones, como pueden ser modificar estadísticas o usuarios. Esta herramienta es completamente independiente al cliente, aunque se conecte al mismo servidor. Únicamente tendrá acceso a ella la persona que se ponga como administrador del software en su entorno de trabajo. El desarrollo esperado de esta etapa es de dos semanas.

- **Test Final:** En este punto ya tenemos todo el proyecto con su versión final y con cada etapa testeada individualmente. Es el momento de juntarlo todo y hacer una prueba final en un entorno lo más real posible. Este test se espera que dure todo el tiempo posible, lo que con esta planificación sería de 5 semanas. Esta etapa no solo sirve para buscar posibles bugs, si no también para poder cambiar algunos aspectos que no terminen de encajar o añadir mejoras, ya sean funcionales o de usabilidad a medida que se testea la aplicación.

- **Redacción Memoria:** A la vez que se realiza el test final se planifica realizar toda la escritura de la memoria. El tiempo estimado para la misma es de 5 semanas.

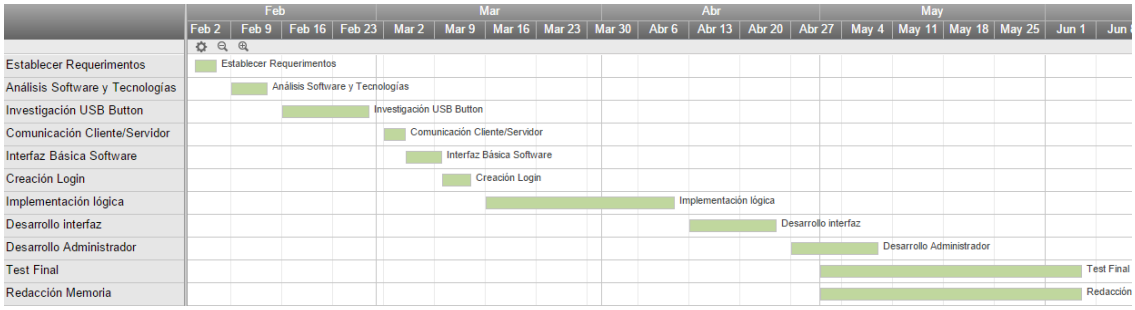


Tabla 2.7.3.1. Planificación Inicial

2.7.4 Planificación Real

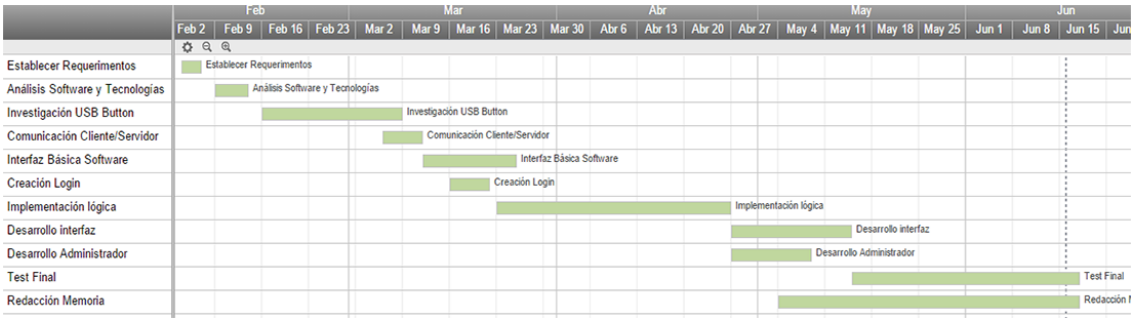


Tabla 2.7.4.1. Planificación Real

El apartado anterior explica la planificación inicial que hice y que estimé correcta en su momento. ¿Pero cuál ha sido el tiempo que me ha llevado cada etapa del proyecto? ¿Qué problemas han surgido que han podido hacer variar este planteamiento? Cuando se realiza un proyecto de media o larga duración las estimaciones suelen fallar por la aparición de problemas no previstos de antemano. Es importante saber reorganizarse y adaptarse a las situaciones para evitar que estos problemas e imprevistos tengan el menor impacto posible en el proyecto, su desarrollo y sobre todo que tengan el menor impacto en la fecha de finalización. En la figura adjunta se puede apreciar en forma de diagrama de Grantt el tiempo final que se ha dedicado a cada etapa y que pasamos a explicar.

- **Establecer Requerimentos:** Esta etapa tuvo una previsión correcta. Me reuní dos veces con J.Daniel Prades para poder definir todo lo que había que montar en el proyecto, las funcionalidades previstas y software externo que se iba a tener que aplicar, como es el caso del USB Button. Durante la misma semana que se nos notificó la asignación de este proyecto se realizó la etapa tal y como estaba previsto.

- **Análisis Software y Tecnologías:** El análisis de las tecnologías a utilizar también llevó el tiempo previsto. La semana siguiente a la del establecimiento de requisitos se destinó a decidir las tecnologías más adecuadas para el correcto funcionamiento del software. Se escogió utilizar RMI para crear la aplicación cliente-servidor, JSON para el intercambio de datos y JavaFX para la interfaz. La explicación detallada sobre estas elecciones están en sus respectivos apartados del punto de Implementación.

- **Investigación USB Button:** Hacer funcionar el pulsador USB desde la aplicación no fue nada fácil. De hecho considero que ha sido una de las etapas más difíciles de realizar. Esto significó que en esta etapa ya tuvimos un retraso de una semana respecto a la previsión inicial. Pasamos de realizar la investigación e implementación del pulsador en dos semanas (las dos finales de febrero) a tres, terminando al final de la primera semana de marzo consiguiendo poder comunicarnos con lenguaje Java con el pulsador.

- **Comunicación Cliente/Servidor:** La complejidad de la tarea cumplió la planificación, pero el tiempo en días destinado a la misma se vio afectada por el retraso de la investigación del USB Button. La previsión principal era comenzarla una vez terminada la anterior y que apenas llevara un día o dos su desarrollo. Al retrasarse una semana el conseguir hacer funcionar el pulsador decidí retrasar esta etapa unos días con el fin de poder dedicarme al hardware completamente. Al ver en ese momento que no sabía cuánto se me podría alargar esa etapa decidí readaptarme y planificar nuevamente estas etapas iniciales, por lo que inicié este desarrollo aún sin terminar la anterior etapa. El dedicarme a dos etapas a la vez provocó que, aunque la complejidad era la prevista, el tiempo destinado a esta tarea se alargara un poco más a lo que se puede ver en el diagrama de Grantt del apartado anterior.

- **Interfaz Básica Software:** Esta etapa también resultó más complicada de lo previsto en un primer momento, debido al uso de una tecnología que no había utilizado nunca hasta ahora. Había realizado interfaces simples con librerías conocidas, pero nunca había utilizado Java FX. Por lo que documentarme a fondo sobre sus características, formas de hacerlo funcionar y descubrir todas las ventajas que me podían aportar llevó más de la semana inicial prevista y se me alargo a dos semanas.

- **Creación Login:** Con esta etapa repetí el proceso que hice con la de la comunicación cliente/servidor. Al ver que la implementación básica de la interfaz se alarga más tiempo del estimado y con el fin de no retrasar todo el proyecto en cadena, pasé a implementar esta etapa a la vez que creaba la anterior. La creación del login no me llevó apenas más tiempo del estimado, puesto que requería de una parte de Java FX que ya conocía en ese momento y que no necesitaba de las partes que aún necesitaba practicar y documentarme más.

- **Implementación Lógica:** Una vez terminado el login pasé a empezar a implementar la lógica de la aplicación sin más demora. A estas alturas llevaba un desfase de una semana respecto al planteamiento inicial y pese a que la etapa de la interfaz estaba casi terminada, todavía faltaban algunos flecos. Al estar la interfaz tan avanzada me permitía poder empezar la implementación sin ningún tipo de problema, por lo que cuanto antes empezara con esta etapa mucho mejor. Esta etapa duró unas cinco semanas y aunque fue bastante bien viendo el tiempo final que me llevó, también surgieron algunos problemas como en cualquier implementación de este calibre. Todos estos problemas se resolvieron sobre la marcha y la estimación estuvo bastante cerca de lo estimado.

- **Desarrollo Interfaz:** La etapa anterior terminó en la última semana de abril y comencé a finalizar el desarrollo de la interfaz. De la misma manera que me encontré problemas por el uso de una tecnología completamente nueva en la versión simple, también los encontré ahora. Esto provocó que el desarrollo final de la interfaz se alargara una semana más de lo planificado inicialmente. También influyeron algunos

cambios de estilo que beneficiaban la usabilidad de la aplicación y que metí en esta etapa.

- **Desarrollo Administrador:** Puesto que esta etapa era independiente de todas las anteriores pasé a implementarla paralelamente a la etapa anterior. De esta manera conseguía no demorarla más y que se me pudiera echar el tiempo encima. El desarrollo era simple y claro, por lo que no surgieron problemas con esta herramienta y la estimación fue la correcta o incluso mejor.

- **Test Final:** El test comenzó una vez terminada la aplicación entera. A estas alturas estaba también la herramienta del administrador que se llegó a terminar incluso antes que la interfaz del cliente. El test estaba previsto empezarlo la primera semana de mayo, pero se demoró hasta mediados de mes. Aún así se tuvo alrededor de 5 semanas para poder testear y mejorar la aplicación.

- **Redacción Memoria:** Al mismo tiempo que se realizaba el test de la aplicación comencé a redactar este informe. La etapa de test es la que menos carga de trabajo me daba del proyecto y era el momento idóneo para poder enfocar la memoria. A estas alturas tenía los suficientes conocimientos sobre todo lo que quería hablar y explicar, una visión global de todo lo que estaba montando y del objetivo que tenía este software y ya había podido consultar todas las referencias y artículos que me parecían interesantes y que necesitaba leer.

3 Implementación

En esta sección se pretende explicar todas las tecnologías utilizadas, el motivo por el cuales se escogieron, las funcionalidades exactas que tiene el software y las pruebas que se han realizado para poder comprobar su funcionamiento, siempre acordes a las funcionalidades descritas previamente.

3.1 Tecnologías utilizadas

La primera gran decisión que se tuvo sobre el proyecto fue escoger en qué lenguajes y sobre qué tecnologías se quería montar todo el software. Las opciones eran diversas debido a la gran variedad de lenguajes de programación que he tocado durante la carrera o a la gran cantidad de tecnologías que no había tenido la oportunidad nunca de utilizarlas pese a haber leído sobre ellas y que me parecían un buen momento para poder tener un primer contacto.

3.1.1 Java / RMI

En lo que menos dudé fue en escoger Java para realizar el proyecto. ¿Por qué Java? Porque es el lenguaje de programación con el que me siento más cómodo y el que más me gusta. Me parece un lenguaje de presente y futuro, utilizado en varios sectores y que te permite programar para distintos dispositivos, aunque en esta práctica únicamente nos centramos en una aplicación de escritorio. Pero siempre es un buen momento para poder profundizar más en el lenguaje y en su funcionamiento.

Escogido Java y teniendo claro que había que montar una aplicación distribuida cliente-servidor, el objetivo era buscar una tecnología que me lo permitiese y cuyas características se adaptaran a las exigencias del proyecto. Para este paso debía tener en cuenta algunos aspectos del software que tenía que construir, como el uso que se le quiere dar y el tráfico que puedo permitirme en la red donde estará colocado el servidor. Teniendo en cuenta que este proyecto está preparado para grupos de trabajo, en redes intranet y que el tráfico generado por mi aplicación sería mínimo, quería un mayor grado de abstracción a la hora de la comunicación, por lo que descarté utilizar sockets y creando mi propio RFC donde crear todo un sistema de comunicaciones entre el cliente y el servidor.

Descartados los sockets nos quedamos con RMI (Remote Method Invocation), tecnología con la que ya había tenido la oportunidad de trabajar en algunas asignaturas y que me había parecido bastante interesante de utilizar.

3.1.2 USB Button y JNA

La principal característica del software es la utilización de un pulsador USB para poder manejar tus estados de trabajo y poder ver el estado de los demás compañeros según el color que tenga. También es, a su vez, uno de los puntos más complicados que se han tratado.

Junto al USB Button se incluye un archivo dll (uno para arquitecturas de 32 bits y otro para las de 64 bits) que contiene una serie de funciones ejecutables que se comunican con el pulsador. Pero, ¿cómo utilizar desde Java este fichero dll?

La primera opción que se contempló fue intentar crear un pequeño programa en C++ que fuera capaz de cargar y comunicarse con dicho fichero, y desde Java cargar mediante JNI (Java Native Interface) este programa que haría de intermediario. Se consiguió cargar el programa en C++ y hacerlo funcionar desde Java, pero éste no era capaz de hacer de enlace con el fichero dll, haciendo inviable esta opción.

