



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**Proyecto fin de Carrera Grado**

**Ingeniería Del Software**

**DreamStill**

**Realizado por  
Juan Ramón Rodríguez Rosado**

**Dirigido por  
D. José Antonio Parejo Maestre**

**Departamento  
Lenguajes y Sistemas Informáticos**

**Sevilla, 14 de Marzo de 2017**

# Índice general

<b>Lista de figuras</b>	<b>4</b>
<b>Lista de tablas</b>	<b>7</b>
<b>1. Introducción</b>	<b>8</b>
1.1. El problema . . . . .	9
1.2. La solución . . . . .	10
1.3. Estructura del documento . . . . .	11
<b>2. Tecnologías</b>	<b>13</b>
2.1. Tecnologías de desarrollo del proyecto . . . . .	14
2.1.1. Lenguajes de programación . . . . .	14
2.1.2. Frameworks y librerías . . . . .	16
2.1.3. Herramientas de desarrollo . . . . .	26
2.2. Tecnologías de planificación y gestión del proyecto . . . . .	28
2.2.1. Toggl . . . . .	28
2.2.2. Zenhub . . . . .	29

<i>ÍNDICE GENERAL</i>	<b>3</b>
2.2.3. Google Drive, Docs y Sheets . . . . .	29
2.3. Tecnologías de gestión del código, integración y despliegue continuo . . . . .	31
2.3.1. Github . . . . .	31
2.3.2. Travis . . . . .	32
2.3.3. Heroku . . . . .	33
2.4. Resumen de las tecnologías . . . . .	33
2.4.1. Resumen Tecnologías de desarrollo . . . . .	34
2.4.2. Resumen Tecnologías de planificación y gestión del proyecto . . . . .	35
2.4.3. Resumen Tecnologías de gestión del código, integración y despliegue continuo . . . . .	36
<b>3. Análisis y Diseño</b>	<b>37</b>
3.1. Análisis . . . . .	38
3.1.1. Backlog del producto . . . . .	38
3.1.2. Historias de Usuario . . . . .	39
3.1.3. Mockups . . . . .	42
3.2. Diseño . . . . .	46
3.2.1. Diagrama de Componentes . . . . .	46
3.2.2. Diagrama de Despliegue . . . . .	48
3.2.3. Diagrama de Capas y Tecnologías . . . . .	50
3.2.4. Diagrama UML . . . . .	51
<b>4. Planificación</b>	<b>53</b>

4.1. Detalles de la planificación . . . . .	59
4.1.1. Sprint 1 . . . . .	60
4.1.2. Sprint 2 . . . . .	62
4.1.3. Sprint 3 . . . . .	64
4.1.4. Sprint 4 . . . . .	66
4.1.5. Sprint 5 . . . . .	68
4.1.6. Sprint 6 . . . . .	70
4.1.7. Sprint 7 . . . . .	72
4.1.8. Sprint 8 . . . . .	74
4.1.9. Sprint 9 . . . . .	76
4.1.10. Sprint 10 . . . . .	78
4.1.11. Sprint 11 . . . . .	80
4.1.12. Sprint 12 . . . . .	82
4.1.13. Sprint 13 . . . . .	84
4.2. Presupuesto . . . . .	86

# Índice de figuras

2.1. Resumen Tecnologías de Desarrollo . . . . .	34
2.2. Resumen Tecnologías de planificación y gestión del proyecto .	35
2.3. Resumen Tecnologías de gestión del código, integración y despliegue continuo . . . . .	36
3.1. Mockup Página de Login y Registro . . . . .	43
3.2. Mockup Página Principal . . . . .	44
3.3. Mockup Página de Resumen de Sueño Diario . . . . .	45
3.4. Diagrama de Componentes . . . . .	46
3.5. Diagrama de Despliegue . . . . .	48
3.6. Diagrama de Capas y Tecnologías . . . . .	50
3.7. Diagrama de Clases . . . . .	51
4.1. Metodología Scrum . . . . .	54
4.2. Tablero Kanban (Zenhub) . . . . .	57
4.3. Gráfico Burndown (Zenhub) . . . . .	58
4.4. Gráfico Velocity Tracking (Zenhub) . . . . .	59

4.5. Sprint 1 . . . . .	60
4.6. Sprint 2 . . . . .	62
4.7. Sprint 3 . . . . .	64
4.8. Sprint 4 . . . . .	66
4.9. Sprint 5 . . . . .	68
4.10. Sprint 6 . . . . .	70
4.11. Sprint 7 . . . . .	72
4.12. Sprint 8 . . . . .	74
4.13. Sprint 9 . . . . .	76
4.14. Sprint 10 . . . . .	78
4.15. Sprint 11 . . . . .	80
4.16. Sprint 12 . . . . .	82
4.17. Sprint 13 . . . . .	84

# Índice de cuadros

3.1. Backlog del Producto . . . . .	38
4.1. Backlog del Producto . . . . .	55
4.2. Backlog del Producto . . . . .	56

# Capítulo 1

## Introducción

El estilo de vida moderno conlleva un ritmo frenético de las tareas que realizamos, provocando un cambio en las costumbres actuales. Sin embargo, hay actividades que han de mantenerse, ya que el descuido de las mismas tiene consecuencias desfavorables para nuestra salud. Entre éstas actividades encontramos la alimentación, el deporte y el descanso entre muchas otras. Es en ésta última actividad en la que se va a centrar el presente proyecto.

¿Sería usted capaz de responderme cuántas horas duerme normalmente? ¿Cree que duerme las horas suficientes? Probablemente no sea capaz de responder a estas preguntas, porque actualmente, debido a nuestra vida ajetreada o a la inclusión de otras actividades, no le damos la suficiente importancia al descanso. Teóricamente se deben dormir de media, dependiendo de la edad por supuesto, entre siete y nueve horas [?]. Sin embargo, ¿Cree que usted cumple con esa media?

Además, el tema de la monitorización del sueño o en inglés “Sleep Tracking” se encuentra en auge, ya que con las nuevas tecnologías podemos realizar un seguimiento más exacto y sencillo de nuestras actividades físicas, nuestra nutrición o nuestro descanso. Ésto despierta mucho interés entre los usuarios que desean llevar un estilo de vida saludable y que desean mejorar sus hábitos para mejorar su calidad de vida.

Debido a éste problema, actualmente existen diversas soluciones que permiten medir nuestro sueño, tanto en términos de cantidad de horas



como de calidad del mismo. Pero muchas de éstas soluciones son privativas y necesitan de una plataforma o aplicación específica. Así, un usuario que cambia por ejemplo de smartwatch, pulsómetro o incluso de Smartphone, una práctica bastante habitual hoy día [?], o al que le surge la necesidad de consultar sus datos desde su ordenador, puede no llegar a tener acceso a los datos de su propio sueño, ya que necesitará una aplicación que sólo se ejecuta en determinadas plataformas.

### 1.1. El problema

El problema reside en un principio en la variedad de plataformas móviles, ya que dependiendo del Sistema Operativo que tengamos en nuestro Smartphone tendremos acceso a los datos de unas u otras aplicaciones.

El proyecto nace con la idea de eliminar ésta barrera y dar soporte a todos los usuarios sin importar de que plataforma accedan mediante una aplicación web, por lo que cualquier usuario con un navegador web instalado pueda acceder a sus propios datos de sueño.

Además, nos encontramos con problemas a la hora de revisar nuestros datos de sueño, ya que en unas aplicaciones los datos son muy detallados pero carecen de funcionalidad, y en otras, ofrecen éstas funcionalidades pero el detalle que ofrecen en la aplicación no es suficiente, sin olvidar la falta de homogeneidad entre unas y otras. También, nos encontramos con las carencias a la hora de mostrar los datos obtenidos de las distintas fuentes, por ejemplo, aplicaciones como Google Fit o Apple Health se limitan a mostrar el intervalo de sueño de ese usuario, es decir, la hora a la que se quedó dormido/a y la hora a la que se despertó, sin ofrecer mayor información, aún cuando los datos detallados están disponibles.

Por tanto nuestra propuesta se basará en tres pilares fundamentales: resolver el problema de la diversidad de plataformas y abstraer al usuario que utilice la aplicación de dicha diversidad, agrupar en una sola aplicación todos los reportes que ofrecen algunas plataformas sin necesidad de que el usuario tenga que consultarlos diariamente y añadir mayor información a los reportes que utilizan distintas plataformas sobre el sueño de un usuario en la aplicación de la misma, en base a los datos reales de monitorización capturados.

## 1.2. La solución

El presente proyecto tiene la finalidad de mostrarle a los usuarios su calidad de sueño, cuantas horas han dormido, en qué estado y una representación de estos datos de tal forma que puedan intuirlos de “un vistazo”. Además, la aplicación que se va a desarrollar será transparente a los cambios entre aplicaciones de medimiento de sueño, ya que podrá obtener todos los datos tanto de una aplicación u otra.

La solución consiste en crear una aplicación web, a la que el usuario podrá acceder y revisar sus distintos “eventos”. Se conocerá como evento toda aquella actividad recopilada de otras aplicaciones en las que el usuario ha registrado una actividad de sueño. Ésta aplicación web dará a los usuarios de manera sencilla la posibilidad de integrarse con distintas aplicaciones. Con ésta solución se pretende que el usuario de una manera simple pueda ver los datos de ambas aplicaciones sin tener la necesidad de ir a buscarlo a fuentes distintas. Para que la interfaz resulte lo más intuitiva posible, se creará un calendario en la vista principal, desde el que el usuario que acceda a la aplicación podrá ver los eventos que han sido extraídos de las distintas fuentes representados en los días de éste calendario y podrá ir accediendo individualmente a cada uno de ellos sin la necesidad de verlos detalladamente todos.

Dentro de cada evento anteriormente comentado, el usuario se encontrará con dos gráficas. La primera de ellas representará los movimientos o estados (siendo el estado: durmiendo, inquieto o despierto) que ha generado el usuario durante sus horas de sueño. La segunda gráfica, representará la calidad del sueño ofreciendo un resumen de cada uno de los estados por los que el usuario ha pasado a lo largo de la actividad y dando una idea de como ha sido la calidad de sueño general de ese usuario.

El interés de ésta solución se encuentra en ofrecerle a los usuarios potenciales de la aplicación, tener una aplicación multiplataforma que recoge datos de distintas fuentes, posibilitándole a aquellos usuarios que tienen varios dispositivos para el registro de sus actividades físicas o que quieren tener una sola aplicación para la visualización de los datos del mismo, el no necesitar cambiar entre aplicaciones para ver los datos de una u otra.

Respecto al resto de alternativas, nos diferenciamos en distintos aspectos. El primero es que algunas aplicaciones se restringen a los sistemas

operativos para los que han sido desarrollados, nuestra aplicación, al tratarse de una aplicación web no tendría éste problema. Por otro lado, no todas las fuentes se sincronizan con las aplicaciones que traen los sistemas operativos para el registro de la actividad física, lo cual imposibilita el poder almacenar los datos de todas las fuentes que deseemos en éstos, nuestra aplicación se destaca por ofrecer de manera más directa una integración con éstos servicios.

Por otra parte cabría pensar en usar la aplicación que se han desarrollado nativamente para estos dispositivos. Lo que diferencia el presente proyecto de éstas aplicaciones, es que permite mostrar más detalles sobre los datos, mostrarlos en una estructura más representativa y aunar distintas fuentes, sin importar la plataforma o marca de donde provengan.

### 1.3. Estructura del documento

En el capítulo 2, se definirán las tecnologías que se han usado a lo largo de la realización del proyecto, nombrándolas y explicando para que se han usado cada una de ellas.

En el capítulo 3, se mostrará el análisis y el diseño de la aplicación, el lector comprobará cómo es la estructura de la aplicación y el diseño de la misma, gracias a distintos diagramas y gráficos que le ayudarán a interpretarlos.

En el capítulo 4, se verá la planificación que se ha seguido a lo largo del proyecto, la metodología que se ha seguido para dicha planificación y el número de interacciones, junto con la descripción de las mismas, que se han seguido en el transcurso del proyecto.

En el capítulo 5, se tratará el desarrollo del proyecto, tratando niveles más técnicos del mismo y mostrando como se han ido desarrollando las distintas clases, componentes y vistas que han sido necesarias para que el proyecto tenga el acabado final que se presenta.