Con la primera opción descartada surgieron otras, desde cambiar el propio hardware del pulsador hasta intentar cargar el dll directamente desde Java utilizando JNI. Esta última opción también se descartó debido a que una vez conseguí aprender y entender dicho framework de programación, vi que se basaba en incluir en el código C++ una referencia hacia el framework para poder utilizar una nomenclatura del mismo. Se aplica un JNIEXPORT en las funciones que quieres exportar del programa en C++ y utilizar posteriormente en Java cargando dichos archivos comprimidos. Por tanto, esto significaba tener que descomprimir el propio dll, intentar modificarlo como se tiene que hacer en JNI y volverlo a compilar para usarlo en Java, algo que no estaba del todo claro que nos pudiera funcionar. Así que se cambió el enfoque para intentar tener un nuevo punto de vista de cómo poder solucionar este problema, volviendo al problema original. ¿Cómo cargar un DLL desde Java de una manera no invasiva? La solución fue JNA (Java Native Access).

Esta librería da la opción de cargar bibliotecas compartidas sin necesidad de incluir ningún código en la misma. De esta manera podemos desde Java cargar cualquier dll independientemente de la manera que éste haya sido comprimido. Para poder hacerlo funcionar hay que crear una interfaz java que defina todas las funciones y/o estructuras que se desean usar y que están definidas dentro del dll. Las funciones que nosotros necesitamos utilizar son las siguientes:

- **PacInitialize:** Inicializa todos los USBButtons conectados y te devuelve un entero con el número de dispositivos, o 0 si no encuentra ninguno.
- **PacShutDown:** Se encarga de finalizar el USB Button.
- **USBButtonConfigureColor:** Recibe por parámetro la ID del dispositivo asignada en el PacInitialize y un array de bytes que configura el USBButton. En esta función coge el color asignado en el array de bytes y se lo manda al dispositivo para cambiar su color.
- **USBButtonConfigurePermanent:** Recibe por parámetro la ID del dispositivo asignada en el PacInitialize y un array de bytes. Del array coge el color del pulsador, el color y la cadena de teclas que se envian al pulsarlo. De esta manera nosotros podemos configurarlo de manera que muestre un mensaje por pantalla o una cadena de texto cuando lo pulsas.

El dll trae más funciones que no necesitamos utilizar o que no corresponden al dispositivo tratado aquí. Es un dll con funciones compartidas entre todos los dispositivos Pac (Pac-LED64, PacDrive y U-HID LEDs).

Por lo que respecta al array de bytes con el que se configura el dispositivo se puede ver la estructura en la Figura 3.1.2.1.

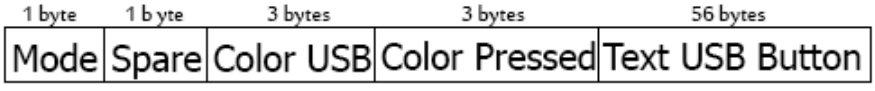


Figura 3.1.2.1 Estructura array de bytes

Observamos que está formado por un total de cinco elementos que se reparten todos los bytes del array.

- **Mode:** Ocupa el primer byte del array e indica el tipo de pulsación que queremos que tenga el pulsador. Si hay que hacer click o doble click (0x00 Alternate, 0x01 Extended, 0x02 Both). En la práctica se implementa la opción Extended.
- **Spare:** Configuración alternativa. Por defecto inicializamos a 0x00.
- **Color USB:** Ocupa tres bytes del array (posición 2, 3 y 4) y contiene el color con el que se muestra el color. Cada byte corresponde a uno de los canales (R,G,B).
- **Color Pressed:** Mismo formato que el color anterior. Contiene el color que toma el USB Button cuando el usuario pincha sobre él.
- **Text USB Button:** Ocupa el resto del array y se puede configurar cualquier tecla del teclado. Ya sea una cadena de texto, una combinación de caracteres, etc...

Puesto que el hardware no trae por defecto ninguna manera de poder detectar su utilización por parte del usuario, la manera en la que podemos averiguar cuando ha sido pulsado es poniendo una combinación de caracteres que estamos seguros que sólo el pulsador será capaz de reproducir. Por tanto, por defecto se configura la parte de "Text USB Button" con una cadena de caracteres para todos los dispositivos que se conectan a nuestro software. La captura de este evento la realizamos con JNativeHook, que definiremos y explicaremos más adelante.

El usuario no debe tener ninguna preocupación por configurar su pulsador, puesto que el propio software se encarga de ello siempre que detecta un USB Button nuevo.

3.1.3 JSON

Una vez escogidas las tecnologías para montar la aplicación cliente/servidor faltaba escoger la forma de hacer la transferencia de datos entre ambas y de poder almacenar datos de relevancia para poder calcular las estadísticas y todo lo relacionado con el software. Las opciones eran muchas e interesantes, por lo que para escoger tenemos que volver a observar las características del proyecto. Recordemos que estamos en un entorno intranet y grupos de trabajo relativamente pequeños, por lo que implementar toda una base de datos era demasiado excesivo para los datos que queremos almacenar. Descartado esto la tecnología por la que optamos es JSON junto a la librería que nos ayuda a trabajar con ellos en Java.

Todos los datos que se guardan se hacen mediante JSONs, al igual que todos los datos que el servidor le envía al cliente. Las estructuras de las puntuaciones siguen todas el mismo patrón, con la pequeña diferencia que puedan tener algún atributo de más. Por ejemplo, para almacenar los puntos de cada día tenemos un formato tal que así:

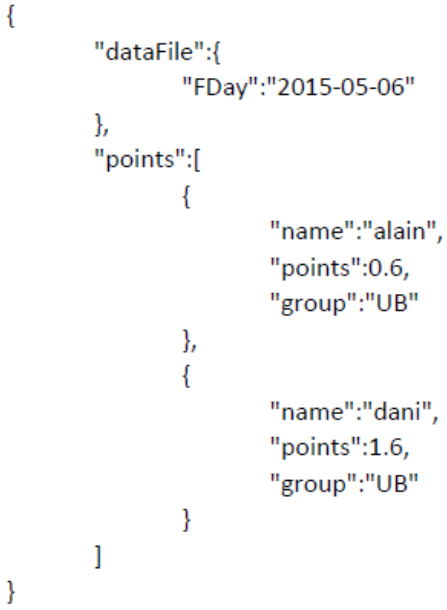


Figura 3.1.2.1 Estructura array de bytes

3.1.4 JNativeHook

Anteriormente hemos hablado de la comprobación que hace el software para saber si un usuario está realizando correctamente el pomodoro, siempre y cuando intentando ser lo menos intrusivos posible en el ordenador de dicho usuario. Para ello lo que se hace es capturar los eventos de teclado y ratón para comprobar que el usuario está trabajando con el ordenador.

Java te permite capturar por si mismo este tipo de eventos, pero sólo para cuando el usuario tiene el foco de la pantalla puesto sobre la aplicación, cosa que no nos sirve para nuestro objetivo. Es por ello que hay que salir a buscar librerías externas que nos permitan seguir capturando estos eventos esté el usuario donde esté, ya sea en un navegador web, en un software cualquiera que tenga que utilizar o moviéndose por cualquier carpeta de windows. De esta manera podemos mantener nuestro software observando estos eventos mientras se ejecuta en segundo plano y sin necesidad de que el usuario tenga que estar pendiente del programa ni que tenga que ir haciendo clicks de confirmación cada X minutos para poder comprobar que sigue trabajando. Nos limitamos únicamente a observar los movimientos que realiza.

Para conseguir esto se ha optado por utilizar la librería JNativeHook. Dicha librería permite crear listeners de Java a nivel global y es capaz de capturar cualquier tipo de evento que pueda salir de un teclado (key press, key release and key typed) y los eventos lanzados por el ratón (mouse down, mouse up, mouse click, mouse wheel, mouse drag, mouse move).

La última release de esta librería es de enero del 2015, por lo que es una librería en constante actualización y mantenimiento.

3.1.5 JavaFX

JavaFX fue lanzado a finales de 2008 por Sun Microsystems (comprada por Oracle Corporation) con la intención de competir contra otras tecnologías punteras del momento como eran Flash o Silverlight. Una de sus principales características es su capacidad para crear interfaces para distintos dispositivos (como aplicaciones de escritorio, móviles, tablets y TV), convirtiéndose así en la primera plataforma que lo consigue y en una opción más que recomendable. Antes de su creación para poder realizar desarrollos entre dispositivos tenías que descargar y aprender a utilizar varias librerías independientes, con JavaFX esto desaparece y usando una única herramienta y un único diseño podemos tener el mismo resultado.

Esta plataforma se presentó como la gran revolucionaria de la época con la intención de acaparar todo el mercado en un futuro. Fue un proyecto ambicioso donde se buscaba monopolizar la creación de interfaces y que cualquier proyecto quisiera terminar usando su tecnología. Desde el lanzamiento de la versión 2.0 se considera una GUI de siguiente generación.

Para conseguir esto no se limitó simplemente a crear una librería de Java que añadiera funcionalidad, si no que se creó su propio código y sus propios scripts compatibles con Java, llamado JavaFX Script. Esto quiere decir que podemos crear una interfaz de usuario utilizando JavaFX y beneficiándonos de todas las ventajas de Java, puesto que su compilación es sobre la propia plataforma. Se quería que este nuevo lenguaje fuera lo más entendible y amigable para los desarrolladores, por lo que su sintaxis es muy parecida a Java. Esto también implica que sea compatible con otras librerías como, por ejemplo, Swing. Nosotros podemos utilizar ambas tecnologías sin ningún tipo de problema ni conflicto y de esta manera enriquecer con todas las ventajas que tienen ambas nuestra interfaz gráfica. Como podemos comprobar en la Figura 3.1.5.1, JavaFX se ejecuta sobre la propia Java Virtual Machine y que dispone de tres versiones de runtime. Uno para dispositivos móviles, otro para escritorio y la tercera es para JavaTV. La propia plataforma se encarga de escoger runtime de manera dinámica detectando el dispositivo.

Pero sus características no solo se reducen a lo mencionado hasta ahora. JavaFX va más allá y te permite crear tus interfaces usando html y/o css, algo que para quienes tengan experiencia en desarrollo web les puede venir muy bien. Es cierto que no es en lenguaje nativo, si no adaptado a la sintaxis de la plataforma, pero está lo suficientemente desarrollado como para que puedas construir cualquier cosa. También puedes poner vídeos, imágenes e incluso utilizar Canvas.

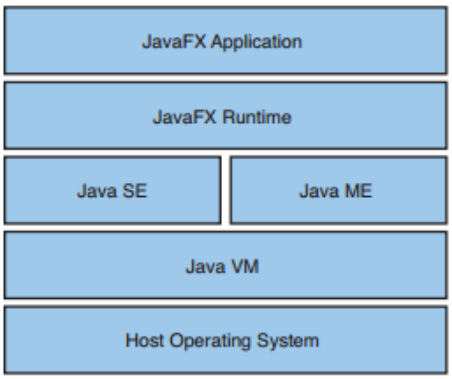


Figura 3.1.5.1 Arquitectura JavaFX

La estructura interna de JavaFX tiene forma de árbol. El punto de partida de cualquier construcción de una interfaz bajo esta plataforma es el contenedor Stage. Normalmente se utiliza un único Stage por aplicación, pero se podría tener más de uno si fuera necesario. Este contenedor está compuesto por una o varias escenas, que serían las pantallas de nuestro software. Si tenemos tres pantallas crearemos tres escenas que iremos intercambiando en el Stage base a medida que el usuario vaya navegando entre ellas. Por tanto, tendremos siempre aplicaciones que se construyen en su totalidad al ejecutarlas y que, posteriormente, lo único que hará es intercambiarlas en el Stage manteniéndolas actualizadas si requiere de datos dinámicos.

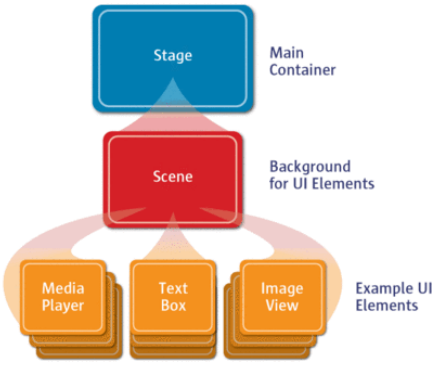


Figura 3.1.5.2 Estructura JavaFX

Una definición más exacta de los distintos estados es el siguiente:

- **Stage:** Es el objeto superior de la aplicación. Actúa como contenedor de la aplicación. Se pueden modificar sus propiedades, como es el icono de la aplicación, el título y la visibilidad de sus elementos.
- **Scene:** Contenedor que aloja la interfaz de usuario y todas sus propiedades (tamaño, características del cursor, estilos, contenido, etc...)
- **Node:** Aquí englobamos el resto del contenido como bloques, independientemente del elemento que sea (items, vídeos, textbox, labels...). Un conjunto de nodes se denomina Group.

3.2 Funcionalidad

Hasta ahora hemos entrado a valorar todo el funcionamiento del software a un nivel teórico. En este apartado se pretende entrar a explicar de manera más detallada y funcional varios aspectos importantes de la aplicación. Este apartado debe servir como una guía para entender todo lo que nos podemos encontrar al utilizar nuestro software.

3.2.1 Bandeja de Windows

En primer lugar pasaremos a explicar todo lo que nos podemos encontrar en la bandeja de windows. Como ya hemos comentado anteriormente, nuestra aplicación pretende ser lo menos intrusiva posible en el ordenador del usuario, pero debemos tener en cuenta de que no todo el mundo tiene por qué disponer del USB Button con el que se ha montado

la aplicación. Es por ello que se creó un equivalente al botón en la propia interfaz del programa y que hemos puesto una serie de ayudas en la bandeja de windows para que el usuario sepa en todo momento el estado en el que se encuentra. Estas ayudas se resumen en dos tipos distintos: texto y audio. Ambos son configurables (se pueden activar y desactivar) desde la pantalla Settings de la aplicación, pero recomendamos su uso.

La ayuda mediante audio consiste en un sonido del sistema cada vez que ocurre algún evento importante en el programa. Normalmente estos eventos se refieren a algún cambio de estado, ya sea porque debes confirmar el pomodoro o está a punto de ser cancelado por inactividad.

En lo que respecta a las ayudas textuales existen dos tipos. La primera la podemos ver en la Figura 3.2.1.1, donde vemos el resultado de poner el ratón encima del icono de pomodoro. Un pequeño texto aparecerá indicando el estado en el que estamos y el tiempo restante para terminar.

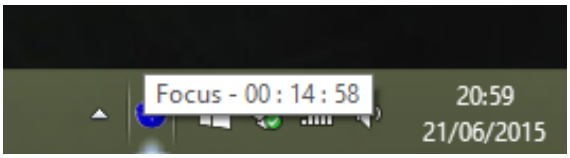


Figura 3.2.1.1 Descripción del estado

El segundo tipo son una serie de notificaciones que nos aparecen sobre el icono y que nos informan de los cambios de estados, tal y como podemos ver en la Figura 3.2.1.2. En este ejemplo tenemos el mensaje que aparece cuando un usuario está inactivo y está a punto de cancelarse el pomodoro.

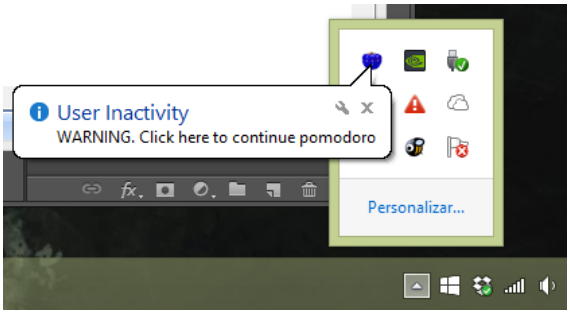


Figura 3.2.1.2 Mensaje de alerta informativo

Hay diversos mensajes de este tipo para los distintos eventos que nos podemos encontrar. La definición de los eventos y mensajes los encontramos en la tabla 3.2.1.3.

Event/Title	Body
Confirm Pomodoro	Pomodoro finished. Click to confirm!
Ready Pomodoro	Ready for new Pomodoro
End Day	Congratulations. You have completed all Pomodoros
Warning Time	WARNING. Click here to continue pomodoro
Cancel pomodoro	Pomodoro has been canceled by user inactivity

Tabla 3.2.1.3. Resumen de las noticiaciones de la bandeja de windows

3.2.2 Archivo de Configuración

La aplicación requiere de un archivo de configuración que debe existir dentro de la carpeta data. En este archivo carga y guarda algunos datos que son de utilidad para mejorar la usabilidad y experiencia del usuario con la aplicación. Cuando nos conectemos por primera vez veremos que los campos de Settings donde indicamos la ip, puerto y usuario de conexión están vacíos. Una vez los rellenemos y nos conectemos con éxito, dichos datos la aplicación los guardará en su fichero de configuración para que la próxima vez que inicies la aplicación haga la conexión automáticamente sin necesidad de que tengas que volver a poner los datos. En caso de no querer conectarte al mismo servidor, únicamente tendrás que ir a la pantalla Settings, desconectarte y poner los nuevos datos. Nuestra aplicación siempre guardará los datos de tu última conexión realizada con éxito para realizarla automáticamente cada vez que inicies el software.

Esta misma operación se realiza con el resto de opciones configurables en la pantalla Settings y en la pantalla principal de la aplicación, donde el software guarda la configuración de tu último pomodoro intentado. Con esto conseguimos que un usuario que siempre utilice la misma medida de tiempo para su Focus Time no tenga que estar configurando siempre el tiempo, y sea la propia aplicación quien sea capaz de recordarlo y ahorrarle ese trabajo al usuario. Podemos ver un ejemplo del fichero de configuración en la Figura 3.2.2.1


```
{
    "configPomodoro":{
        "sliderValue":15
    },
    "serverConnection":{
        "hostname":"localhost",
        "port":"8080"
    },
    "connection":{
        "password":"probando123",
        "name":"alain",
        "group":"UB"
    },
    "configMessages":{
        "msgTray":true,
        "sound":true
    },
    "status":{
        "finishStatus":"2015-06-21 21:18:01",
        "status":1
    }
}
```


Figura 3.2.2.1 Ejemplo JSON del fichero de configuración

3.2.3 Pantalla de Stats

Tal y como comentamos en la explicación de JavaFX, lo que realizamos en el software es crear todas las pantallas (escenas) al inicio y luego únicamente mantenerlas actualizadas y cambiar la que se encuentra en el Stage a medida que el usuario va navegando entre ellas. Esto implica que debes tener un seguimiento de todas en todo momento para evitar que el usuario cuando acceda a ella se encuentre con información mal actualizada. Para evitar el exceso de tráfico en la red haciendo peticiones para actualizar las estadísticas, lo que hacemos es pausar dichas peticiones cuando el usuario no está en la pestaña de Stats y está navegando por otras partes de la aplicación. En cambio, cuando el usuario

vuelve a acceder a ella lo que hacemos es una petición previa a mostrar la escena para poder actualizar los datos y seguir con el flujo normal de la página, que hace peticiones al servidor cada cinco segundos para tener los datos actualizados casi a tiempo real.

En caso de que no estemos conectados a ningún servidor tampoco se realizan las peticiones y las tablas y gráficas saldrán vacías.

3.2.4 Pomodoros

Ya hemos hablado de lo que consideramos un pomodoro correcto o incorrecto y de los tipos de estados que puede tener un usuario durante su ciclo de vida, pero no hemos mencionado algunos detalles a nivel funcional que es de vital importancia conocerlos.

Hay dos momentos clave en el que es importante interactuar con el programa para poder continuar con su funcionamiento normal. El primero de ellos es cuando termina un pomodoro y debe ser confirmado. Esto se requiere para comprobar que el usuario sigue delante del ordenador trabajando y no se ha ido antes de tiempo. Por tanto, cuando termina el tiempo comienza el estado "Push Pomodoro" que dará hasta un máximo de 30 segundos de margen para que el usuario pulse el USB Button o haga click sobre el pulsador de la interfaz. No sirve pulsar cualquier tecla, únicamente se puede pulsar en los objetos mencionados para poder confirmarlo. El otro momento importante es cuando comienza el aviso por inactividad. Este estado (Warning Time) salta cuando estamos dos minutos sin que nuestro software detecte uso en el ordenador. Dicho estado estará activo únicamente durante 10 segundos y si sigue la inactividad cancelará el Pomodoro pasando al estado Ready. En cambio, para poder cancelar este tiempo Warning y que continúe el pomodoro sin ningún problema se puede pulsar cualquier tecla del teclado o cualquier botón del ratón. Podemos ver un resumen de estos dos estados en la tabla 3.2.4.1.

Estado	Explicación
Warning Time	Dura un máximo de 10 segundos. Aparece tras dos minutos de inactividad.
Push Pomodoro	Dura un máximo de 30 segundos. Se inicia al terminar la cuenta atrás de un pomodoro

Tabla 3.2.4.1. Resumen del Warning Time y del Push Pomodoro

Para aumentar la visibilidad del tiempo que lleva una persona trabajando y darle un feedback más exacto, también incluimos una pequeña seña visual que pueda ser fácilmente reconocible por el usuario y sin necesidad de mirar nada más sepa en qué punto exacto está del pomodoro. Nos referimos a un pequeño parpadeo en el USB Button (y, por tanto, también lo podríamos ver en la réplica de la interfaz) que se da en momentos puntuales. Tal y como resumimos en la tabla 3.2.4.2, cuando completamos el 25% de un pomodoro mostramos un pequeño parpadeo sobre el USB Button. Esto le indica al usuario que ya ha cumplido el primer cuarto del tiempo configurado. Cuando se completa el segundo cuarto del tiempo (es decir, el 50%) realizamos la misma operación pero realizando dos parpadeos. Por cada 25% del tiempo que pasa sumamos un parpadeo más, así con ver el número de parpadeos que ha realizado el pulsador puede saber si se encuentra al principio del tiempo, a mitad o ya en la recta final del tiempo que ha configurado para el pomodoro.

Situacion	Número de Parpadeos
25% completado	1 parpadeo
50% completado	2 parpadeos
75% completado	3 parpadeos

Tabla 3.2.4.2. Parpadeos durante los pomodoros

Por último cabe añadir que una vez el pomodoro ha terminado y el usuario lo ha confirmado es cuando se da por válido y se hace la suma de sus puntos. No se realiza cuando termina el relax time ni en ningún otro caso, consideramos que el pomodoro está bien realizado una vez ha terminado el periodo de Focus Time.

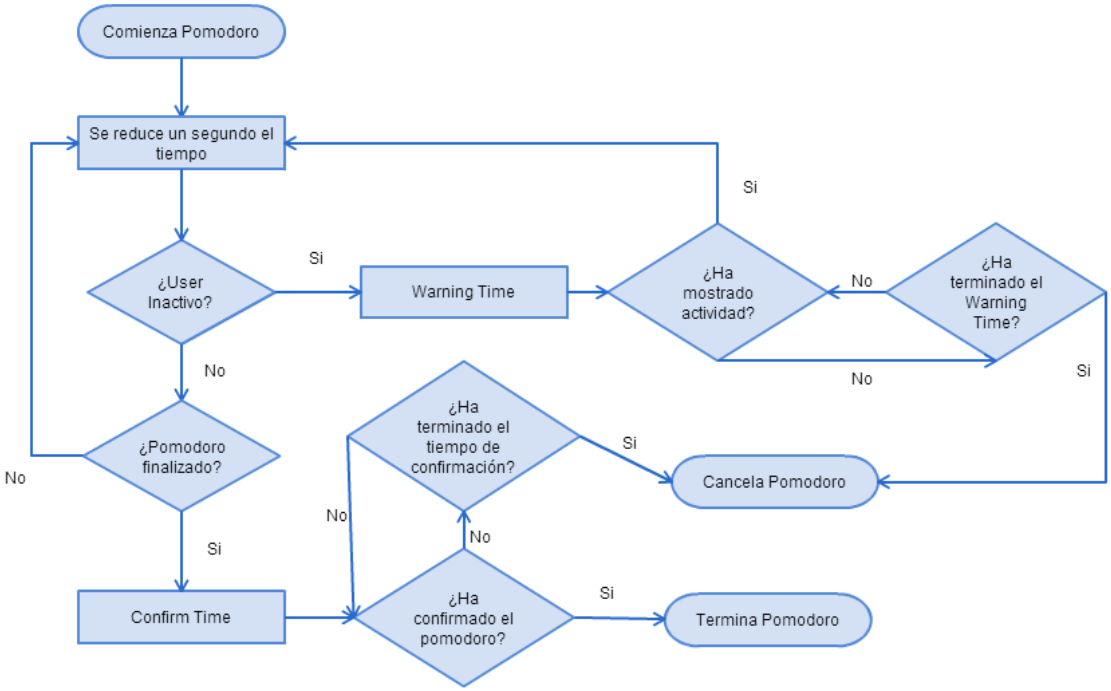


Figura 3.2.4.3 Diagrama de flujo de un pomodoro

3.2.5 Administrador

Como todo software, también tenemos una herramienta de administración creada. Desde ella se puede hacer algunas configuraciones sobre el servidor que se gestiona. La idea de esta herramienta es que sólo pueda tener acceso aquella persona responsable del mismo. Sus funciones son las de generar todos los usuarios de sus grupos, asignarles una contraseña y un nombre de usuario, además del grupo o grupos a los que puede pertenecer. Los usuarios en ningún momento pueden cambiar estos datos, es única y exclusivamente función del administrador del sistema.

Con esta herramienta el administrador será capaz de hacer lo siguiente:

- Crear, modificar y borrar usuarios del servidor.
- Consultar todas las estadísticas disponibles en el servidor, independientemente del tiempo que lleven.

Esta herramienta está diseñada para hacer las funciones más básicas y necesarias sobre el servidor. Con esto ya se puede tener un control total sobre la parte más importante, que es la gestión de los usuarios y de los empleados que pueden tener acceso al servidor mediante sus clientes.