En el capítulo 6, se detallarán las pruebas que se han diseñado para validar la solución propuesta, por lo que se comprobará el apartado referente al “testing” de la aplicación.

En el capítulo 7, se explicará el manual de usuario, en el que gracias a las historias de usuario, el lector podrá conocer las distintas funcionalidades de la aplicación y como llevarlas a cabo.

En el capítulo 8, se determinarán las conclusiones a las que se han llegado tras la realización del proyecto.

En el capítulo 9, se encuentran todas las referencias que se mostrarán a lo largo de la presente memoria, muchas de ellas serán útiles para el lector que desee conocer la fuente de las que provienen algunos de los datos que se muestran.

## Capítulo 2

# Tecnologías

La cantidad de tecnologías que hemos utilizado para realizar el presente proyecto es amplia, ya que algunas de las tecnologías principales usadas para su desarrollo necesitan de herramientas adicionales para realizar mejor su función.

En cuanto a las tecnologías de desarrollo, se han utilizado tecnologías recientes y que están muy enfocadas al desarrollo de aplicaciones web, utilizando lenguajes de programación muy cercanos a éste tipo de aplicaciones, como JavaScript ó TypeScript.

Respecto a las herramientas de planificación, éstas han ayudado a la hora de mantener una planificación y calcular posibles desvíos o impactos de un nuevo cambio. Los resultados obtenidos mediante éstas herramientas se expondrán en el capítulo 4 de la presente memoria.

En cuanto a las tecnologías de gestión del código, de integración y de despliegue continuo, se han utilizado plataformas web, lo que posibilita que la información que ofrecen esté online y sea visible para cualquier usuario que esté interesado en conocerla.

## 2.1. Tecnologías de desarrollo del proyecto

A su vez, dividiremos éste apartado en los siguientes: Lenguajes de programación, frameworks y librerías y herramientas de desarrollo.

### 2.1.1. Lenguajes de programación

#### JavaScript



<https://www.javascript.com>

JavaScript es el lenguaje principal del proyecto. Su elección se debe a que es un lenguaje actual y versátil y pensado para ser ligero pero eficaz. Además, al elegirse Node.js para el Back-end, la elección de este lenguaje cobra aún más sentido ya que como se verá posteriormente, Node.js es un entorno de ejecución para JavaScript. Además, se usará éste lenguaje para la lógica en el Front-end y será muy relevante para su correcto funcionamiento y visualización.

Éste lenguaje de programación se diseñó por Netscap Y Mozilla en 1995 y se vio influido por lenguajes como Java, C o Python [?]. Es un lenguaje de programación interpretado y orientado a objetos, pero a diferencia de otros lenguajes como Java es débilmente tipado y dinámico. Es usado principalmente en el Front-end o “lado del cliente”, aunque actualmente se usa en el Back-end o lado del servidor, entre otros gracias a Node.js. Todos los navegadores modernos lo integran. Es un lenguaje de programación que apareció como un lenguaje de “scripting” pero que como se ha comentado tiene muchas más aplicaciones actualmente, llegando a niveles de complejidad y prestaciones comparables con otros lenguajes de primer nivel.

Se podría decir que JavaScript ha sido contemplado durante años por muchos usuarios como un lenguaje para dar lógica a las páginas web,

pero actualmente su uso es mucho más extendido, y está presente en muchos ámbitos.

## TypeScript



<https://www.typescriptlang.org>

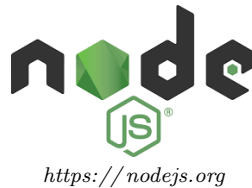
TypeScript es otro de los lenguajes relevantes del proyecto, gracias a él se consigue una mayor lógica en la aplicación sin tener que recurrir a JavaScript, aunque éste sea el producto final de la compilación de TypeScript. Su inclusión en el proyecto se basa en la facilidad para programar una vez que se han adquirido los debidos conocimientos previos y en que es el lenguaje en el que se ha programado Angular2, del que hablaremos más adelante y que ha sido elegido para representar la parte dedicada al Front-end de la aplicación.

Fue diseñado por Microsoft en 2012 y se influyó en lenguajes como Java, C++ y por supuesto JavaScript [?]. Es un lenguaje de programación libre y Open Source mantenido por su diseñador, Microsoft. Está enfocado a aplicaciones escalables construidas con Javascript, añadiendo tipos, clases y módulos a éste lenguaje. Soporta herramientas para un escalado importante en aplicaciones construidas bajo JavaScript para cualquier navegador, servidor o sistema operativo. Como se ha comentado anteriormente, es un lenguaje compilado el cual produce ficheros en JavaScript.

Finalmente destacar que es un lenguaje tipado, y que Visual Studio Code, del que se hablará más adelante, tiene un gran enfoque en éste lenguaje, por lo que ambos son la pareja perfecta para un desarrollador que desee programar en TypeScript.

### 2.1.2. Frameworks y librerías

#### Node.js



Node.js es el entorno elegido para el Back-end de la aplicación. Su elección se basó en la fama que está adquiriendo ésta tecnología actualmente y en la gran comunidad que la apoya, la cual proporciona innumerables paquetes que se pueden añadir a la aplicación y que ahorran tiempo de desarrollo y de programación al estar ya programados. Además Node.js ofrece un robusto conjunto de opciones para aquellos desarrolladores que desean realizar la implementación de una aplicación web, como es el caso del presente proyecto.

Éste entorno en tiempo de ejecución es multiplataforma, de código abierto y se basa en la capa del servidor (Back-end como se ha comentado anteriormente), fue lanzado en 2009 por Ryan Lienhart [?] y permite lanzar una sencilla aplicación web que nos salude con el típico “Hola mundo” en una dirección Http local en sólo unas 10 líneas de código, lo cual resulta de gran utilidad a la hora de implementar las distintas necesidades que satisface un servidor web. Cómo se ha comentado con anterioridad, Node.js permite ejecutar código JavaScript en el lado del servidor, difundiendo éste lenguaje más allá del Front-end, lugar al cual se suele estar acostumbrado ver, ésta implementado con C++ y JavaScript y gracias a ser multiplataforma permite que sea utilizado por cualquier desarrollador sin necesidad de tener un sistema operativo u otro.

Gracias a éste entorno, se considera que se ha agilizado en gran medida la programación de la parte servidor o Back-end de la aplicación, ya que una vez superada la curva de aprendizaje como ocurre en el resto de entornos de éstas características, la programación de métodos en el servidor se hace muy sistemática y permite en pocas líneas de código realizar funcionalidad para los que otros entornos necesitan varias clases con varias



líneas de código para realizar un comportamiento similar. Sería interesante comentar para finalizar éste apartado que Node.js nos ofrece en su web dos enlaces de descarga, uno a la más actual y otro a la versión estable (nombrada con las siglas LTS) en el caso de la aplicación del presente proyecto, al ser una tecnología relativamente nueva y al ser crítica para la aplicación, se decidió tomar la opción LTS que nos ofrece completa estabilidad y la seguridad de tener una versión que ha sido probada y pasada a la rama estable del proyecto Node.js.

## Npm



<https://www.npmjs.com>

Npm ha ayudado a lo largo del desarrollo del proyecto, ya que gracias a él se ha podido instalar un elevado número de paquetes, configurándose fácilmente, llevando el control de sus versiones de una forma simple y desinstalando los que ya no necesitamos de una manera tan sencilla como se instalan, sin preocuparnos de nada más de lo que realmente nos importa, que es la generación del código fuente y la importación de paquetes que necesitamos para nuestro proyecto.

Por tanto, se deduce que efectivamente Npm es un gestor de paquetes. Éste gestor de paquetes viene instalado a partir de la versión 0.6.3 de Node.js y se ejecuta desde la línea de comandos teniendo un fácil acceso a las dependencias de una aplicación escrita bajo Node.js. Permite a los desarrolladores instalar aplicaciones Node.js a partir de un único archivo de configuración y está escrito completamente en JavaScript. Su desarrollador fue Isaac Z. Schlueter, el cual lo desarrolló a raíz de una frustración que sufrió mientras trabajaba con CommonJS.

Finalmente cabría destacar que es un gestor de paquetes para Node.js muy acogido y aclamado por la comunidad de desarrollos enfocados a éste tipo de proyecto y que a pesar de algunos fallos a bajo nivel, para el

uso cotidiano que suele darle un desarrollador de éste tipo de aplicaciones ofrece una gran sencillez y garantía de comodidad al usuario que lo use.

## Express

The logo for Express.js, featuring the word "express" in a lowercase, sans-serif font. The letters are thin and light gray.

*<http://expressjs.com>*

Express ha sido un compañero indispensable para el desarrollo en el Back-end de la aplicación, gracias a él se ha podido agilizar aun más el desarrollo de ésta parte de la aplicación, proporcionando un gran número de funciones y métodos muy usados en las operaciones que proporcionan la mayoría de servidores, tales como: el renderizado de páginas web (en Html o en Jade tecnología que se verá más adelante), sockets, métodos para registrar peticiones Get o Post entre otras, manejo de Cookies, manejo de los datos de las Request o las Responses y muchas más funcionalidades.

Es un framework para Node.js gratis y de código libre que fue lanzado en 2010 por varios desarrolladores [?]. Está escrito en JavaScript y es al igual que Node.js multiplataforma. Es muy usado en la construcción de servidores para aplicaciones web y hay una gran cantidad de documentación al respecto en Internet.

Gracias a él, se ha conseguido condensar gran parte de la funcionalidad de la parte del servidor de la aplicación en una sola clase JavaScript, la cual al ser llamada mediante Node.js nos proporciona una dirección local en la que obtenemos un servidor completamente funcional.

## Angular 2



<https://angular.io>

Angular ha sido el elegido para representar gran parte de la parte del Front-end (en concreto todas las vistas que se visualizan una vez que ingresamos en la aplicación con un usuario registrado en la misma). Apareció hace más de 4 años conocido como AngularJS [?] y ha eliminado el “JS”, no sólo porque ya no se base en éste, utilizando TypeScript en su lugar, si no por la gran evolución que ha sufrido en su segunda versión.

Angular 2 es robusto, y apasiona a los desarrolladores que lo usan, entre los que me incluyo a mi, el autor de ésta memoria, ofrece una gran cantidad de características que permiten al desarrollador delegar gran parte de la funcionalidad de la aplicación en el cliente, lo cual agiliza en gran medida la carga de la aplicación en dispositivos como los que solemos usar, que están más que capacitados para ejecutar aplicaciones Angular sin la necesidad de estar realizando peticiones al servidor constantemente. Al estar programado en TypeScript supone una capa de abstracción superior a la que ya nos ofrece JavaScript y además el JavaScript que se genera al compilar éste código está optimizado y minimizado, por lo que conseguimos mayor rendimiento y mayor facilidad al programar. Angular nos ofrece muchas características interesantes, desde su propio “router” para navegar por las distintas páginas de la aplicación sin necesidad de llamar al servidor, renderizado universal, programación reactiva gracias a librerías como Rxjs o la creación de componentes e incluso servicios, que dirigirán la aplicación desde el lado del cliente como si de un servidor se tratase.

Es fácil por tanto olvidar en ocasiones en que lado de la aplicación estamos, ya que debido al gran abanico de funcionalidades que nos ofrece Angular y concretamente más en Angular 2, en ocasiones podríamos llegar a plantearnos en qué lado de la aplicación estamos, si servidor o cliente, y es éste potencial entre otros el que está consiguiendo que Angular coseche tanto éxito y se esté conformando como una solución simple y profesional

en el mercado de las aplicaciones web.

## Firestore



<https://firebase.google.com>

Firestore es la base de datos escogida para la aplicación. Nos planteamos distintas opciones, y tal como se detalla en el modelo MEAN (MongoDB + Express + Angular + Node) muy típico en las aplicaciones web desarrolladas en Node.js, la opción que se suele elegir es MongoDB, sin embargo, nosotros nos hemos decantado por ésta opción. Desechamos las bases de datos relacionales ya que para nuestra aplicación tenían más sentido las no relacionales, además, Firestore cuenta con varios factores a tener en cuenta, entre los que se encuentran la actualización en tiempo real de los datos, en todos los dispositivos que accedan a la Base de datos, o la utilización de una API REST para acceder a los mismos.