3.3 Casos de Uso

En este apartado vamos a hablar de los casos de uso que nos podemos encontrar en la aplicación a nivel de roles. Vamos a empezar a mencionar los más significativos en clave usuario y, posteriormente, pasaremos a la aplicación del administrador para analizar los suyos.

3.3.1 Usuario

Cualquier usuario que use la aplicación tiene dos funcionalidades bien diferenciadas. La primera guarda relación con toda la lógica de los pomodoros. Realizar un pomodoro implica tener que hacer una configuración previa, iniciarlo y confirmarlo una vez haya terminado. Mientras que configurar e iniciar el pomodoro no requiere de una conexión con el servidor, terminarlo sí que lo requerirá salvo que queramos perder el tiempo realizado. También podemos estar logueados al iniciar el pomodoro y durante el mismo cambiar de servidor, en ese caso el pomodoro se registrará en el servidor en el que estés conectado al finalizar.

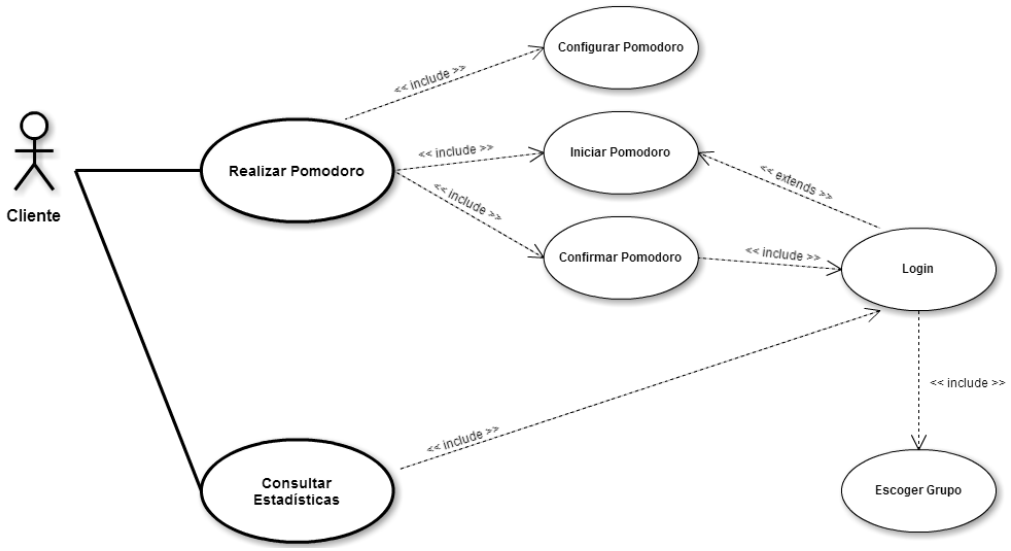


Figura 3.3.1.1 Casos de uso del usuario

La segunda funcionalidad consiste en consultar las estadísticas personales o las grupales. Éstas estadísticas son siempre relacionadas a un servidor, por lo que siempre necesitaremos estar logueados para consultarlas. Las individuales nos muestran nuestras puntuaciones en el servidor en cuestión, mientras que las grupales nos muestran las puntuaciones que tenemos en el grupo asignado en dicho servidor.

3.3.2 Administrador

Con la herramienta administrativa del servidor se pueden realizar gestiones sobre los usuarios o consultar estadísticas completas filtradas por día, semana o meses. Todas estas funcionalidades requieren de un login en el servidor. Si el usuario quiere modificar o borrar un usuario incluye la tarea de localizarlo en primer lugar. Esto es debido a que la edición se hace sobre la misma tabla y necesitas ir a la fila en cuestión para poder seleccionarla y hacer la acción correspondiente. Para añadir un usuario no hace falta que busquemos en la tabla, simplemente clickar en la opción y guardar los cambios.

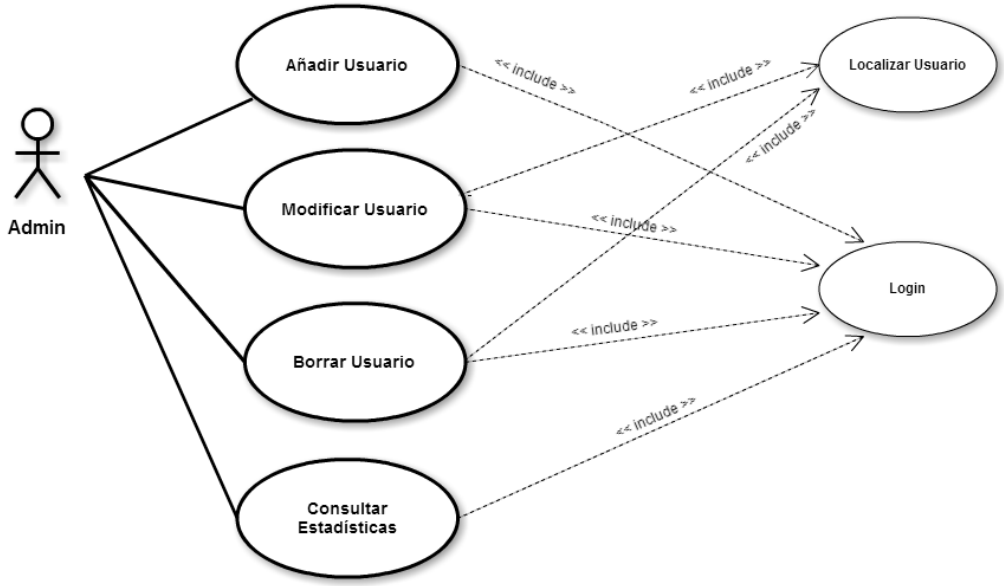


Figura 3.3.2.1 Casos de uso del Administrador

En cuanto a consultar las estadísticas el único proceso que interviene es el login en el servidor.

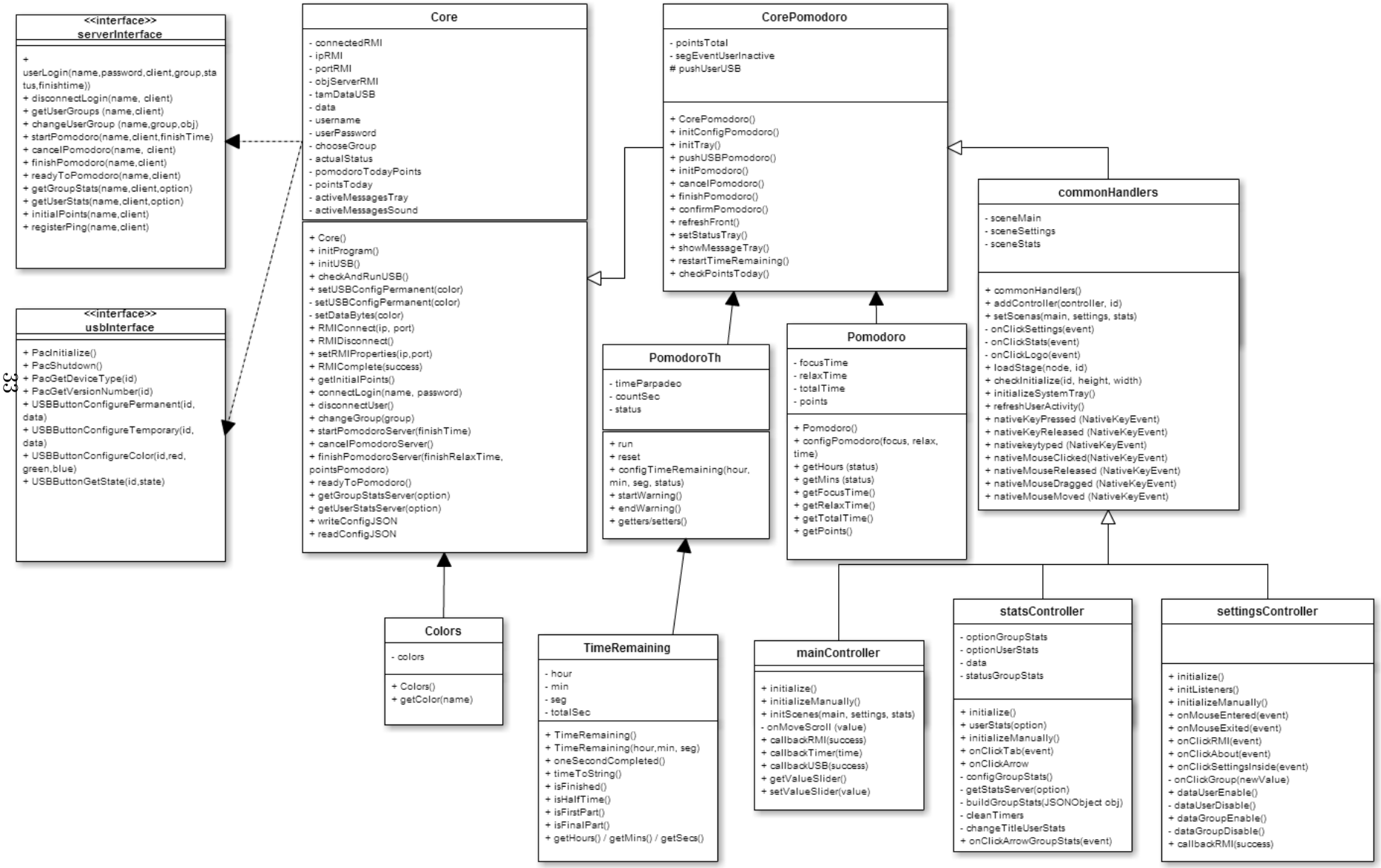
3.4 Arquitectura

La arquitectura planteada para el desarrollo del software está basado en el patrón MVC (Modelo-Vista-Controlador). Se ha intentado separar en todo momento la gestión de la información con la cual opera el software, de los eventos que invoca el usuario con sus actos y la presentación de dicha información.

Los ficheros de JavaFX (.fxml) contienen todos los elementos relacionados con las vistas. Cada vista tiene asociado un controlador Java que gestiona todos los eventos relacionados, ya sea por interacción del usuario o por eventos internos que pidan actualizar la interfaz. Por otra parte, estos controladores heredan de una API que hace toda la gestión interna del software. Gestiona los pomodoros, lanza los eventos para actualizar la interfaz, hace las peticiones al servidor para que devuelva los datos necesarios, etc. Esta API está compuesta por varias clases con una estructura de herencia y cada una tiene una función específica que explicaremos más adelante. También hace la gestión del botón USB, facilitando una serie de funciones para poder cambiar el color del mismo sin necesidad de saber cómo funciona el pulsador. En el análisis inicial del proyecto se pensaron varias formas de organizar la arquitectura, pero al final se decidió por crear esta especie de API al que todos los controladores se tuvieran que "suscribir" para poder usar todas sus funciones sin necesidad de saber su estructura interna ni su funcionamiento. Realizar una especie de caja negra donde lo único que debe interesar al desarrollador es la función a la que debe llamar, los parámetros que le debe pasar y el resultado que dicha función le devolverá. A partir de ese resultado él trabajará en lo que necesite realizar.

Con esta estructura no solo se persigue el patrón mencionado, si no realizar un código escalable y reutilizable. Cualquier nueva vista que se tuviera que crear para la aplicación únicamente tendría que heredar de la API mencionada y usarla como capa de abstracción respecto a su funcionalidad. Únicamente necesita saber las funciones a las que debe llamar para poder hacer la petición y gestión de datos.

En la siguiente página podemos echar un primer vistazo general del diagrama de clases.


<<interface>>
serverInterface
+
userLogin(name,password,client,group,status,finishtime))
+ disconnectLogin(name, client)
+ getUserGroups (name,client)
+ changeUserGroup (name,group,obj)
+ startPomodoro(name,client,finishTime)
+ cancelPomodoro(name, client)
+ finishPomodoro(name,client)
+ readyToPomodoro(name,client)
+ getGroupStats(name,client,option)
+ getUserStats(name,client,option)
+ initialPoints(name,client)
+ registerPing(name,client)
<<interface>>
usbInterface
+ PacInitialize()
+ PacShutdown()
+ PacGetDeviceType(id)
+ PacGetVersionNumber(id)
+ USBButtonConfigurePermanent(id, data)
+ USBButtonConfigureTemporary(id, data)
+ USBButtonConfigureColor(id,red, green,blue)
+ USBButtonGetState(id,state)
Core
- connectedRMI
- ipRMI
- portRMI
- objServerRMI
- tamDataUSB
- data
- username
- userPassword
- chooseGroup
- actualStatus
- pomodoroTodayPoints
- pointsToday
- activeMessagesTray
- activeMessagesSound
+ Core()
+ initProgram()
+ initUSB()
+ checkAndRunUSB()
+ setUSBConfigPermanent(color)
- setUSBConfigPermanent(color)
- setDataBytes(color)
+ RMIConnect(ip, port)
+ RMIDisconnect()
+ setRMIProperties(ip,port)
+ RMIComplete(success)
+ getInitialPoints()
+ connectLogin(name, password)
+ disconnectUser()
+ changeGroup(group)
+ startPomodoroServer(finishTime)
+ cancelPomodoroServer()
+ finishPomodoroServer(finishRelaxTime, pointsPomodoro)
+ readyToPomodoro()
+ getGroupStatsServer(option)
+ getUserStatsServer(option)
+ writeConfigJSON
+ readConfigJSON
CorePomodoro
- pointsTotal
- segEventUserInactive
pushUserUSB
+ CorePomodoro()
+ initConfigPomodoro()
+ initTray()
+ pushUSBPomodoro()
+ initPomodoro()
+ cancelPomodoro()
+ finishPomodoro()
+ confirmPomodoro()
+ refreshFront()
+ setStatusTray()
+ showMessageTray()
+ restartTimeRemaining()
+ checkPointsToday()
commonHandlers
- sceneMain
- sceneSettings
- sceneStats
+ commonHandlers()
+ addController(controller, id)
+ setScenas(main, settings, stats)
- onClickSettings(event)
- onClickStats(event)
- onClickLogo(event)
+ loadStage(node, id)
+ checkInitialize(id, height, width)
+ initializeSystemTray()
+ refreshUserActivity()
+ nativeKeyPressed (NativeKeyEvent)
+ nativeKeyReleased (NativeKeyEvent)
+ nativekeytyped (NativeKeyEvent)
+ nativeMouseClicked(NativeKeyEvent)
+ nativeMouseReleased (NativeKeyEvent)
+ nativeMouseDragged (NativeKeyEvent)
+ nativeMouseMoved (NativeKeyEvent)
PomodoroTh
- timeParpadeo
- countSec
- status
+ run
+ reset
+ configTimeRemaining(hour, min, seg, status)
+ startWarning()
+ endWarning()
+ getters/setters()
Pomodoro
- focusTime
- relaxTime
- totalTime
- points
+ Pomodoro()
+ configPomodoro(focus, relax, time)
+ getHours (status)
+ getMins (status)
+ getFocusTime()
+ getRelaxTime()
+ getTotalTime()
+ getPoints()
Colors
- colors
+ Colors()
+ getColor(name)
TimeRemaining
- hour
- min
- seg
- totalSec
+ TimeRemaining()
+ TimeRemaining(hour,min, seg)
+ oneSecondCompleted()
+ timeToString()
+ isFinished()
+ isHalfTime()
+ isFirstPart()
+ isFinalPart()
+ getHours() / getMins() / getSecs()
mainController
+ initialize()
+ initializeManually()
+ initScenes(main, settings, stats)
- onMoveScroll (value)
+ callbackRMI(success)
+ callbackTimer(time)
+ callbackUSB(success)
+ getValueSlider()
+ setValueSlider(value)
statsController
- optionGroupStats
- optionUserStats
- data
- statusGroupStats
+ initialize()
+ userStats(option)
+ initializeManually()
+ onClickTab(event)
+ onClickArrow
- configGroupStats()
- getStatsServer(option)
- buildGroupStats(JSONObject obj)
- cleanTimers
- changeTitleUserStats
+ onClickArrowGroupStats(event)
settingsController
+ initialize()
+ initListeners()
+ initializeManually()
+ onMouseEntered(event)
+ onMouseExited(event)
+ onClickRMI(event)
+ onClickAbout(event)
+ onClickSettingsInside(event)
- onClickGroup(newValue)
+ dataUserEnable()
- dataUserDisable()
+ dataGroupEnable()
- dataGroupDisable()
+ callbackRMI(success)

3.4.1 Core

Comenzamos el análisis por la clase principal y la superior de nuestra arquitectura: el Core.

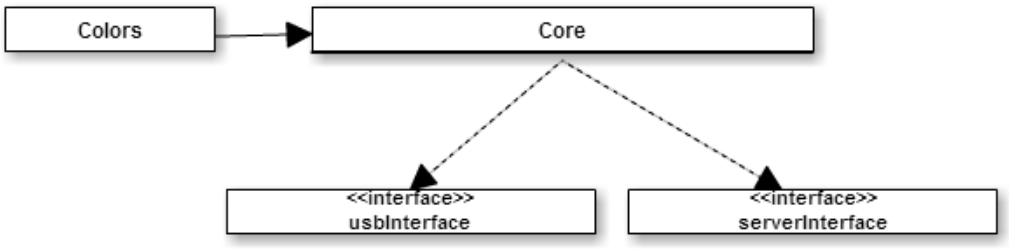


Figura 3.4.1.1 Diagrama Core

La funcionalidad principal de esta clase es encargarse de cualquier comunicación o evento externo al software. Aquí incluimos cualquier llamada al servidor, la gestión del pulsador USB y las llamadas a su dll, y tanto la carga como escritura de ficheros de los ficheros de configuración necesarios. Por tanto, cualquier interfaz estará siendo utiliza por el Core (en este caso tanto usbInterface que, como ya hemos visto, se utiliza para comunicarnos con el pulsador USB, como la serverInterface que se encarga de comunicarnos con el servidor). De esta manera desde cualquier parte de nuestra aplicación podemos acceder a modificar el pulsador o a realizar llamadas a nuestro servidor.

Por otra parte, también contiene una instancia a una clase de colores personalizados donde se define para cada color unos valores R,G,B necesarios para incluirlos en el array (como ya explicamos en el apartado del USB Button) del pulsador. Dichos valores suelen ir según el color de su canal entre 0 y 255, pero recordemos que en Java el tipo byte va desde -128 hasta 127. Para no tener que estar repitiendo en todo momento los valores exactos de cada color únicamente debemos usar la función que nos facilita la clase Color donde mediante un string le diremos el color que queremos y nos devolverá los valores correctos y que necesitamos para mandarlo al hardware.

Debido a que también se encarga de realizar la escritura y carga de la configuración del software, es necesario que algunos estados globales que son necesarios en toda la aplicación se tengan que declarar aquí y deban ser accesibles en todo momento por los métodos en cuestión. Aunque son las clases inferiores quienes los actualizan, el Core debe ser capaz de acceder a ellos para poder realizar algunas funcionalidades ya explicadas como la memoria de los settings del usuario.

3.4.2 CorePomodoro

Del Core hereda el CorePomodoro. Esta clase se encarga de toda la gestión interna de los pomodoros, tanto de la estructura de lo que sería un Pomodoro hasta su inicialización, sus estados y toda la lógica interna. Lo primero que realiza es asignar los colores a los estados, para así poder tenerlos declarados en un mismo sitio y acceder siempre a estos datos cuando cambiamos de estado. También nos simplifica a la hora de cambiar el nombre de los estados o si algún día se replantea el color de los mismos, puesto que sólo tendríamos que ir a esta inicialización, cambiar el dato que queramos y automáticamente se cambiaría en todo el software sin mayor esfuerzo.

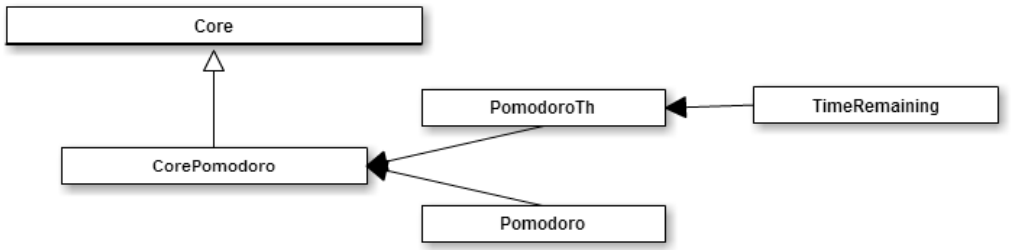


Figura 3.4.2.1 Diagrama CorePomodoro

Otra de sus funcionalidades es la de gestionar los eventos que lanzan los cambios de estado del software o de un Pomodoro. Esto significa gestionar las alertas que se envían al icono situado en la bandeja de windows, al igual que sus sonidos.

También debe realizar la gestión de los threads relacionados al pomodoro. Recordemos que usar threads nos permite que el usuario pueda navegar y utilizar la aplicación mientras está realizando un pomodoro o cualquier otra acción. Por tanto el CorePomodoro gestiona la clase PomodoroTh que es la encargada de llevar la cuenta atrás del tiempo restante, de las notificaciones por haber completado 1/4, 1/2 o 3/4 partes del pomodoro en activo y de gestionar los cambios de estado. Con cambios de estado nos referimos a ir comprobando si debe empezar el tiempo de aviso al usuario por inactividad, o el tiempo de confirmación porque el pomodoro ha terminado o si debe pasar a ready. Esta clase únicamente avisa del cambio de estado que se debe realizar, pero la gestión del evento la realiza el CorePomodoro. Para la cuenta atrás que realiza el software se ha creado una clase personalizada llamada TimeRemaining. Dicha clase tiene una función llamada oneSecondCompleted que resta un segundo al tiempo cada vez que es llamada. Por tanto, podemos deducir que el funcionamiento lógico de un pomodoro realmente es un timer que en cada segundo va llamando a esta función para que reste un segundo al tiempo.

Por último, tenemos la clase Pomodoro que define la estructura que debe tener. Si desglosamos un pomodoro nos salen distintos atributos, como son el Focus Time, Relax Time, Total Time (suma de los dos tiempos anteriores) y points (puntos del pomodoro en caso de completarse).

3.4.3 CommonHandlers

El objetivo principal del commonHandlers es la de controlar todos los eventos comunes de la aplicación y la gestión de cambios de escena de JavaFX. Hereda directamente del CorePomodoro y según la descripción anterior podemos asignarle dos subtareas dentro de su gestión de eventos globales:

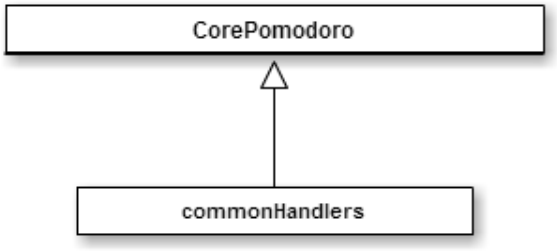


Figura 3.4.3.1 Diagrama commonHandlers

- **Eventos del ordenador:** Con estos eventos nos referimos a implementar la librería JNativeHook vista y explicada en anteriores apartados. Es en esta clase donde gestionamos e implementamos los eventos que nos permiten controlar si el usuario está utilizando el ordenador y, en consecuencia, si está trabajando durante un pomodoro. Estos eventos también nos permiten averiguar cuando el usuario ha hecho uso del pulsador. Recordemos que la única manera real de poder saber si el usuario ha pulsado nuestro USB Button es mediante una cadena de caracteres especial que se le ha configurado. Por lo que en los eventos de teclado está la lógica que va analizando todos los datos de entrada buscando la cadena de caracteres del USB Button y avisar cuando se pulse.

- **Cambios de Escena:** Cada vez que el usuario interactúa con el menú superior de la aplicación y quiere cambiar de pantalla el controlador en cuestión pide un cambio de escena. Ese cambio de escena lo gestiona también esta clase, siendo capaz de reconocer la escena cargada actualmente por el usuario y a cuál quiere ir. Con estos datos y la librería de JavaFX es capaz de realizar dicha acción e invocar un método del controller de la escena a cargar que permita actualizar los datos a mostrar en caso de ser necesario. Esto nos sirve para, por ejemplo, la pantalla de Stats, que para ahorrar tráfico en la red únicamente se actualiza cuando el usuario la está viendo. Cuando cambia de pantalla, la aplicación deja de pedir al servidor que actualice los datos, pero para evitar que el usuario al volver a entrar los vea desactualizados se hace una carga previa a la visualización de dicha pantalla. El método que realiza esta carga es al que invoca el commonHandlers y que podemos localizar en el diagrama de clases como initializeManually().

3.4.4 Controllers

Con la explicación del commonHandlers hemos terminado de explicar lo que denominamos la API o la Caja Negra del software. Todos los controladores creados para gestionar su vista deben heredar de dicha clase para poder utilizar todas las funcionalidades de la aplicación, tal y como podemos comprobar en la figura 3.4.4.1.