Fue lanzada en Abril de 2012 y es además de una aplicación web, una aplicación móvil con herramientas e infraestructuras diseñadas para ayudar a los desarrolladores a construir aplicaciones de alta calidad [?]. Ofrece una gran cantidad de servicios a los desarrolladores, aunque en nuestro caso nos centraremos en el servicio de almacenamiento en su Base de Datos, conocido como “Realtime database” el cual ofrece librerías para clientes que utilicen su plataforma y que permiten la integración de éstos con la misma [?].

Gracias a ésta Base de Datos, se ha conseguido almacenar y gestionar los datos de una forma ágil y familiar, ya que al acceder a ellos a través de métodos como “GET” o “PUT” resulta mucho más cómodo que tener la necesidad de un componente aparte para configurar la conexión con la Base de Datos y gestionar todas las consultas a partir del mismo.

## Selenium



<http://www.seleniumhq.org>

Selenium será el encargado en la aplicación de realizar las pruebas automáticas de interfaz. Gracias a éste entorno de pruebas, se podrán ejecutar distintos scripts con los que comprobar la funcionalidad de la aplicación del proyecto, detectando fallos en la interfaz tanto a nivel de funcionalidad o incluso visuales, si los distintos elementos no son representados como deberían. Incluye la posibilidad de realizar las pruebas casi en la totalidad de los navegadores web modernos, en nuestro caso, sólo lo utilizaremos con Google Chrome en las pruebas ejecutadas en local y con Mozilla Firefox en las pruebas ejecutadas en el servidor de Integración Continua.

Fue desarrollado originalmente por Jason Huggins en 2004 y es un proyecto de código abierto [?]. Permite ser ejecutado además de en la mayoría de navegadores modernos, en los sistemas operativos Windows Linux y OSX, por lo que ofrece sus servicios a la mayoría de desarrolladores sin necesidad de preocuparse por el sistema operativo que ejecuten. Ofrece un IDE pero sólo esta disponible en modo de extensión para Mozilla Firefox, y permite grabar editar y depurar pruebas.

## Jasmine



<https://jasmine.github.io>

Jasmine será el framework elegido para diseñar los tests de la aplicación. Ofrece una solución muy popular para tests unitarios y/o funcionales.

Junto con Webdriver, que es la herramienta encargada junto con Selenium de ejecutar dichos test en un navegador, ofrecen una experiencia amena y didáctica, permitiendo ver visualmente los fallos que cometemos ya sea a la hora de escribir los tests o de ejecutarlos.

Fue desarrollado por Pivotal Labs en Septiembre de 2010 y es Open Source, es un framework de testing para JavaScript y está muy influenciado en otros frameworks como ScrewUnit o JSSpec [?].

## Cucumber



<https://cucumber.io>

Cucumber será el framework de testing que nos permitirá escribir los tests en texto plano, y a partir de éstos sacar el código en Jasmine necesario para ejecutarlos. Gracias a éste framework, la escritura de los tests resulta mucho más sencilla y más humana, ya que los tests se escriben en lenguaje natural siguiendo sólo algunas especificaciones que necesita el lenguaje para relacionarlos con los tests escritos en JavaScript.

Proporciona por tanto, una experiencia a la hora de realizar los tests más amena que a la que normalmente estamos acostumbrados, ya que nos permite aunque sea por un momento abstraernos del código y pensar en la verdadera funcionalidad de los tests, sin tener que preocuparnos en un principio de como llevar éstos a cabo. Además, la curva de aprendizaje es sencilla, ya que sólo necesitamos conocer como se escriben los tests en lenguaje natural, puesto que al lanzar éstos el propio framework nos dirá cómo serán los métodos para enlazarlos con dichos tests en lenguaje natural y sólo tendremos que preocuparnos de desarrollar el código para la ejecución de los mismos.

## Protractor



<http://www.protractortest.org>

Protractor es el framework elegido para la ejecución de los tests escritos en Cucumber y posteriormente traducidos a Jasmine. Al incluir por defecto Selenium, su integración con éste es sencilla, sin embargo, con Cucumber la cosa cambia. Ha sido necesario una curva de aprendizajes y distintos ensayos de prueba/error para conseguir integrar ambos frameworks, pero he de decir que los resultados los merecen.

Protractor es una herramienta de Google para la implementación de pruebas E2E principalmente en aplicaciones de Angular [?], aunque como ellos mismos indican también puede utilizarse para realizar pruebas en otro tipo de aplicaciones, como es nuestro caso, ya que la página de Login por ejemplo no está implementada bajo éste lenguaje. Su instalación resulta sencilla, y gracias a un fichero de configuración que además ofrece varias opciones para customizar las distintas funciones de Protractor, podremos “tener corriendo” nuestros tests en poco tiempo.

## Passport



<http://passportjs.org>

Passport es el encargado de la autenticación de los usuarios en nuestro sistema. Es un framework de Node.js que cuenta con gran popularidad entre los desarrolladores ya que ofrece un gran número de posibilidades para

realizar la autenticación a sus sistemas, entre las que se incluyen el acceso con una cuenta de Google, Twitter ó Facebook entre otras. Por supuesto también ofrece un método de autenticación local, con el que el desarrollador puede manejar sus usuarios en función de sus propios datos almacenados en su Base de Datos.

En nuestro caso utilizaremos el método de autenticación local, los datos de los usuarios estarán almacenados en Firebase y Passport se encargará de verificarlos cuando un usuario desee acceder a la aplicación. Su implementación se realiza mediante sesiones en nuestro caso, aunque también ofrece la posibilidad de ofrecer dicha implementación mediante tokens, los cuales actualmente son muy usados en frameworks de autorización como OAuth2.

## Jade



<https://pugjs.org>

Jade es la librería escogida para las páginas que se renderizarán directamente en Node.js. Se ha escogido Jade ya en mi opinión ofrece una sintaxis más cercana a los desarrolladores de aplicaciones web, sin olvidar el hecho de que incluye funcionalidades extras. Además, hay montones de páginas que convierten una página escrita en html a jade en cuestión de segundos y viceversa, por lo que el cambio entre un lenguaje de escritura y otro es inmediato, siendo bastante interesante probar éste paquete que además es muy popular entre los desarrolladores de aplicaciones web que tienen Node.js como Back-end.

Jade ha sido renombrada recientemente a Pugjs, ya que hubo problemas con el nombre que ya estaba siendo utilizado por otra empresa del sector con anterioridad. En Internet podemos encontrar montones de guías de como escribir nuestras páginas web en Jade en aplicaciones con Node.js y Express, por lo que el uso de esta librería está siendo cada vez más extendido.



## Mail Listener

Mail Listener es una librería esencial en algunas funcionalidades de nuestra aplicación. Gracias a ésta librería, se leen los correos que reciben el e-mail de la aplicación, entre ellos los correos de Morpheuz sobre los datos de los usuarios y pueden ser procesados y almacenados en la Base de Datos. Además, ofrece una posibilidad de demonio (más comúnmente conocido como Daemon en inglés) que nos estar a la espera de nuevos e-mails y procesar todos o alguno de éstos cuando lleguen.

En el caso de la aplicación existe un filtro para los correos de los usuarios de la aplicación Morpheuz, de tal forma que cada vez que un correo llega, se identifica el usuario al que pertenecen y se procesa el correo con los datos de ese usuario.

## Node Mailer



Node Mailer permitirá a nuestra aplicación enviar e-mails a los usuarios. En un principio se incluyó con la finalidad de enviar los correos de recuperación de contraseña a los usuarios que la olvidasen. Ésta librería ha adquirido gran fama como en el caso de Passport gracias a la gran variedad de servicios que incorpora por defecto, entre los que se encuentran el envío de correos por el servidor de Google u Outlook por ejemplo.

Su configuración resulta rápida como podemos ver en el ejemplo que ofrecen en su web y en poco tiempo estaremos mandando nuestro primer e-mail al usuario que deseemos, sin necesidad de configurar parámetros demasiados avanzados.

## Typings

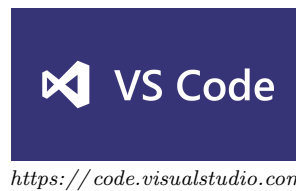


Typings es un paquete indispensable para los desarrolladores que programan en TypeScript. Provee una manera simple de manejar e instalar definiciones en éste lenguaje. Usa un archivo de configuración a partir del cual es capaz de resolver dichas definiciones.

Typings tiene capacidad para recoger los datos de distintas fuentes, aunque en nuestro caso la usaremos mayoritariamente para definiciones en paquetes instalados desde NPM.

### 2.1.3. Herramientas de desarrollo

#### Visual Studio Code



Es el editor de código fuente elegido para desarrollar el código fuente del proyecto durante la duración del mismo. Existen alternativas similares como lo son: WebStorm, Atom o SublimeText, todos muy aclamados y usados por un gran número de usuarios. He de reconocer que éste editor de código fuente era desconocido para mí, ya que sólo había utilizado la versión de Visual Studio estándar o la que estamos acostumbrados a usar a la hora de programar lenguajes como C++. Fue la sencillez de la interfaz, la opción

de una gran cantidad de extensiones útiles ya la cantidad de opciones que trae ya instaladas; como la pestaña para la gestión del código en Github, la pestaña para Debuggear, o la consola en la misma pantalla del código entre otras, lo que me hizo aventurarme a utilizar éste editor, aconsejado por mi tutor el cual había oído muy buenas críticas sobre el por parte de los desarrolladores de Angular 2. A fecha de redacción de la presente memoria, no me arrepiento de mi elección, ya que creo que fue muy acertada, es ligero pero robusto, sencillo pero con multitud de funciones útiles y en general no hechas nada en falta respecto a otras opciones, es más, incluso en otras opciones he echado en falta opciones que trae preinstaladas éste.

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS, por lo que su punto fuerte entre otros es ser multiplataforma. Además, es muy customizable, tanto en temas, como en accesos directos desde el teclado, como en ajustes del propio editor. Es gratis y libre a pesar de que su licencia es una licencia propietaria por parte de Microsoft. Fue anunciado el 29 de Abril de 2015 [?]. Entre otras muchas características, VS Code, cuenta con resaltado de sintaxis para lenguajes como Batch, C++, Java, Objective-C, etc. “Snippets” para lenguajes como Groovy, Markdown o Swift, sugerencias inteligente de código en lenguaje como JavaScript o TypeScript, refactorización de código en lenguajes como TypeScript o C# y “Debugging” en lenguajes como JavaScript y TypeScript para proyectos en Node.js (éste último sería nuestro caso), y todo ésto lo incluye por defecto, pudiendo aumentar aún más sus características a través de extensiones. Es por estas características entre otras, un editor de código fuente recomendable para desarrolladores que busquen un entorno más ligeros a los ya acostumbrados en los distintos lenguajes de programación, pero sin perder demasiadas características frente a éstos.

## 2.2. Tecnologías de planificación y gestión del proyecto

### 2.2.1. Toggl



<https://toggl.com>

Toggl es una gran herramienta para la gestión del tiempo y de la planificación del mismo en proyectos. Con anterioridad había conocido herramientas similares, ya que tarde o temprano todos tenemos la necesidad de conocer el tiempo que invertimos en las distintas tareas o proyectos que realizamos. Toggl permite medir éste tiempo de una manera eficaz y sencilla. Su interfaz es clara, permitiendo pulsar el botón “Play” para medir el tiempo sin necesidad de describir que tarea vamos a realizar, ya que para Toggl lo importante es empezar con la actividad sin necesidad del conocido “ruido” al que solemos estar sometido, pudiendo definir el propósito de esa actividad después.

Toggl ofrece una solución multiplataforma, en toda clase de dispositivos, desde su interfaz web, su extensión para navegadores basados en Chromium (incorporando su botón para empezar una actividad en páginas tan importantes como Github o Trello) y sus aplicaciones para escritorio, smartphone o incluso smartwatch, por lo que no poder empezar a cronometrar una actividad de manera sencilla no será nunca más una excusa.

Además, ofrece informes detallados de las actividades que hemos realizado exportando los mismos a distintos formatos, por lo que al final de un proyecto podemos por ejemplo, revisar el número de horas imputada de cada uno de los miembros en cada actividad de una manera sencilla, ya que con sólo pulsar un botón tendremos un informe con toda ésta información. Por todo ésto y mucho más, Toggl es una herramienta muy elegida y aclamada, con la que podremos medir nuestro tiempo y llevar un control de en qué lo invertimos.