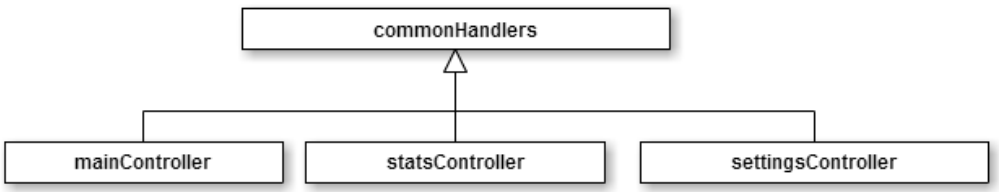


Figura 3.4.4.1 Diagrama controladores

Todos los controladores tienen como atributos las id's de los elementos que necesitan de la interfaz, ya sea para actualizar sus datos o para leerlos. Si no se declaran en el controlador mediante su id no podremos manejarlos. También tienen todas las funciones abstractas que la API no implementa y que es necesario que lo hagan los controladores. Es una manera de poder realizar callbacks para actualizar cosas de la interfaz. Por ejemplo, cuando se realiza la conexión por RMI y se completa correctamente, desde el Core se lanza la función abstracta callbackRMI() pasándole un "true" para identificar el estado de la conexión. De manera interna el Runtime de Java sabe a qué controlador debe ir

para ejecutar esta función y realizar la lógica que haya en ella. En este caso en concreto dicha lógica consistiría en actualizar la interfaz. Si estamos en el settings sería cambiar el estado del botón de "Conectar" y sus labels asociados, o en caso de estar en la página principal su funcionalidad sería pasar a verde el estado del server. Por este motivo es importante que los controladores implementen las funciones abstractas que necesita la API para informar de algunos eventos que ocurren durante la ejecución de la aplicación.

Como hemos comentado anteriormente, estos controladores también tienen todos los eventos que pueden surgir por la interacción del usuario con la interfaz. Todos estos eventos comienza con el prefijo "on" acompañado de la acción y una breve descripción. Es importante seguir una misma semántica para así poder identificar de un primer vistazo para qué sirve cada función y lo que hace. Estos eventos se encargan de hacer las llamadas pertinentes a la API para realizar la lógica que el usuario requiere mediante su interacción.

3.4.5 Timers

Todos los timers necesarios para desarrollar todas las funcionalidades del software heredan de la clase TimerTask. Esta clase es abstracta, por lo que todas las clases que la utilicen deben implementar el método run, encargado de iniciar la ejecución del thread y donde estará la lógica que necesitamos realizar. Esta clase se utiliza junto a la clase Timer (java.util.Timer), que nos permite parametrizar algunas opciones interesantes, como la hora a la que queremos que se inicie el thread y cada cuanto tiempo queremos ejecutarlo.

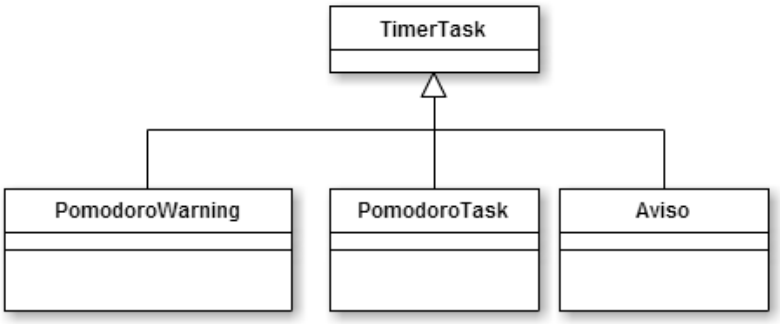


Figura 3.4.5.1 Diagrama Timers

3.5 Pruebas Realizadas

La etapa de testing ha sido bastante extensa en el proyecto y por suerte hemos podido contar con la ayuda de más gente para poder realizarla. Las pruebas realizadas las podemos dividir en dos etapas.

La primera de ellas se produjo durante la **etapa de desarrollo**. En esta etapa el test que se realizó fue exhaustivo por parte de J.Daniel Prades y por mi parte. Durante el desarrollo de cada sprint yo iba probando las nuevas funcionalidades implementadas en mi máquina local y antes de la entrega al finalizar el sprint realizaba un test en un servidor facilitado por mi profesor. Durante este mismo tiempo Daniel iba testeando la versión que le había entregado al finalizar el anterior sprint para ir sacando posibles bugs, nuevas mejoras y nuevas ideas que podíamos analizar para incluir en el desarrollo.

Pese al test individual que nosotros le podíamos hacer a la aplicación, el objetivo de la misma es su funcionamiento en un entorno de trabajo, por lo que era más que recomenda-

ble buscar un sitio donde poder implementar este software y someterlo a un entorno real de trabajo para comprobar el cumplimiento de los objetivos que nos impusimos. De poco nos servía crear una aplicación que funcionara con dos usuarios, pero que luego cuando añadimos más gente al entorno empieza a fallar o resulta que hemos creado un software que no es del gusto de los empleados. Por ello es importante realizar una fase de testing en un entorno lo más parecido posible al que está orientado para poder analizar los resultados, las opiniones de los empleados y tener un feedback mucho más eficaz y completo. Esto se consiguió gracias al grupo de trabajo de Daniel que con mucho gusto accedieron a utilizar el software durante sus jornadas de trabajo para que pudiésemos analizar los resultados, tener más gente que nos pueda hacer un feedback sobre la aplicación y todo sea dicho de paso, les pueda servir para poder utilizar una nueva metodología de trabajo y comprobar si les es de utilidad.

Este test es la que consideramos como la segunda etapa y que se realizó **una vez el desarrollo se dio por finalizado.** Esta etapa continúa a día de hoy y se espera que siga así como mínimo hasta el día de la entrega de este proyecto.

4 Manual de Usuario

4.1 Requisitos e Instalación

Los requisitos mínimos para los que se ha diseñado el software son los siguientes:

- **Sistema Operativo:** Se recomienda tener Windows 7 o superior instalado.
- **Java:** Última versión de Java para hacer funcionar el cliente y el administrador. Para el servidor requiere una versión igual o superior a la 1.6.
- **Eclipse:** Entorno de desarrollo con el que se ha creado la aplicación. Sólo es necesario en caso de querer cargar el código.
- **Bandeja de iconos:** Un Sistema Operativo que permita ejecutar el software poniendo un icono en el tray.

Junto al proyecto se ha incluido todo lo necesario para poder ejecutar el software en distintas carpetas. NO se incluye el USB Button. Los pasos recomendados a seguir para poder ejecutar correctamente el software es el siguiente:

- **Paso 1:** En primer lugar debemos levantar nuestro servidor. Para ello ir a la carpeta "Server" y ejecutar el archivo .bat. Se abrirá una consola que nos mostrará nuestra ip, servername, puerto en el que está escuchando (siempre será el 8080) y si ha conseguido cargar estadísticas. Que no encuentre alguna no significa que se haya ejecutado mal, simplemente que no ha encontrado estadísticas para ese periodo de tiempo. Se adjuntan ficheros con estadísticas hasta el mismo día de la entrega.

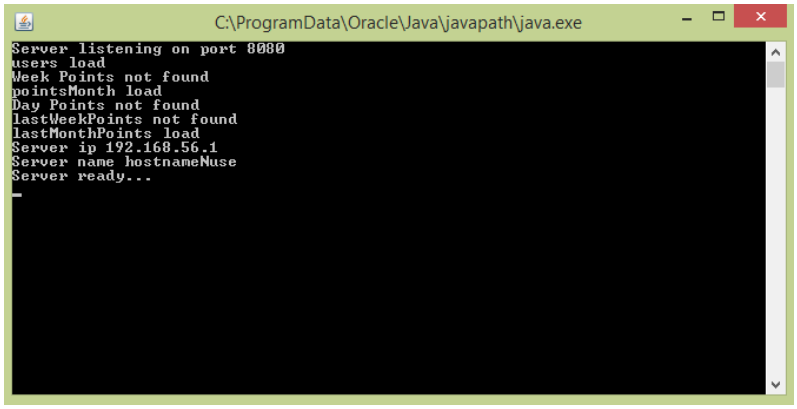


Figura 4.1.1 Ejemplo de ejecución del servidor en localhost

- **Paso 2:** Ir a la carpeta del cliente y ejecutar el .bat disponible. Se iniciará la aplicación en la bandeja de windows y si hacemos doble click sobre el icono veremos la interfaz.
- **Paso 3:** Con la interfaz abierta nos vamos a Settings y comprobamos los datos de conexión. Si es nuestra primera conexión aparecerán vacíos, por lo que si hemos realizado bien el primer paso los rellenaremos de la siguiente manera:
  - **Server IP:** localhost

- **Server Port:** 8080
- **Username:** alain
- **Password:** probando123

- **Paso 4:** *(Paso opcional solo en caso de querer probar el administrador)* En caso de querer probar el Administrador ir a la carpeta correspondiente y ejecutar el .bat. Se abrirá una pantalla de login que deberéis rellenar con los datos de la figura 4.1.2 (siempre y cuando hayais levantado con éxito el servidor en el paso 1).

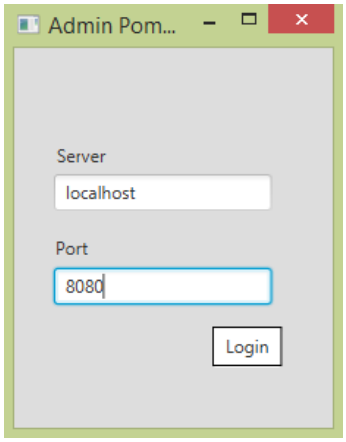


Figura 4.1.2 Ejemplo de ejecución del administrador en localhost

4.2 Interfaz

Este apartado pretende ser un manual práctico de la interfaz que nos encontraremos en la aplicación y tiene como objetivo la explicación de todo su contenido a nivel de usabilidad. Una vez leído este apartado seremos capaces de manejarnos sin ningún tipo de problema por la interfaz, navegar mediante las distintas opciones y ser capaces de hacer servir cualquiera de las funcionalidades del software. Aunque a veces es complicado separar la interfaz y su usabilidad respecto a la funcionalidad, en este apartado intentamos no entrar en ningún momento a nivel funcional, puesto que esto lo podremos encontrar en el apartado 3.2.

4.2.1 Iniciando la Interfaz

La aplicación se ejecutará directamente en la bandeja de windows. El icono de la aplicación siempre es un tomate, aunque el color puede variar según el estado en el que nos encontremos. Supongamos que es nuestra primera ejecución, entonces lo encontraremos como un tomate verde. Para poder utilizar la interfaz deberemos hacer doble click sobre el icono o un click derecho para mostrar el menú desplegable y pinchar sobre la opción "Open".

Figura 4.2.1.1 Icono Pomodoro

Cuando se abre la interfaz desde la bandeja de windows se abre la última pantalla en la que el usuario estaba antes de minimizar. En caso de que sea la primera ejecución iniciará por defecto en la pantalla principal. En caso de que mantengamos el ratón encima del icono de la aplicación veremos como aparece sobre él un mensaje donde indica el estado en el que te encuentras y el tiempo restante. De esta manera no hace falta que abras la interfaz para consultar esta información.

4.2.2 Pantalla Principal y Menú

En todo momento se ha intentado trabajar en una interfaz minimalista, fácil de usar, de aprender, intuitiva para cualquier persona independientemente del nivel de experiencia que pueda tener con los ordenadores y que fuera capaz de situar en todo momento al usuario para evitar que se perdiera entre las opciones. Por este motivo podríamos dividir cada pantalla en dos partes: el menú superior (que es igual en cualquier parte de la aplicación) y el resto del contenido. Puesto que el menú es igual sólo hablaremos de él en este apartado.

El menú de la interfaz está situado en la parte superior de la aplicación para que el usuario siempre tenga visibilidad de su contenido, las acciones que puede ejecutar y de un simple vistazo saber en qué apartado está. Esto lo conseguimos haciendo un menú de color rojo salvo la opción en la que se encuentra, que siempre tendrá un fondo blanco para diferenciarse. Por ejemplo, si observamos la Figura 4.2.2.1 (número 1) observamos que en blanco está marcada la opción central del menú que representa la página principal de la aplicación. Las otras dos opciones del menú son Settings y Stats. En cuanto al diseño hemos optado por un rojo (color identificativo de los pomodoros) y no dar ningún tipo de relieve, dejando una interfaz completamente plana y bastante minimalista.

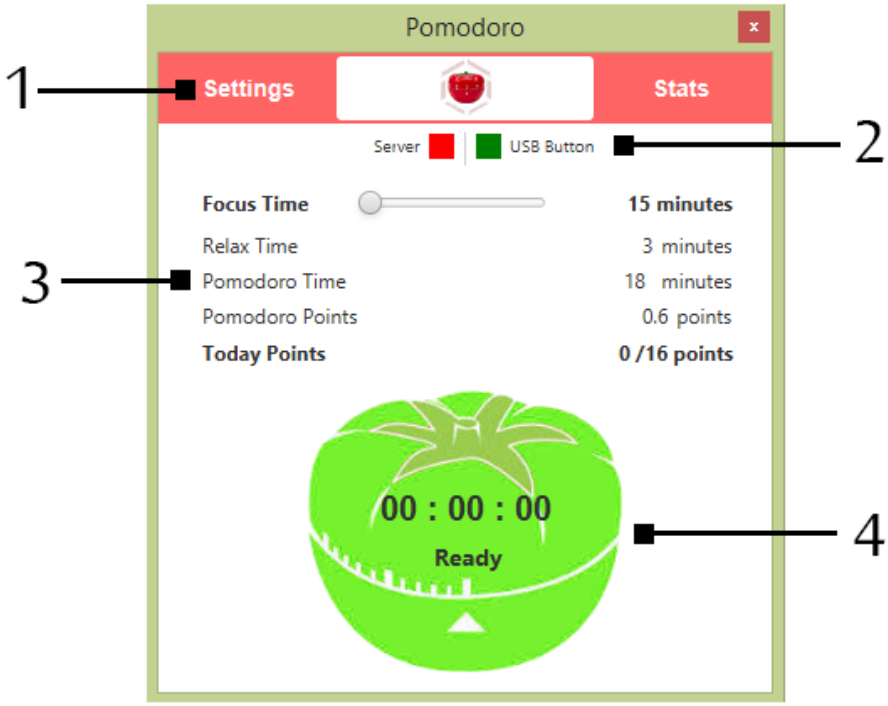


Figura 4.2.2.1 Pantalla principal de la aplicación

La página principal es la encargada de permitir configurar el siguiente pomodoro que vayamos a realizar y ver el estado del mismo. La configuración se puede realizar desde las opciones situadas en el punto 3 de la Figura anterior. Lo que el usuario debe modificar es el Focus Time mediante el slider adjunto. Los demás datos se calculan automáticamente partiendo del tiempo que el usuario haya configurado en el paso anterior (Relax Time, Pomodoro Time y Pomodoro Points). En cuanto al Today Points indica el número de puntos que ha realizado hoy dicho usuario sobre el total de puntos que puede realizar.

También observamos en la interfaz la imagen de un tomate. Esta es la representación del pulsador USB que ya hemos mencionado en varios apartados y que tendrá la misma funcionalidad. Es necesario para la gente que no tiene dicho pulsador, ya que de esta manera pueden hacer funcionar la aplicación sin ningún tipo de problema. Siempre que hablemos de alguna funcionalidad del USB Button es equivalente al pulsador representado en nuestra interfaz. De la misma manera que cambiamos el USB Button de color también cambiamos el color del icono del tomate de la interfaz, mientras que el estado y el tiempo restante también aparece representado sobre el mismo. La actualización de este tomate es en tiempo real, por lo que se puede seguir en todo momento desde aquí el tiempo restante de nuestro pomodoro o de nuestro relax time.

Por último, podemos comprobar de un simple vistazo el estado de conexión con el servidor y con nuestro USB Button. Si observamos la Figura anterior (punto 2) estos recuadros tienen una breve descripción a su lado para poder identificarlos correctamente y pueden tomar dos colores. En caso de estar rojo significa que no estamos conectados, mientras que si el color es el verde significa que la conexión se ha establecido correctamente. En este caso nos estaría indicando que tenemos el USB Button conectado y funcionando, mientras que no estamos conectados al servidor.

4.2.3 Settings

La pantalla Settings nos permite configurar todos los aspectos personalizables de la aplicación. En ella veremos que tenemos tres pestañas que podemos configurar.

La primera (1 en la Figura de este apartado) nos permite configurar los datos de conexión y conectarnos/desconectarnos con el servidor. Cuando estamos desconectados del servidor se permite la modificación de los cuatro parámetros (Server IP, Server Port, Username y Password) así como pulsar el botón "Connect" que aparecerá. Mientras el software intenta realizar la conexión dicho botón pasa a "Wait" y no nos permitirá modificar los campos hasta que no haya terminado el proceso de conexión. Una vez haya conseguido conectar pasará al estado que vemos en la Figura 4.2.3.1 donde nos habilita el botón que nos permite terminar la sesión con el servidor y poder modificar de nuevo los campos de conexión. En caso de que no consiga conectar volverá al estado inicial dejando modificar los parámetros y nos volverá a poner el botón para realizar la conexión.

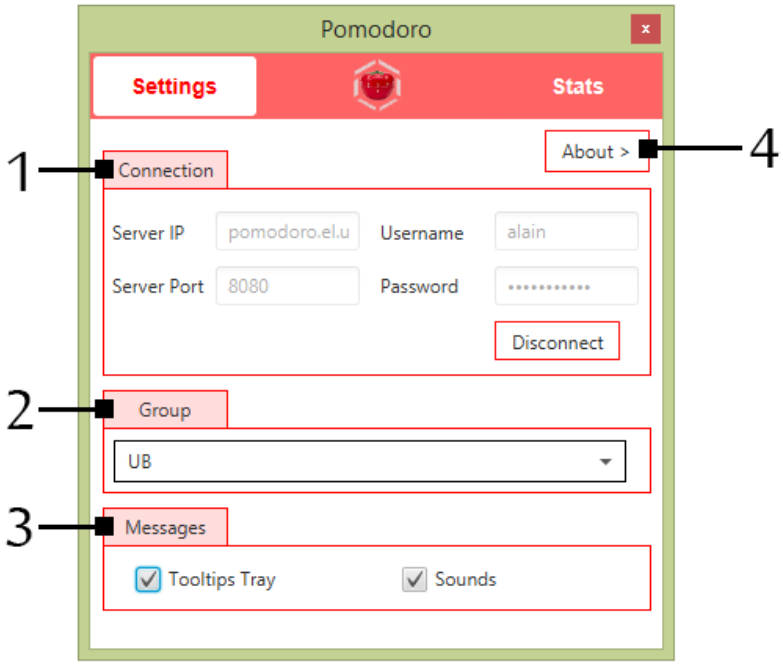


Figura 4.2.3.1 Pantalla Settings

Lo siguiente que se puede configurar es el grupo al que pertenecemos (2). Este campo aparece vacío cuando no estamos conectados a ningún servidor, puesto que los grupos es algo configurable según el servidor en el que estés. Una vez se haya establecido la conexión, automáticamente nos indica en qué grupo estamos trabajando. En caso de tener varios grupos podremos desplegar la lista desplegable y seleccionar el grupo. Podremos ver que éste nuevo grupo pasa a situarse en el primer lugar de la lista y el que se muestra por defecto sin tener la lista desplegada. Esto significa que realizaremos los pomodoros en este grupo.

En último lugar podemos configurar los sonidos y notificaciones que nos envía la aplicación a la bandeja de windows. Ambas se pueden activar o desactivar según queramos. Si nos interesa que la aplicación nos vaya notificando de los eventos que van sucediendo activaremos los mensajes y/o el sonido, pero si no queremos que nada nos desconcentre

lo desactivaremos.

También tenemos una opción para ir a la página "About" de la aplicación, donde podremos consultar una información más personalizada sobre los creadores de la aplicación. En esta sección se indican los autores del proyecto, así como la universidad y facultad en la que se ha creado.

4.2.4 Stats

El apartado de estadísticas únicamente funciona cuando se está conectado a un servidor. En caso contrario siempre aparecerán vacías, puesto que son datos que sólo se pueden conseguir mediante peticiones al servidor, único lugar donde se almacena toda la información necesaria para calcular estas estadísticas.

Actualmente podemos encontrar dos tipos de estadísticas (punto 1 de la figura): User Stats, que se encarga de mostrar las estadísticas del usuario en concreto, y Group Stats, que muestra todas las estadísticas disponibles sobre todos los miembros del grupo donde estás asignado. Comenzaremos por explicar la pestaña User Stats, de la que podemos ver un ejemplo en la Figura 4.2.4.1 adjunta.

Como hemos dicho, esta pestaña se centra únicamente en las estadísticas del usuario a nivel único. Aquí solo encontraremos nuestras puntuaciones en forma de gráficas según el filtro que asignemos. Están basadas en las puntuaciones que hemos ido conseguido a lo largo de los últimos días realizando pomodoros y nos permite ver qué días hemos conseguido más pomodoros o qué días hemos flojeado más en su realización. Esto se realiza tomando como referencia que el eje x representa fechas, mientras que el eje y indica las puntuaciones. Todas las gráficas vienen representadas por dos series: Una que nos indica el número de puntos exactos que hemos realizado durante ese día, y la otra que nos indica la suma acumulada de puntos que hemos conseguido ese mismo día.

Si observamos el punto 3 de la figura veremos que nos permite cambiar la gráfica según los días que queramos consultar. Las opciones son las siguientes:

- **Last 7 Days:** Muestra las puntuaciones que ha realizado el usuario durante los últimos 7 días naturales.
- **This Month:** Muestra las puntuaciones realizadas por el usuario en el mes actual en el que estamos.
- **Last Month:** Muestra las puntuaciones que ha realizado el usuario durante el mes anterior.

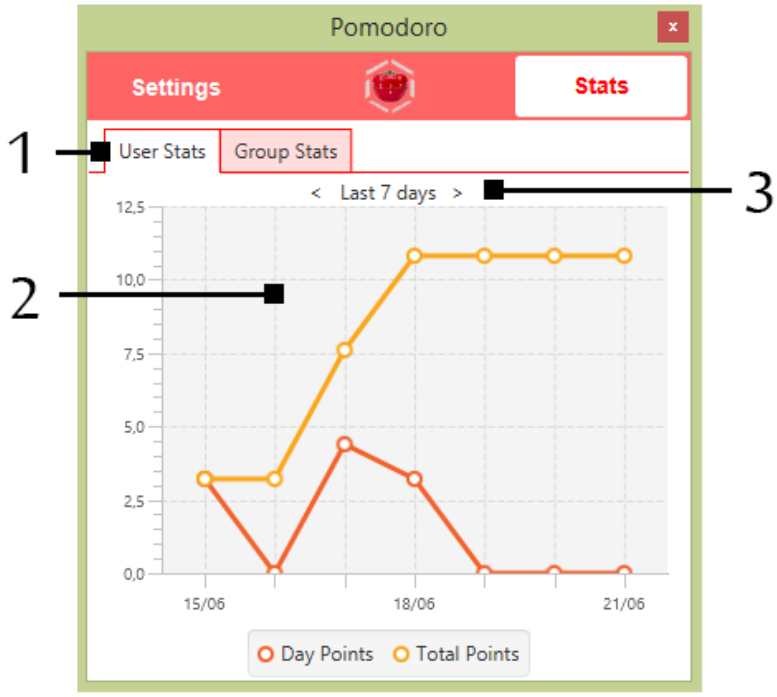


Figura 4.2.4.1 Pantalla User Stats

Si cambiamos de pestaña y entramos en la de Group Stats nos encontraremos con nuevas estadísticas, pero esta vez a nivel del grupo de trabajo entero. En este caso en lugar de mostrarse en gráficas se muestra en tablas, donde podemos ver el nombre de usuario, los puntos que lleva en el periodo de tiempo en cuestión, su estado actual (definidos en la Tabla 2.3.1) y el tiempo restante que le queda en dicho estado en caso de que sea necesario.

Igual que en el caso anterior podemos configurar el tiempo con el que queremos ver las estadísticas, siendo las opciones las siguientes:

- **Today:** Muestra los datos conseguidos únicamente en el día en curso.
- **This Week:** Muestra los datos filtrados por los días de la semana en la que estamos.
- **Last Week:** Muestra los datos de la semana pasada.
- **This Month:** Muestra los datos de este mes.
- **Last Month:** Muestra los datos del mes pasado.

La tabla se ordena por puntuación de mayor a menor y nos permite ver el estado de nuestros compañeros de trabajo para saber si podemos molestarles o no, y en caso de no poder, cuanto tiempo falta para tener su momento de descanso.

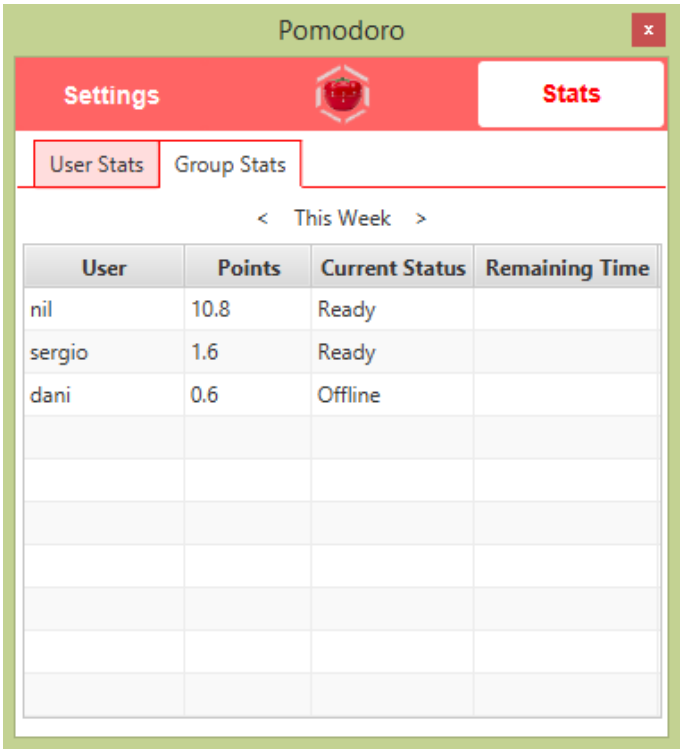


Figura 4.2.4.2 Pantalla Group Stats

4.2.5 Cerrar la aplicación

Todas las pantallas de la interfaz vienen con la opción close arriba a la derecha en forma de X. En este caso en cuestión, si pinchamos en ella lo único que conseguiremos será que la aplicación pase nuevamente a la bandeja de windows, pero no salir de ella. Si queremos cerrarla por completo deberemos hacer click derecho sobre el icono de la aplicación y clickar en la opción "Exit" de la lista desplegable.