## 2.2. TECNOLOGÍAS DE PLANIFICACIÓN Y GESTIÓN DEL PROYECTO 29

### 2.2.2. Zenhub

# ZenHub

<https://www.zenhub.com>

Zenhub es la herramienta que nos ayudará en la planificación del proyecto. Zenhub es una extensión para navegadores basados en Chromium que permite añadir funcionalidades a Github, entre las que se destacan la inclusión de un tablero Kanban, la posibilidad de escribir historias épicas o asignar puntos de historias a las issues o la elaboración de reportes o un burndown del milestone en el que nos encontramos.

Ofrece una solución recomendable a aquellos desarrolladores que utilizan la plataforma Github y quieren seguir una metodología ágil en el desarrollo de su proyecto. Es una tecnología usada por grandes empresas como se destaca en la portada de su página. Cómo ellos mismos recomiendan, junto con Slack y Github, Zenhub es una pieza que encaja a la perfección y que ayudará tanto a los desarrolladores como al director del proyecto a llevar un mejor seguimiento de las distintas actividades del mismo.

### 2.2.3. Google Drive, Docs y Sheets

# G Suite

<https://gsuite.google.com>

Google ofrece ésta completa suite ofimática online de manera gratuita, en nuestro caso lo utilizaremos para la documentación generada durante el proyecto. Entre otras muchas características la que más se resalta

es la colaboración en tiempo real de distintos miembros, por lo que ambos miembros de un equipo pueden estar escribiendo a la vez en un documento sin necesidad de verse bloqueados por el otro.

Gracias a Drive almacenaremos nuestros archivos de manera online y multiplataforma, pudiendo acceder a ellos en todo momento. Con Docs podremos escribir documentos y exportarlos a la mayoría de formatos de la actualidad y además en ellos podrán colaborar distintos miembros de manera online y en directo, sin necesidad de tener que estar transfiriendo el documento entre ellos. Con Sheets dispondremos de una hoja de cálculos muy robusta, con la que generar los datos que necesitamos y al igual que en Docs, podremos compartirla con otros miembros y exportarla la mayoría de formatos existentes en la actualidad.

En resumen, gracias a éste conjunto, se puede generar y almacenar documentación en un equipo de manera online sin la necesidad de tener que estar transfiriendo los archivos de un lado a otro y trabajando con mayor comodidad, ya que todos los cambios y archivos permanecen en la nube.

## 2.3. Tecnologías de gestión del código, integración y despliegue continuo

### 2.3.1. Github



<https://github.com>

Github es el gestor de código fuente elegido para almacenar todo el código del proyecto. Sin embargo, no sólo nos limitaremos a utilizar Github con ésta intención, si no que almacenaremos las tareas del proyecto en forma de Issues y gracias al complemento Zenhub podremos llevar a cabo la planificación del mismo en ésta herramienta. Github funciona bajo la tecnología Git, la cual permite tener repositorios con una gestión distribuida y que está integrada en Visual Studio Code. El resultado es una gestión del código cómoda y con la que no tendremos que estar cambiando entre herramientas para llevar nuestro código fuente al día.

Github es plataforma de Git, gracias a su interfaz web y a la gran cantidad de funcionalidades y repositorios que ofrece, es una de las opciones más escogidas para los desarrolladores que prefieren Git a otras alternativas como SVN. El código que se almacena en nuestro repositorio se almacena de manera pública, aunque Github ofrece también repositorios privados. La elección de mostrar el código fuente de nuestra aplicación, es poder obtener feedback del mismo por parte de futuros usuarios y que se ofrece seguridad a los usuarios que conocen la materia y usan la aplicación, porque pueden ver ellos mismos como está construida la aplicación.

En definitiva Github ha sido una herramienta indispensable a lo largo del desarrollo del proyecto, sobre todo en mi caso particular, ya que tengo dos equipos de trabajo y gracias a Github no he notado el cambio entre uno y otro, ya que era tan fácil como subir el código al final de cada sesión y descargarlo al principio de la otra. Para más información sobre nuestro código en Github puede consultar el código del mismo en el repositorio de nuestra organización: <https://github.com/TFG-US-DreamStill>

### 2.3.2. Travis



Travis es el servidor de Integración Continua escogido para el proyecto. Ofrece integración con Github, de tal forma que cada vez que se realiza un “push”, es decir, se sube código al servidor remoto en Github de la aplicación, éste lanza una serie de comandos que pueden definirse fácilmente en un archivo de configuración “.yaml” [?], en el caso del proyecto, se prepararía el entorno de pruebas, se construiría la aplicación y se lanzarían las pruebas correspondientes, comprobando que todo funciona correctamente.

Los servidores de Integración Continua tienen muchas funcionalidades útiles, pero entre todas destacaría la funcionalidad de lanzar las pruebas automáticamente en cada subida de código fuente y comprobar que todo sigue funcionando correctamente. Éste punto es crítico para pruebas de regresión, ya que puede ocurrir que modifiquemos algo que afecte a algo que no teníamos planeado y éste deje de funcionar, gracias a un servidor de Integración Continua y a las debidas pruebas éste problema se vería en el mismo momento en el que subimos el código fuente. Aunque se puede activar o no, una gran funcionalidad de Travis es la notificación al usuario por correo electrónicos de eventos importantes en su código fuente, entre los que se destacan las notificaciones en el caso de que algún test falle, o en el caso de que se arregle un test, es decir, ese o esos mismos tests que fallaban antes, tras una subida vuelven a dar una respuesta positiva.

Gracias a Travis por tanto se ha realizado un seguimiento más directo del código fuente. He de reconocer que en alguna ocasión he recibido emails con fallos en los cambios que había reconocido, es más, Travis añade al título de los “commits” en Github un “tick” o una “x” en función del resultado de éste, y gracias a ésta información he podido detectar fallos en mi código fuente de una forma más versátil y cómoda. Se puede consultar la información del proyecto en Travis en: <https://travis-ci.org/TFG-US-DreamStill>



### 2.3.3. Heroku



Heroku es el servidor de Despliegue Continuo escogido para el proyecto. Ofrece integración con Github y Travis, ya sea con la combinación de ambos o sólo con alguno de ellos. Con Github y al igual que ocurre con Travis al realizar un “push”, se construiría el proyecto y se desplegaría en uno de los contenedores que ofrece Heroku. En el caso de integrar además Travis, se podría configurar a Heroku para que sólo realice el despliegue de la aplicación en el caso de que Travis no detecte ningún error en la misma.

Ésta herramienta resulta de gran utilidad, ya que tendremos los resultados de nuestro código fuente en todo momento de manera Online y cualquier usuario o desarrollador podrá consultarlo de manera actualizada y en directo, sin necesidad de descargar el código fuente del proyecto y arrancar el servidor.

La elección de Heroku ha estado motivada por la integración con Github y Travis, además de ser gratuito con funcionalidades suficientes para cubrir las necesidades del proyecto, por lo que resulta una herramienta muy recomendable para aquellos desarrolladores que quieran tener sus resultados Online y desarrollen, entre otras plataformas, en Node.js. Se puede acceder al proyecto desplegado en Heroku en: <https://dreamstillapp.herokuapp.com>

## 2.4. Resumen de las tecnologías

Para concluir éste apartado y a modo de resumen, con la finalidad de mostrar las tecnologías usadas de un formato más visual, se ha incluido ésta sección, en la que representaremos gráficamente las tecnologías usadas, en que parte se han usado y cuál es su finalidad, para que el lector tenga la capacidad de reconocerlas e identificarlas de un vistazo y tenga un esquema visual de las mismas y su implicación en el proyecto.

### 2.4.1. Resumen Tecnologías de desarrollo

En ésta sección mostraremos las tecnologías que se han usado para el desarrollo del proyecto clasificándolas en función del ámbito al que pertenecen cada una de ellas.

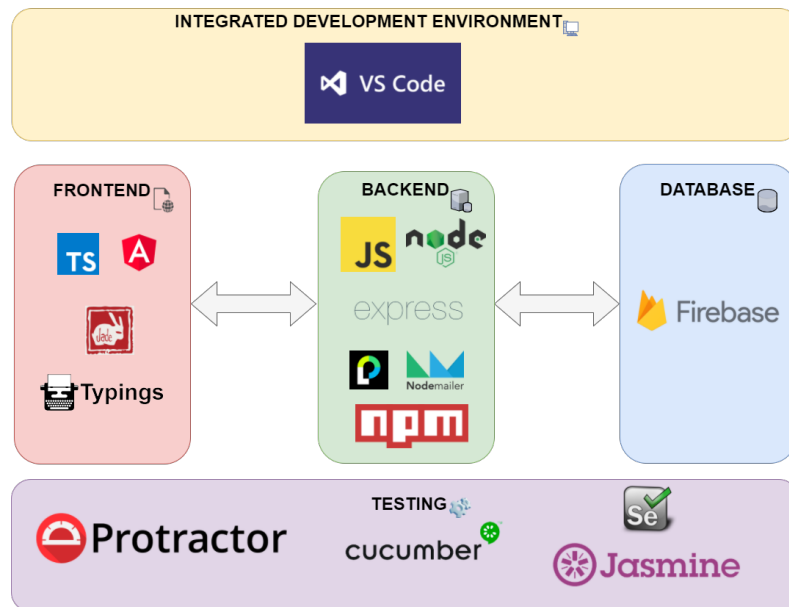


Figura 2.1: Resumen Tecnologías de Desarrollo

### 2.4.2. Resumen Tecnologías de planificación y gestión del proyecto

Las tecnologías que se han usado para generar la documentación y la gestión del proyecto son las que a continuación se muestra, mostrando gráficamente en que se centran cada una de ellas.

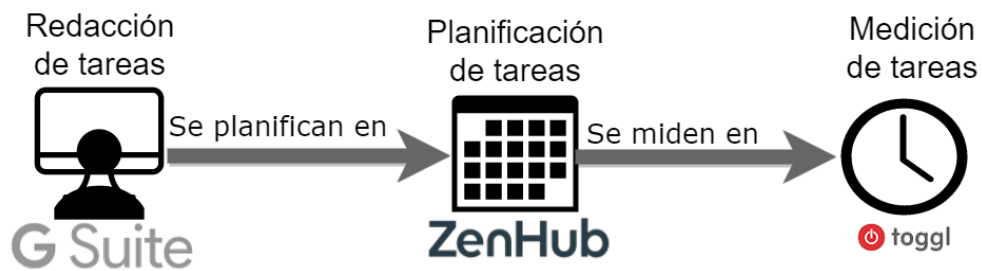


Figura 2.2: Resumen Tecnologías de planificación y gestión del proyecto

### 2.4.3. Resumen Tecnologías de gestión del código, integración y despliegue continuo

Éstas tecnologías se encuentran todas en la “nube” por lo que sus resultados se encuentran al alcance de cualquier usuario que desea revisarlos, es por ello la representación que se le ha dado en éste apartado.

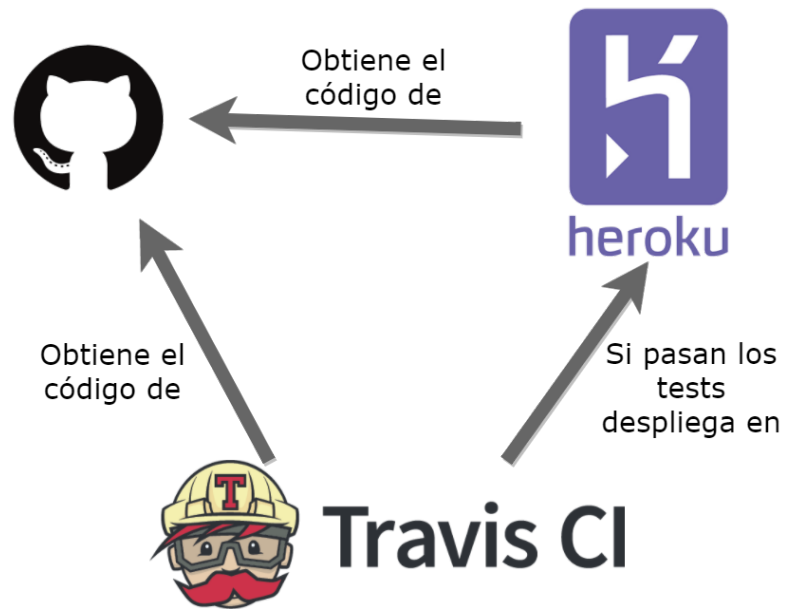


Figura 2.3: Resumen Tecnologías de gestión del código, integración y despliegue continuo

## Capítulo 3

# Análisis y Diseño

Dividiremos el capítulo en secciones con los distintos diagramas que constituyen el análisis y el diseño de la aplicación. Mostraremos además en éste capítulo los mockups del proyecto, cabe destacar que pueden diferir en algunos casos, ya que los dichos mockups se realizaron al comienzo de la aplicación, como se mostrará en el capítulo 4 y la aplicación al seguir un procedimiento de ágil, ha sufrido cambio respecto a los requisitos planteados inicialmente.

### 3.1. Análisis

#### 3.1.1. Backlog del producto

A continuación se detallará el Backlog del producto. Éste contiene los requisitos y el Sprint en el que han de ser implementados, cabe destacar que en algunos casos los requisitos se han dividido en “Fases”, ésto es debido a la estimación seguida, por la que en el caso de estimar un requisito demasiado grande, se divide en trozos más pequeños, con la filosofía “Divide y vencerás” y con la finalidad de poder implementar dicho requisito con éxito.

Cuadro 3.1: Backlog del Producto

ID	Nombre	Iteración / Sprint
1	Montaje de la infraestructura	1
2	Instalación y elección de las Herramientas	1
3	Obtención de datos de Morpheuz	1
4	Planificación inicial	1
5	Subsistema de login	2
6	Almacenamiento de datos en Firebase	2
7	Daemon Correo y almacenamiento en Firebase	2
8	Procesamiento de datos de Morpheuz	2
9	Elegir Framework CSS	2
10	Planificación inicial	2
11	Estimar puntos de historia	3
12	Plantear historias de usuario	3
13	Login Back-End	3
14	Almacenamiento de login en Firebase	3
15	Dummy Chart con movimientos y un componente de Angular	3
16	Mockups aplicación	3
17	Incluir Angular Material	3
18	Imagen Corporativa General	4
19	Autenticación al usar Firebase	4
20	HU1: Registro en la aplicación	4
21	HU2: Loguearse en la Aplicación	4
22	Integrar el proyecto con un CI en la nube	4
23	HU3: Acceder a la página principal	4
24	HU4: Gestión de datos de perfil y validación	5
25	HU5: Representación de datos de firebase en calendario	5
26	HU6: Integración de identificación en servicios de 3 <sup>ra</sup> en los datos de perfil.	5
27	HU7: Representación detallada de datos en las gráficas conectada con el calendario.	5
28	HU8: Terminar el proceso batch para que procese todos los mails de un usuario.	5
29	Integración con la API de Fitbit - Fase 1	6
30	Integración con la API de GoogleFit - Fase 1	6
31	Actualizar historias de usuario	6
32	Integración con la API de Fitbit - Fase 2	7
33	Integración con la API de GoogleFit - Fase 2	7
34	Tests - Fase 1	7
35	Tests - Fase 2	8
36	Crear página para que un usuario pueda enlazar sus datos de Morpheuz con la App	8
37	Estudiar Integración con la API de Sleep As Android	8
38	Añadir opción para recuperar contraseña	8
39	Documentar apartado 1 - Introducción	9
40	Documentar apartado 2 - Introducción	9
41	Documentar apartado 4 - Introducción	9
42	Documentar apartado 1 - Revisión	10
43	Documentar apartado 2 - Revisión	10
44	Documentar apartado 4 - Revisión	10
Continúa en la página siguiente		

Cuadro 3.1 – continuación de la página anterior

ID	Nombre	Iteración / Sprint
45	Añadir sistema de alertas	10
46	Documentar apartado 1 - Finalización	11
47	Documentar apartado 2 - Finalización	11
48	Documentar apartado 4 - Finalización	11
49	Documentar apartado 3 - Introducción	11
50	Documentar apartado 7 - Introducción	11
51	Integración con la API de Fitbit - Fase 3	11
52	Parámetro Alertas	12
53	Corregir errores en la documentación	13
54	Documentar apartado 6 - Introducción	13
55	Documentar apartado 8 - Introducción	13

En el capítulo 4, se detallará con mayor información el Backlog que a continuación se muestra, mostrando la estimación en horas y puntos de historia que se estimaron.

### 3.1.2. Historias de Usuario

Al igual que ocurre con los Mockups mostrados anteriormente, algunas de éstas han sido actualizadas o incluso se han añadido más funcionalidades que las definidas en el diseño de las mismas, ya que éstas se diseñaron al inicio de la aplicación. Sin embargo, como se ha podido comprobar en el Backlog, se han implementado en varios Sprints, por lo que la variación final sufrida respecto a el diseño inicial no ha sido tan notable como en el caso de los Mockups.

Seguidamente pasamos a la definición de las Historias de Usuario:

- HU1:

**ID:**1      **Nombre:** Registro en la aplicación.  
**Descripción:** *Como* usuario quiero poder registrarme en la aplicación *para* poder acceder a la aplicación al loguearme con los datos del registro.  
**Pruebas de aceptación:**

- Registrarse con los datos correctos y comprobar que efectivamente se realiza con éxito el registro.
- Revisar que al introducir datos erróneos la aplicación avisa de los fallos en esos datos.

- HU2:

**ID:2    Nombre:** Loguearse en la aplicación.

**Descripción:** *Como* usuario quiero poder loguearme en la aplicación con los datos de mi registro *para* poder acceder a mis datos en la aplicación.

**Pruebas de aceptación:**

- Ingresar los datos de Login correctos y comprobar que se da acceso al usuario a la aplicación.
- Revisar que al introducir datos erróneos la aplicación avisa de que los datos introducidos no corresponden con los de ningún usuario registrado.

- HU3:

**ID:3    Nombre:** Acceder a la página principal.

**Descripción:** *Como* usuario quiero poder acceder a la página principal de la aplicación *para* poder ver mi calendario y seleccionar los días e informes que necesite.

**Pruebas de aceptación:**

- Comprobar que los datos que se le muestran al usuario son los pertenecientes a ese usuario.
- Revisar que todos los enlaces funcionan correctamente y que no hay ningún error a la hora de extraer los datos de la base de datos.

- HU4:

**ID:4    Nombre:** Gestión de datos de perfil y validación.

**Descripción:** *Como* usuario quiero que mis datos sean validados y gestionados en el formulario de Login y registro *para* verificar que la veracidad de los datos introducidos y ser informado en el caso de que ocurra cualquier error.

**Pruebas de aceptación:**

- Comprobar que los datos introducidos se procesan con éxito.
- Revisar la validación de datos al introducir datos erróneos.



- HU5:

**ID:5 Nombre:** Representación de datos de Firebase en el calendario.

**Descripción:** *Como* usuario quiero que ver reflejados que días del calendario contienen información acerca de mis datos de sueño *para* acceder a la representación de los datos de ese día.

**Pruebas de aceptación:**

- Comprobar que los días que contienen datos de sueño se marcan en el calendario como un evento.

- HU6:

**ID:6 Nombre:** Integración de identificación en servicios de 3<sup>o</sup>s en los datos de perfil.

**Descripción:** *Como* usuario quiero poder integrar servicios de aplicaciones de terceros en la aplicación *para* añadir información de éstos y procesar dicha información.

**Pruebas de aceptación:**

- Comprobar que la integración con éstos servicios se hace correctamente.
- Verificar la correcta identificación en dichos servicios.

- HU7:

**ID:7 Nombre:** Acceder a la página de “Resumen de Sueño” para un día concreto.

**Descripción:** *Como* usuario quiero poder acceder a la página de “Resumen de Sueño” de un día concreto en el que haya registrado datos *para* obtener un informe de los distintos valores del sueño relacionado con ese día.

**Pruebas de aceptación:**

- Comprobar que los datos que se le muestran al usuario son los pertenecientes a ese usuario.
- Revisar que no hay ningún error a la hora de extraer los datos de la base de datos.

- HU8:

**ID:8 Nombre:** Terminar el proceso “batch” para que procese todos los mails de un usuario.

**Descripción:** *Como* usuario quiero que la aplicación cuente con un proceso automático de procesamiento de emails *para* que en el caso de que la aplicación se caiga, se procesen los emails pendientes al reiniciarla.

**Pruebas de aceptación:**

- Comprobar que se procesan los email pendientes correctamente.
  
  
  
  
  
  
  
  
  
- Verificar que el proceso se ejecuta cada vez que se inicia la aplicación.

### 3.1.3. Mockups

A continuación se mostrarán los Mockups diseñados para la aplicación. Cabe destacar fueron diseñados al principio de la misma y que por tanto han sufrido en algunas ocasiones importantes cambios respecto a la implantación que se ha llevado a cabo. Además, algunas vistas se han añadido con posterioridad a los mismos, por lo que no se encuentran representadas por éstos Mockups, los cuales representan la funcionalidad básica de la aplicación.

## Página de Login y Registro

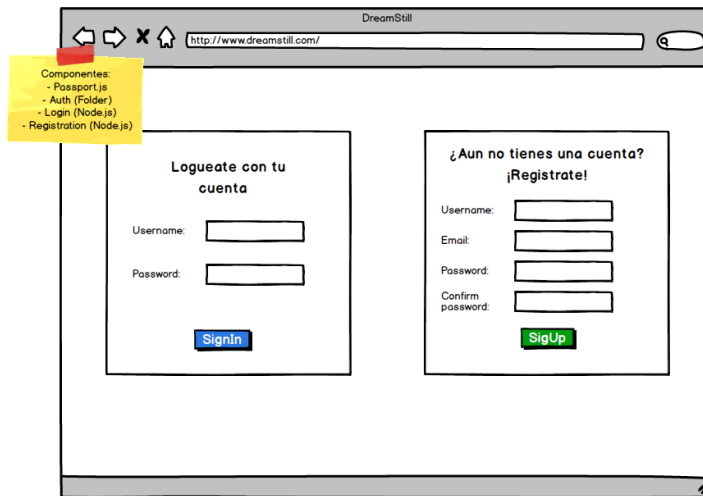


Figura 3.1: Mockup Página de Login y Registro

Se desea hacer el acceso al usuario lo más sencillo posible, tanto para los usuarios que ya son de la aplicación, como los que estarían interesados en serlo. Es por ello que el formulario de registro esté al mismo nivel que el de login, de ésta forma, se pretende que el esfuerzo a realizar entre un usuario que ya sea de la aplicación y uno que no lo sea sea similar, animando así a los nuevos usuarios a empezar en la aplicación.

Además, al estar el formulario de registro visible en todo momento, evitamos ése posible miedo por parte de los usuarios a pensar que tendrán que rellenar una gran cantidad de datos o que el registro se hará pesado, ya que como se puede comprobar, sólo deberá de rellenar algunos datos básicos para acceder a la misma.

## Página principal

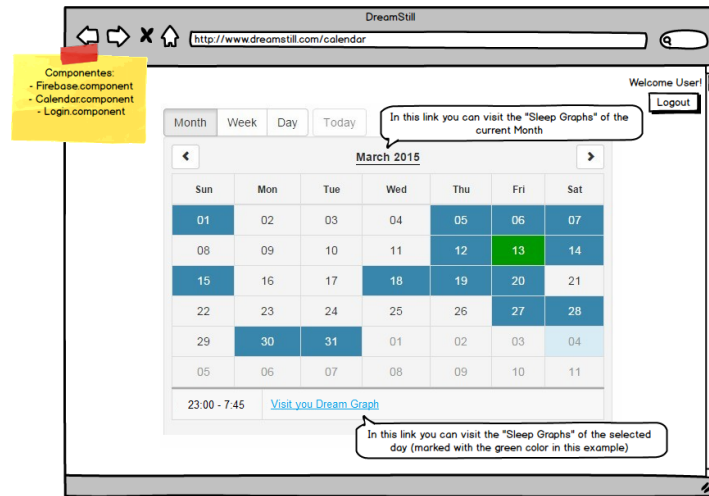


Figura 3.2: Mockup Página Principal

Ésta sería la vista principal que todo usuario que se loguease encontraría nada más entrar. En ella, se encontraría con un calendario en el que se representarían los eventos de sueño y sobre el que podría navegar entre los distintos meses del mismo. Además, el usuario tendrá acceso desde los eventos a la página que contendrá la información de los mismos.