4.2.6 Administrador

Por lo que respecta la herramienta del Administrador sigue las mismas pautas que las del Cliente. Interfaz simple, minimalista y fácil de usar. Recordemos que esta es una herramienta interna a la que únicamente tendrá acceso el Administrador, por lo que solo requiere que el mismo sepa manejarse por la interfaz y se aclare con las funcionalidades que ofrece.

Lo primero que veremos al ejecutarlo será la pantalla de login, donde tendremos que indicar a qué servidor queremos conectarnos y a través de qué puerto.

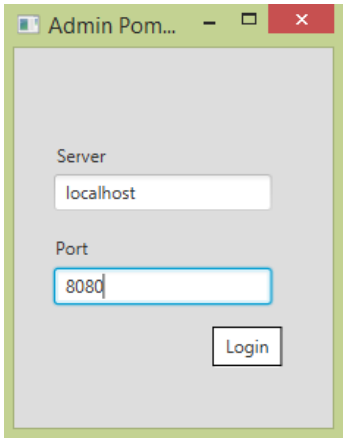


Figura 4.2.6.1 Login del Administrador

Una vez dentro veremos un menú superior de color gris donde tenemos dos opciones.

- **Users:** Nos permite ver todos los usuarios que hay creados en el servidor. Podemos editarlos, eliminarlos o incluso añadir nuevos usuarios. En cuanto a los grupos se pueden poner todos los grupos a los que quieres que pertenezca un usuario separados por una coma. Es importante escribirlos correctamente (puesto que no tenemos una BBDD donde almacenarlos todos) y no dejar una coma al final del último grupo que pongamos. Una vez terminemos de hacer los cambios tendremos que pulsar el botón "save", también situado en el menú superior. Un mensaje nos aparecerá para avisar de que se han guardado los cambios correctamente y necesitaremos reiniciar el servidor para que los tenga en cuenta.
- **Stats:** *(Figura 4.2.6.2)* Nos permite ver todas las estadísticas que hay guardadas en el servidor a nivel de aplicación. Podemos escoger el tipo de dato que queremos (días, mes o semanas) y la fecha exacta. Una vez escogido nos mostrará en forma de tabla todas las puntuaciones que encuentre el servidor aplicando dicho filtro. Estos datos son únicamente de lectura.

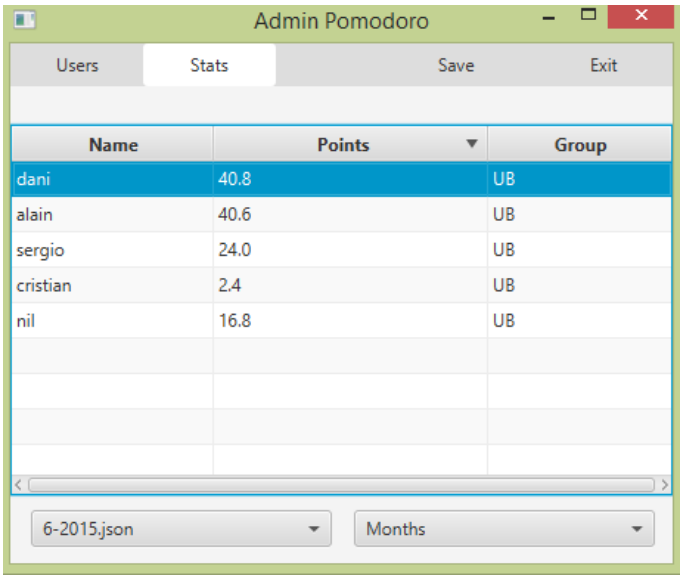


Figura 4.2.6.2 Ejemplo de ejecución del administrador en localhost

Las otras dos opciones del menú nos sirven para guardar los cambios efectuados en la pestaña “Users” y para poder cerrar la aplicación.

5 Futuras Mejoras

El número de funcionalidades y de mejoras que se le puede poner a este software son casi infinitas. Como ya se ha comentado en la introducción, esta aplicación está orientada a que sea una especie de introducción a la metodología propuesta en este TFG. Pero esto no implica que no se deba ir orientando hacia un nuevo nivel que se asemeje más a una aplicación profesional y que también sirva para usuarios más avanzados. ¿Cómo conseguiríamos esto? Pues un primer paso sería aumentando las opciones de configuración de un pomodoro. Recordemos que la metodología original de pomodoros permite realizar un descanso mayor cada cuatro realizados, y que esta funcionalidad decidimos no incluirla porque no le veíamos la utilidad cuando el objetivo es ayudar a una persona a completar tiempos de concentración. Pero sí que es necesario cuando esa persona ya es capaz de realizar pomodoros con éxito y seguidos. Por tanto, retomando el hilo de las mejoras a realizar, un primer paso para avanzar hacia un software más profesional y que abarque un mayor número de usuarios y de distintos niveles, sería implementar dicha funcionalidad. Pero no sería tan fácil como la metodología original, puesto que nuestros pomodoros pueden ser de distinta duración, por lo que se debe crear un algoritmo eficaz que según los tiempos de tus pomodoros te deje hacer un descanso más o menos largo.

Otra gran mejora y que sería de muchísima utilidad para el software sería darle la opción al usuario de ver todos los pomodoros que ha realizado durante las últimas semanas. Hasta ahora solo le mostramos sus puntos, pero nos referimos a unas estadísticas más concretas, donde pueda ver para cada día todos los pomodoro que ha intentado, pudiendo ver la configuración que realizó y si lo terminó con éxito o no. Junto a esta pantalla pueden ir unos datos en forma de porcentaje que indique al usuario su éxito en pomodoros de larga y corta duración. El objetivo de esta mejora sería ayudar al usuario a conocerse mejor, a saber qué tipo de pomodoros le van bien y con cuales rinde mejor. Siguiendo este razonamiento, esta mejora daría pie a poder implementar otra clase de características con el mismo fin. Por ejemplo, se podría incluir la opción de que el usuario pudiera escribir el motivo por el cual no ha podido finalizarlo correctamente y que dicho motivo fuera capaz de verlo en la pantalla descrita anteriormente. De esta manera, al hacer el análisis de su trabajo podrá identificar de manera más clara por qué han fallado los pomodoros y qué aspectos debe mejorar para intentar subir su estadística de completados. Estos motivos pueden ser completamente personalizables por el usuario o seleccionables de un listado fijo que pueda configurar el administrador del servidor y que permita también darle porcentajes al usuario sobre cuál es el principal motivo por el cual está teniendo pomodoros mal realizados.

Para fomentar la participación de los empleados con el software y con la metodología se puede crear un apartado donde se premie a las personas que más lo utilizan y que consiguen terminar más pomodoros con éxito. Estos premios pueden ser semanales o mensuales (o ambos) y pueden ser a nivel del grupo o a nivel de toda la empresa. Con esta medida conseguiríamos aumentar un poco la competitividad entre ellos por realizar pomodoros correctamente y así aparecer en la lista de premios. Todas estas mejoras posiblemente requieran de rediseñar algunas partes de la arquitectura planteada en este TFG. El principal cambio vendría dado por la creación de una base de datos que pudiera gestionar toda esta cantidad de datos. Hasta ahora no es necesaria, pero en caso de incluir todas las estadísticas nuevas y la gestión de esa información sería recomendable la creación de una. Sobre todo si queremos seguir aplicando mejoras, funcionalidades y nuevas estadísticas en un futuro.

Un cambio drástico en la aplicación pero que podría quedar incluso mejor, sería quitar toda la parte de estadísticas del grupo y generales del software y pasarlo a un entorno web. Consitiría en crear una aplicación web donde los empleados pudieran conectarse, loguearse y poder ver todas las estadísticas mencionadas en este apartado o implementadas en este TFG. También se podría consultar el estado actual de tus compañeros y orientar nuestro software hacia nuevos campos más allá de la metodología pomodoro. No solo ayudaríamos al usuario a gestionarse pequeños periodos de tiempo para realizar una tarea que previamente ha decidido, si no que podríamos ayudarle a llevar el control sobre todas sus tareas, incluyendo descripciones e información de interés que necesite tener agrupada en un mismo sitio. Aquí no solo se incluyen las tareas de una única persona, si no que se puede aplicar a nivel de grupo. Si todo el grupo trabaja en el mismo proyecto y se ha hecho la división de tareas, se puede ver quién las tiene asignadas, la descripción de las mismas, poder hacer comentarios en ellas para informar al equipo sobre algún punto o incluso hacerle el traspaso de la tarea a otra persona del equipo.

Por último, pero no por ello menos importante, sería interesante poder ampliar el soporte dado a los sistemas operativos. Poder pasar de Windows a Ubuntu y Mac para poder llegar a un número mayor de usuarios.

- Configurar tiempos más largos de descanso tras X pomodoros.
- Ampliar las estadísticas al usuario de sus pomodoros.
- Permitir al usuario indicar el motivo por el cual no ha podido completar un pomodoro.
- Crear una sección de trofeos para aumentar las ganas de los empleados por terminar pomodoros correctamente.
- Creación de una BBDD capaz de gestionar la nueva información.
- Dar soporte a otros sitemas operativos.
- Adaptar el software a un entorno web.
- Dar un paso adelante en la gestión y ayudar también a gestionar las tareas a nivel de equipo creando un entorno adecuado para que puedan trabajar.

6 Conclusiones

Las interrupciones en un entorno laboral es un problema con el que se encuentran todas las empresas que utilizan el nuevo modelo de oficinas abiertas. Aunque a simple vista no todo el mundo le de la importancia que merece, ésta puede terminar siendo la causa del retraso de una tarea o de un proyecto entero. Es importante ofrecerle a un empleado el entorno adecuado para que pueda concentrarse en su tarea y sacar su máximo rendimiento sin tener que estar haciendo pausas constantes con la consecuente desconcentración.

Por tanto, el objetivo de este proyecto era crear una metodología de trabajo e introducirla en un entorno laboral con la intención de evitar las interrupciones y aumentar la productividad de los empleados. Para conseguirlo se planificó la creación de un software capaz de ayudar a entender y a aplicar dicha metodología. Se analizaron diversas tecnologías para escoger las más útiles para este caso en concreto y nos decantamos por Java, RMI y JavaFX. En cuanto a la arquitectura se pensó en crear una estructura MVC y una API que usaran todos los controladores de la aplicación para hacer la funcionalidad necesaria.

Después de varios meses de trabajo podemos decir que los objetivos que nos pusimos al inicio del proyecto se han cumplido y que la aplicación cumple cada uno de los mismos. No solo hemos conseguido cumplir con las expectativas si no que, como hemos visto en el apartado anterior, creemos que las posibilidades que nos da este software son muy grandes y tiene el suficiente potencial como para ser una alternativa útil para toda aquella persona que busque mejorar su forma de trabajar. Da igual si es en un entorno laboral o es un estudiante que quiere aprovechar mejor el tiempo de estudio en casa, ya que nuestro software se adapta a las necesidades del usuario y es capaz de trabajar con distintos tipos de persona. Conseguimos introducir al usuario en la nueva metodología planteada y darle una alternativa de gestión del tiempo de trabajo. Con práctica, usando el software y aplicando el factor competitivo conseguimos que los usuarios se centren en cumplir correctamente sus pomodoros y dedicar todo el tiempo del mismo a la tarea que tengan pendiente, aumentando así la productividad que consiguen a lo largo del día.

Referències

[1] Lindsey Kaufman: *Google got it wrong. The open-office trend is destroying the workplace.*, The Washington Post, 2014.

[2] Maria Konnikova: *The Open-Office Trap*, The New Yorker, 2014.

[3] So Young Lee and Jay L.Brand: *Effects of control over office workspace on perceptions of the work environment and work outcomes.*

[4] Matthew C.Davis, Desmond J.Leach, and Chris W.Clegg: *The physical environment of the office: Contemporary and emerging issues.*

[5] Jason Feifer: *Offices for all! Why open office layouts are bad for employees, bosses, and productivity*, www.fastcompany.com, 2013.

[6] Francesco Cirillo: *The Pomodoro Technique*, 2006.

[7] Staffan Nöteberg: *Pomodoro Technique Illustrated*, 2006.

[8] Nathaniel Eliason: *Pomodoro: The Ultimate Work Habit*, www.52weeksofhabits.com, 2014.

[9] Mike Cohn: *Succeding with Agile - Software development using Scrum*, 2009.

[10] James Shore and Shane Warden: *The Art of Agile Development*, 2008.

[11] Carl Dea: *JavaFX 2.0: Introduction by Example*, 2012.

[12] Nandini Ramani: *Introducing JavaFX 2.0*, http://medianetwork.oracle.com/video/player/1191127359001

[13] Kim Topley: *JavaFX Developer's Guide*, 2011.