Ésta vista es la que posiblemente haya sufrido uno de los mayores cambios respecto al Mockup planificado. Éstos cambios han sido motivados por el cambio a una interfaz más simple, en la que eliminar posibles menús y aunarlos todos en un mismo botón, que es el que encontramos actualmente en la aplicación y que contiene todas las rutas necesarias para acceder a las distintas partes de la aplicación. De ésta forma, queda una vista mucho más limpia y simple, en la que el usuario podrá centrarse en el calendario y en sus eventos sin distracción alguna.

## Página de Resumen de Sueño Diario

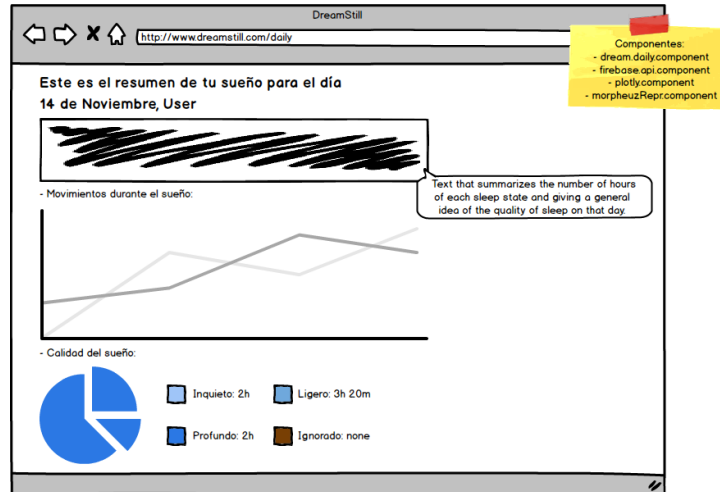


Figura 3.3: Mockup Página de Resumen de Sueño Diario

Se compone de dos gráficas. En la primera de ellas el usuario podrá observar como ha sido su estado de sueño en cada intervalo de tiempo de la noche, conociendo cuales fueron los periodos de mayor actividad y en cuales el usuario estuvo prácticamente inmóvil. En la segunda, se muestra de forma resumida la calidad de sueño que se ha tenido, en función de la actividad de ese usuario durante el intervalo de sueño.

El objetivo de éstas dos gráficas es que el usuario quede informado de su sueño rápidamente y pueda ahondar en más detalles si lo desea interactuando con las mismas.

## 3.2. Diseño

### 3.2.1. Diagrama de Componentes

El siguiente diagrama pretende mostrar cómo está compuesta la aplicación, cuales son sus módulos y cual es la finalidad de cada uno de ellos. Para ello, se definirán la función de cada uno de los módulos por separados, con la intención de que el lector conozca la función de cada uno de ellos.

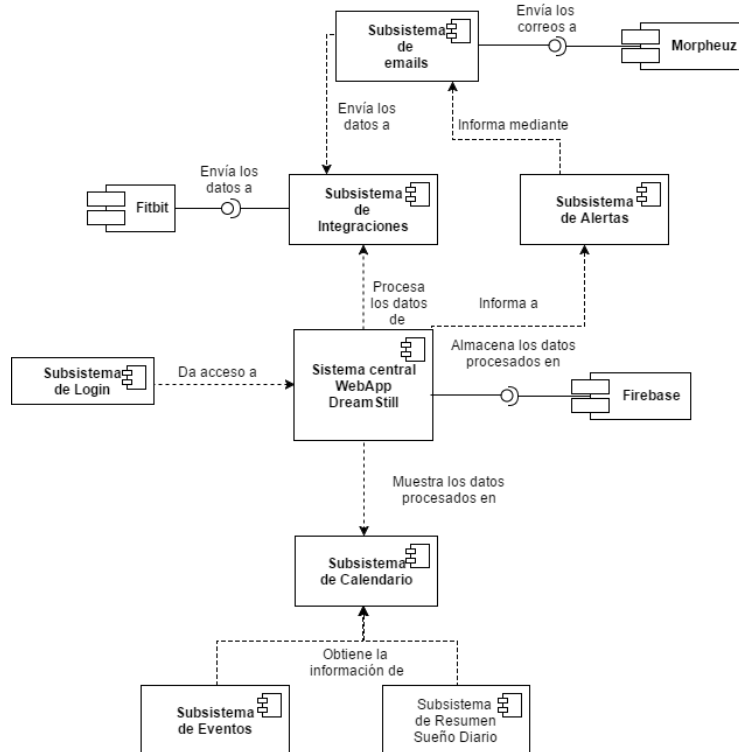


Figura 3.4: Diagrama de Componentes

- Sistema Central WebApp DreamStill: Es el sistema más importante de la aplicación. Se encarga de obtener y procesar los datos y comunicarlos al resto de subsistemas de la aplicación. En él se encontrarían las funciones más importantes de la aplicación, como la comunicación

entre el Subsistema de Integraciones y Firebase o entre éste y el Subsistema de Calendario.

- Subsistema de Integraciones: Es el subsistema encargado de abstraer al Sistema Central de las distintas integraciones que se añadan a la aplicación, gracias a éste subsistema, a pesar de las distintas integraciones que se realicen, todas serán iguales a instancias del Sistema Central. Por tanto, es necesario que en éste Subsistema se haga un tratado de los datos de las distintas APIs para procesarlos y homogeneizarlos.
- Subsistema de emails: Es el subsistema encargado de realizar las distintas funciones de correo existentes en la aplicación. Entre ellas se encuentran la lectura de los correos enviados por los usuarios de Morpheuz con sus datos de sueño a la aplicación, o el envío de correos a los usuarios que han establecido una alerta en el sistema.
- Subsistema de Alertas: Es el subsistema encargado de proporcionar al usuario la opción de crear alertas en el sistema, de tal forma que si se cumple una condición impuesta por el usuario (como por ejemplo dormir menos de 7 horas durante 3 días consecutivos) éste sea informado sin necesitar de tener que acceder a la aplicación para comprobarlo.
- Subsistema de Calendario: Es el subsistema encargado de proveer y mostrar el calendario que finalmente el usuario encuentra al entrar en la página principal de la aplicación. Provee las distintas funciones como pueden ser navegar al mes anterior, al siguiente o regresar a la fecha actual.
- Subsistema de Login: Es el sistema encargado de la autenticación de los usuarios. Se encarga de comprobar a través de la información que le proporciona el Sistema Central las correctas credenciales de los usuarios y de dar acceso a los mismos a la aplicación. Además, se encarga del registro de nuevos usuarios y de la recuperación de la contraseña de los mismos.
- Subsistema de Eventos Calendario: Es el subsistema encargado de proveer los eventos que se mostrarán en el calendario de la aplicación, de tal forma que un usuario pueda ver en el calendario en qué días tiene eventos de sueño.
- Subsistema de Vista Resumen Sueño Diario: Es la vista encargada de mostrar los datos representados gráficamente de una etapa de sueño

concreta. De ésta forma, un usuario puede acceder visualmente a éstos datos y obtener reflexiones interesantes de los mismos.

- **Firebase:** Es el sistema encargado de comunicarse, leer y almacenar todos los datos en Firebase, nuestra base de datos.
- **Fitbit:** Es el subsistema encargado de obtener los datos de Fitbit y transmitirlos al Subsistema de Integraciones para que los procese.
- **Morpheus:** Es el subsistema encargado de controlar las cuentas de Morpheuz de los usuarios y enviar los correos con los datos necesarios al Subsistema de email.

### 3.2.2. Diagrama de Despliegue

Describe la estructura a nivel de hardware de nuestra aplicación. Pretende mostrar de manera gráfica la estructuración de la aplicación dividida en los distintos niveles hardware que la componen. Para nuestro caso concreto estará estructurado en tres niveles.

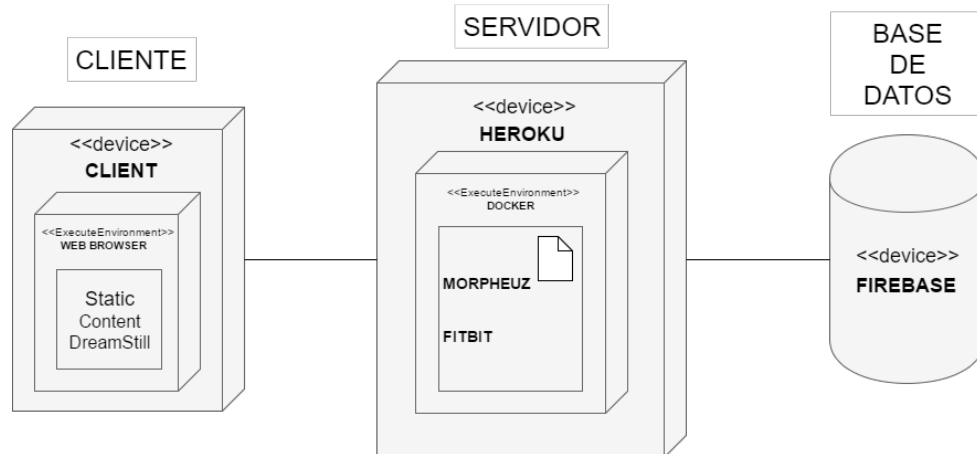


Figura 3.5: Diagrama de Despliegue

El primero de ellos, pertenece a la parte del cliente. El entorno de ejecución será para nuestro caso cualquier dispositivo con acceso a la



aplicación web, éste a su vez ejecutará el contenido de nuestra aplicación perteneciente al lado del cliente, y provocará que se ejecuten los contenidos necesarios del servidor para poder realizar acciones en nuestra aplicación.

Por otro lado, tenemos la parte del servidor. Para nuestro caso concreto, se alojará en los servidores de Heroku que es donde está alojado nuestra aplicación, y el entorno de ejecución será Docker, el cual gracias a sus contenedores provee mayor escalabilidad en la aplicación. Para poder ejecutar el contenido y las funcionalidades de nuestra aplicación, implementaremos la interfaz de Fitbit y la lectura de emails de Morpheuz.

Por último, tenemos la parte de base de datos. Para nuestro caso Firebase, base de datos en tiempo real que nos permitirá tener nuestros datos en la “nube” y tenerlos actualizados de manera continua, además de ofrecernos la posibilidad de tener un fácil acceso a ellos y además de forma segura.

### 3.2.3. Diagrama de Capas y Tecnologías

Se pretende mostrar al lector como están estructuradas las capas de la aplicación. En cada capa se incluirá además las tecnologías que se han usado en cada una de ellas, con la finalidad de diferenciarlas y conocer sus usos.

Para su estructura se ha seguido el modelo Vista/Controlador [?], el cual se divide en tres capas. La primera de ellas, la de persistencia de datos, en la que se sitúa el sistema de almacenamiento de datos y el sistema gestor de almacenamiento de datos si lo hubiera. En la segunda de ellas, se sitúa el controlador, el cual se define como “el director de orquesta” ya que orquesta las distintas llamadas al servidor y proporciona las funcionalidades necesarios al resto de capas. En la tercera de ellas, se sitúa la vista, la cual proporciona los datos del lado del cliente, es decir, los datos que se representan en el navegador del usuario, obteniéndolos del controlador.

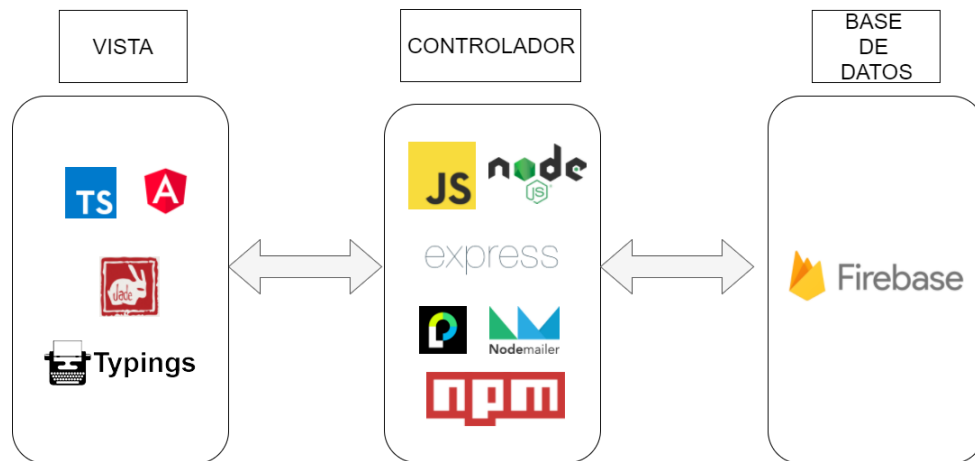


Figura 3.6: Diagrama de Capas y Tecnologías

### 3.2.4. Diagrama UML

Describe el diagrama de clases (Diagrama UML [?]) seguido en el diseño del esquema de la Base de Datos. A pesar de utilizar Firebase y de ser éste una Base de Datos NoSQL, el cual permite una estructura que puede ser propensa a cambios, se ha seguido una estructura prefijada, ya que así se consigue una correcta gestión de la Base de Datos.

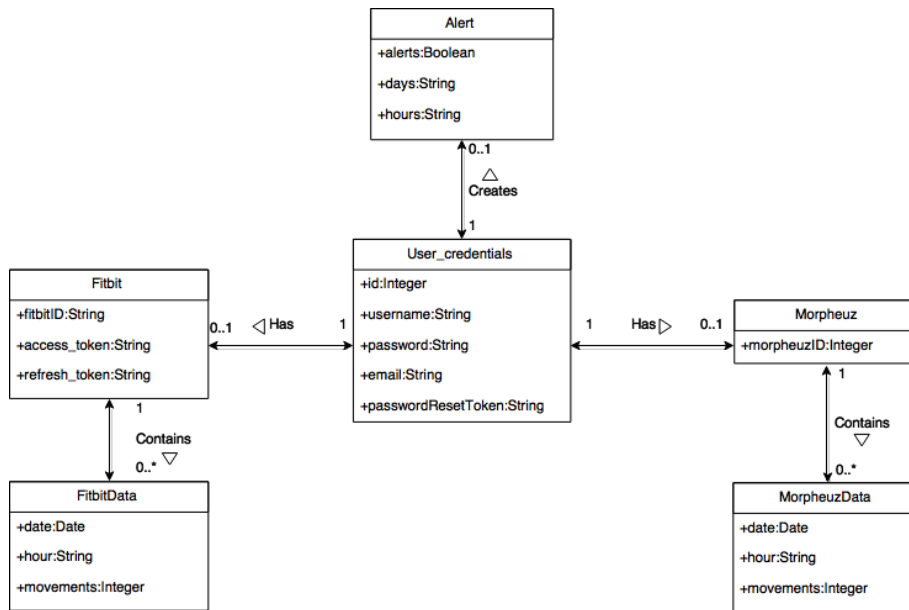


Figura 3.7: Diagrama de Clases

Sus clases son las siguientes:

- **User\_Credentials**: Clase que almacena los datos básicos de un usuario.
- **Alert**: Clase que representa si un usuario tiene el sistema de alertas activados y los parámetros que ha introducido para dicho sistema.
- **Morpheus**: Clase que almacena el ID de Morpheuz de un usuario y relaciona al mismo con sus datos en dicha aplicación.
- **MorpheusData**: Clase que almacena los datos de movimiento para una hora y fecha concreta obtenidos mediante la aplicación Morpheuz.

- Fitbit: Clase que almacena los datos necesarios para obtener los datos de la API de Fitbit de un usuario y relaciona al mismo con sus datos en dicha aplicación.
- FitbitData: Clase que almacena los datos de movimiento para una hora y fecha concreta obtenidos mediante la aplicación Fitbit.

## Capítulo 4

# Planificación

Para la planificación del proyecto se ha seguido una metodología ágil, en la que cada iteración del proyecto se comprendía en un Sprint. Además, se han desarrollado Historias de Usuario y Mockups, se ha creado un Backlog que contiene todos los requisitos a implementar y se ha creado un fichero de planificación con los requisitos que se implementarán en cada semana durante el transcurso del proyecto.

Para llevar el seguimiento de la planificación y calcular posibles desvíos o estimar la duración de ciertas tareas, se han utilizados herramientas destinadas a éste fin. Éstas herramientas nos han proporcionado datos útiles y han conseguido que se observase de forma clara como avanzaba el proyecto en progreso y en tiempo.

Respecto a los requisitos, al seguir una metodología ágil, partimos de una idea inicial, y los requisitos se fueron detallando y añadiendo en cada una de las reuniones que manteníamos mi tutor y yo, éstas reuniones ocurrían al finalizar cada Sprint y suponían el inicio del siguiente, con unos requisitos definidos para dicho Sprint y a partir de los cuales se realizaba la estimación de los mismos. Se ha seguido una metodología SCRUM [?] adaptada a nuestro proyecto, ésta metodología, es una metodología ágil y consiste en la división de trabajos en “Sprints” que son periodos de tiempo en los que se planifican una serie de tareas que implican un incremento en el proyecto [?], ésta metodología es muy usada en proyectos en los que se definen los requisitos durante el transcurso del proyecto y tiene como

beneficios un mayor seguimiento de los entregables por parte del cliente, llamado en ésta metodología “Product Owner”, el cual tendrá una reunión en cada Sprint para conocer como avanza el proyecto y aprobar las tareas para cada uno de éstos Sprints. Por motivos de tiempo ambas partes no podíamos llevar a cabo una “Daily meeting” como se recomienda en dicha metodología, por tanto la metodología de Scrum que hemos seguido sería la que se aprecia en la siguiente figura:

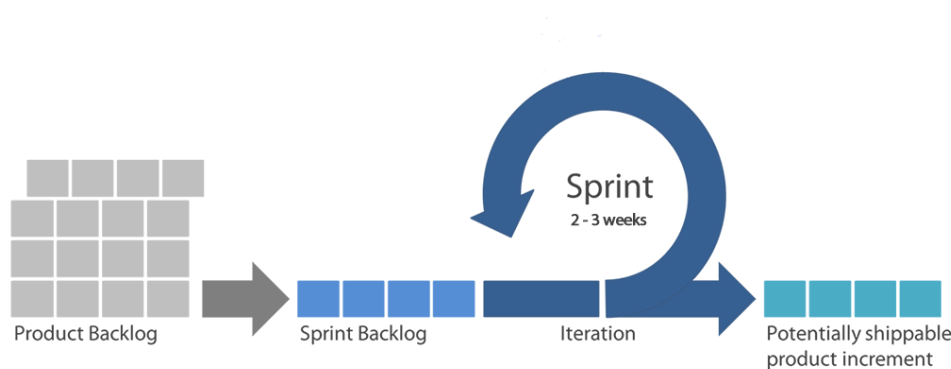


Figura 4.1: Metodología Scrum

Respecto a los Sprints, éstos solían ser de dos semanas. Aunque su duración podía variar, dependiendo del momento o fechas que nos encontráramos (casos especiales como Navidad por ejemplo), o de la carga de trabajo que se pretendiese para dicho Sprint. La decisión de ésta duración de dos semanas fue consensuada, ya que llegamos a la conclusión que era un intervalo de tiempo suficientemente amplio como para realizar una buena iteración de cambios pero no tan amplio como para que el proyecto no avanzase a la velocidad que debiera al no poder estimar o corregir a tiempo los errores en la planificación.

El Backlog, ha sido el documento que se incrementaba al final de cada reunión, ya que en el se acordaban y apuntaban los requisitos que se implementarían en el siguiente Sprint, ha habido casos excepcionales en el que algunos requisitos no han llegado a ser implementados debido a que se apuntaron en éste pero por cambios de rumbo en la aplicación o por la

re-estructuración de ésta no han llegado a tal implementación. Sin embargo, éstos han sido casos aislados, lo habitual es concretar unos requisitos en la reunión, apuntarlos en el Backlog y estimarlos posteriormente. Es por ello que en el Backlog como mostraremos a continuación, se muestran todas los incrementos que ha sufrido el proyecto junto con el Sprint en el que se han producido y los puntos de historia que se le estimaron.

Cuadro 4.1: Backlog del Producto

ID	Nombre	Estimación	Iteración / Sprint	Puntos de historia
1	Montaje de la infraestructura	10	1	13
2	Instalación y elección de las Herramientas	3	1	2
3	Obtención de datos de Morpheuz	1,5	1	1
4	Planificación inicial	2	1	1
5	Subsistema de login	5	2	3
6	Almacenamiento de datos en Firebase	8	2	13
7	Daemon Correo y almacenamiento en Firebase	10	2	21
8	Procesamiento de datos de Morpheuz	6	2	5
9	Elegir Framework CSS	2,5	2	1
10	Planificación inicial	0,5	2	1
11	Estimar puntos de historia	1,5	3	2
12	Plantear historias de usuario	2	3	3
13	Login Back-End	4	3	5
14	Almacenamiento de login en Firebase	2	3	5
15	Dummy Chart con movimientos y un componente de Angular	8	3	8
16	Mockups aplicación	2	3	3
17	Incluir Angular Material	2	3	5
18	Imagen Corporativa General	5	4	8
19	Autenticación al usar Firebase	4	4	5
20	HU1: Registro en la aplicación	3	4	5
21	HU2: Loguearse en la Aplicación	1,5	4	3
22	Integrar el proyecto con un CI en la nube	5	4	13
23	HU3: Acceder a la página principal	3,5	4	5
24	HU4: Gestión de datos de perfil y validación	3	5	5
25	HU5: Representación de datos de firebase en calendario	5	5	8
26	HU6: Integración de identificación en servicios de 3ºs en los datos de perfil.	4	5	8
27	HU7: Representación detallada de datos en las gráficas conectada con el calendario.	6	5	13
28	HU8: Terminar el proceso batch para que procese todos los mails de un usuario.	4	5	8
29	Integración con la API de Fitbit - Fase 1	12	6	21
30	Integración con la API de GoogleFit - Fase 1	10	6	21
31	Actualizar historias de usuario	1,5	6	3
32	Integración con la API de Fitbit - Fase 2	7	7	13
33	Integración con la API de GoogleFit - Fase 2	6	7	13
34	Tests - Fase 1	15	7	21
35	Tests - Fase 2	15	8	21
36	Crear página para que un usuario pueda enlazar sus datos de Morpheuz con la App	3	8	5
37	Estudiar Integración con la API de Sleep As Android	5	8	8
38	Añadir opción para recuperar contraseña	8	8	5
39	Documentar apartado 1 - Introducción	7	9	13
40	Documentar apartado 2 - Introducción	10	9	21
41	Documentar apartado 4 - Introducción	6	9	13
42	Documentar apartado 1 - Revisión	4	10	8
43	Documentar apartado 2 - Revisión	2	10	5
44	Documentar apartado 4 - Revisión	4	10	8
45	Añadir sistema de alertas	8	10	13
46	Documentar apartado 1 - Finalización	4	11	8
47	Documentar apartado 2 - Finalización	2	11	5
48	Documentar apartado 4 - Finalización	3	11	5
49	Documentar apartado 3 - Introducción	9	11	21
50	Documentar apartado 7 - Introducción	9	11	21
51	Integración con la API de Fitbit - Fase 3	6	11	13
52	Parámetro Alertas	7	12	8
53	Corregir errores en la documentación	3	13	5
Continúa en la página siguiente				

Cuadro 4.1 – continuación de la página anterior

ID	Nombre	Estimación	Iteración / Sprint	Puntos de historia
54	Documentar apartado 6 - Introducción	9	13	21
55	Documentar apartado 8 - Introducción	9	13	21

Para la planificación de las semanas que ocurrirían durante la duración del proyecto se creo un documento con la finalidad de almacenar la información sobre la planificación de éstas. Dicho documento consta de 4 columnas, la primera muestra la semana a la que nos referimos indicando la fecha de inicio de la misma. La segunda, muestra lo que se estimó hacer durante esa semana. La tercera columna muestra una planificación inicial que se hizo al principio del proyecto, en la que yo, estimé cuales serían el número de horas que supuse que tendría disponible durante esas semanas. Por último, nos encontramos con una cuarta columna, en la que se muestra el número de horas reales que se le dedicaron al proyecto en esa semana.

Cuadro 4.2: Backlog del Producto

Semana	Tareas	Previsión	Realidad
10-10-2016	Instalación y Elección de las Herramientas, Planificación inicial	5,00	8,55
17-10-2016	Montaje de la infraestructura, Obtención de datos de Morpheuz	11,50	3,70
24-10-2016	Planificación inicial, Subsistema de login, Almacenamiento de datos en Firebase	12,50	8,40
31-10-2016	Daemon Correo y almacenamiento en Firebase, Procesamiento de datos de Morpheuz, Elegir Framework CSS	18,50	6,40
07-11-2016	Estimar puntos de historia, Plantear historias de usuario, Login Back-End, Almacenamiento de login en Firebase	11,50	6,80
14-11-2016	Dummy Chart con movimientos y un componente de Angular, Mockups aplicación, Incluir Angular Material	12,00	6,50
21-11-2016	Imagen Corporativa General, Autenticación al usar Firebase, HU1: Registro en la aplicación	10,00	5,14
28-11-2016	HU2: Login, Integración con un CI, HU3: Acceder a la página principal	8,00	7,52
05-12-2016	Vectorización del logo, HU4: Gestión de datos de perfil y validación, HU5: Representación de datos de Firebase en calendario	5,00	4,75
12-12-2016	HU6: Integración de identificación en servicios de 3 <sup>os</sup> , HU7: Representación detallada de dato, HU8: Terminar el proceso batch.	10,00	7,25
19-12-2016	Actualizar historias de usuario	8,00	1,00
26-12-2016	Integración con la API de GoogleFit	5,00	5,45
02-01-2017	Integración con la API de Fitbit	5,00	3,00
09-01-2017	Finalización de la Integración con las APIs de GoogleFit y Fitbit	10,00	7,40
16-01-2017	Preparar tests aplicación y crear test de prueba en la página principal	12,00	11,95
23-01-2017	Página Morpheuz	12,00	8,76
30-01-2017	Sleep As Android, Tests	8,00	8,94
06-02-2017	Añadir opción para recuperar contraseña	8,00	9,73
13-02-2017	Documentación apartado 1	12,00	7,62
20-02-2017	Documentación apartado 2 y 4	8,00	6,04
27-02-2017	Revisión apartados 1 y 2	8,00	8,05
06-03-2017	Revisión apartado 4 y sistema de notificaciones	12,00	7,04
13-03-2017	Finalizar apartados 1, 2 y 4	12,00	7,35
20-03-2017	Documentación apartados 3 y 7	8,00	7,09
27-03-2017	Integración con la API de Fitbit	8,00	5,60
03-04-2017	Parámetro Alertas	12,00	6,48
10-04-2017	Parámetro Alertas	5,00	0,27
17-04-2017	Corregir errores en la documentación	8,00	3,23
24-04-2017	Corregir errores en la documentación	6,00	2,05
01-05-2017	Documentación Apartado 6 y 8	5,00	
08-05-2017		12,00	



15-05-2017		10,00	
22-05-2017		12,00	
29-05-2017		12,00	
05-06-2017		8,00	
12-06-2017		8,00	
TOTAL	240	338,00	182,06

Para conocer una información más detallada de en qué actividades se han invertido tiempo en el proyecto, nos hemos valido de los informes de Toggl, los cuales, como se mostrará a continuación, detallan de una forma fidedigna en qué se ha invertido el tiempo durante el transcurso del proyecto.

Estos informes pretenden mostrar de una forma más clara el impacto en tiempo que ha tenido cada tarea del proyecto, y la continuidad a la hora de desarrollar el proyecto, ya que salvo en pocas excepciones, el trabajo ha sido continuado durante las semanas que ha transcurrido el proyecto.

Con el fin de ayudarnos con el desarrollo de la metodología ágil, y dado que como se ha comentado en capítulo anteriores hemos utilizado Github para la gestión del código, se ha usado una extensión del mismo llamado Zenhub, de la que también se habló en capítulos anteriores. Gracias a dicha herramienta hemos contado con un tablero Kanban:

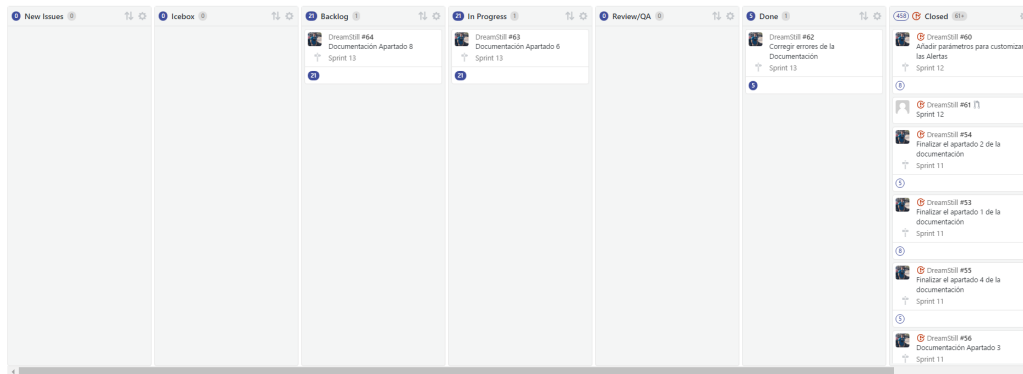


Figura 4.2: Tablero Kanban (Zenhub)

En dicho tablero se apuntan las tareas de cada uno de los Sprints y en el estado en el que se encuentran. En el se han ido registrando los cambios de las tareas (Issues en Github) durante el transcurso del Sprint correspondiente. Además, dicha herramienta, al proporcionarnos una opción para configurar el número de puntos de historia de una tarea, nos realizaba distintos gráficos para conocer el estado del proyecto durante el Sprint correspondiente u otros para ofrecernos una visión global del proyecto. Como se muestra a continuación, durante el transcurso de un Sprint, se realizaban automáticamente gracias a esta herramienta, un gráfico Burndown, el cual mostraba (en gris en el gráfico) un desarrollo ideal de como debían de ir completándose los puntos de historia y a continuación (en azul en el gráfico) mostraba el desarrollo real de dichos puntos de historia en función a como avanzaban las tareas en el tablero Kanban, considerándose como cerradas aquellas tareas que se movían al estado de “Done”:



Figura 4.3: Gráfico Burndown (Zenhub)

Además de éstos Burndown, la herramienta Zenhub almacenaba los puntos de historia estimados y completados en cada Sprint, y representaba un gráfico en el que se mostraba el número de puntos de historia completados en cada Sprint y la velocidad media a la que se implementaban.

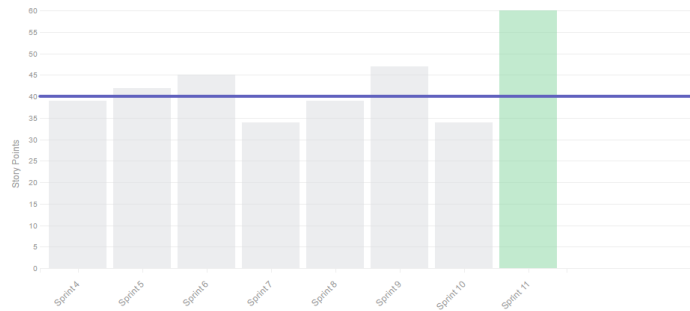


Figura 4.4: Gráfico Velocity Tracking (Zenhub)

## 4.1. Detalles de la planificcaión

En ésta seccion de la planificación se va a detallar el trabajo que se realizó en cada uno de los Sprints a lo largo del proyecto. Además, mostraremos el Burndown de cada uno de los Sprints para analizar cuál fue la desviación respecto a la planificación inicial y se mostrarán los informes de cada uno de los Sprints proporcionados por la herramienta Toggl, gracias a éstos reportes, podremos ver el tiempo que se le dedicó a cada tarea.

#### 4.1.1. Sprint 1

A lo largo del primer Sprint, se desarrollaron las tareas iniciales del proyecto. Entre ellas, se realizó el montaje de la infraestructura, la cual incluía la creación del proyecto en la plataforma Github, la generación de una aplicación básica y la realización de alguna prueba con Firebase.

Se realizó también la elección y la instalación de las herramientas de desarrollo y la primera obtención de datos de Morpheuz, para comprobar cómo era el formato de éstos y como se podían obtener. Por último en éste primer Sprint, se realizó una planificación inicial, planificando las semanas disponibles para la realización del proyecto y estimando las horas que se tendrían disponibles en cada una de ellas.

- Burndown:



Figura 4.5: Sprint 1

- Informe de Toggl:

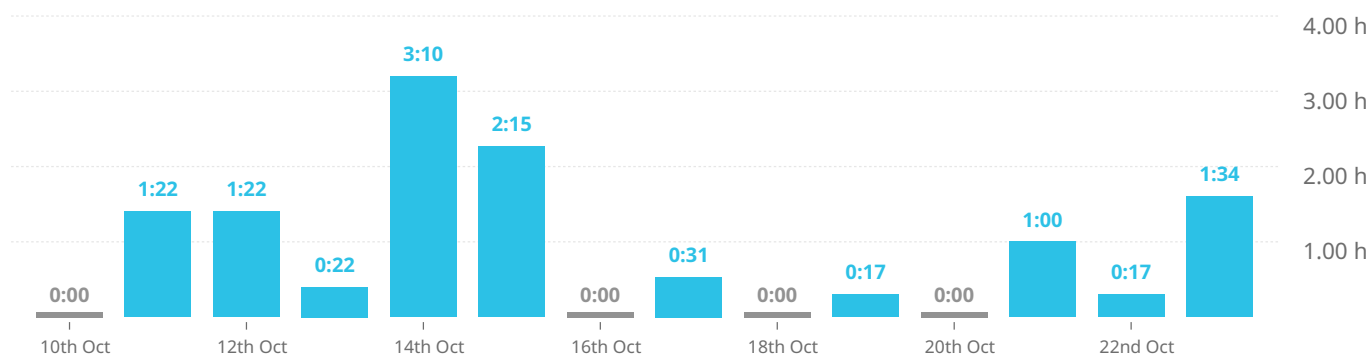
# Summary report



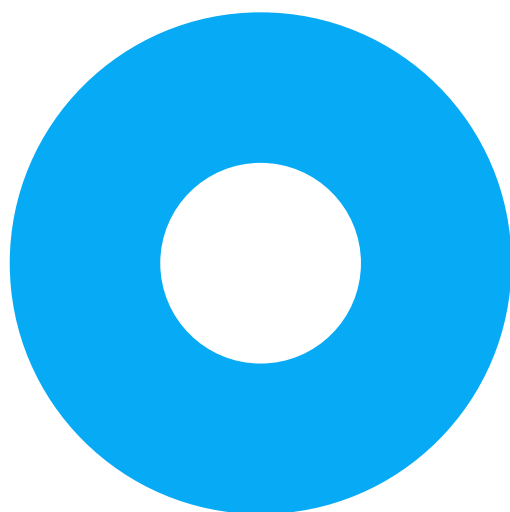
2016-10-10 - 2016-10-23

Total 12 h 13 min

DreamStill selected as projects

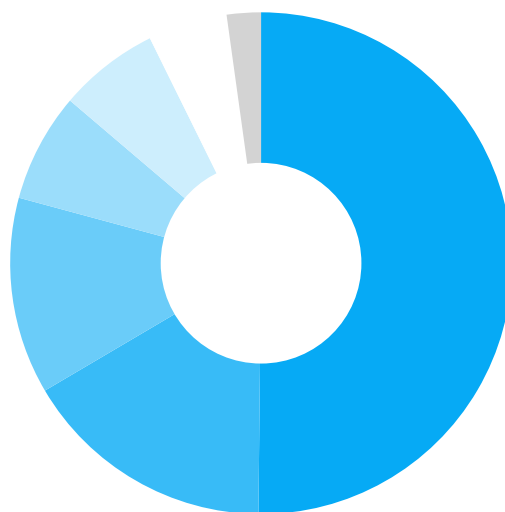



## Projects





 DreamStill 12:13:39


## Time entries





 Montaje de la Infraestructura 6:08:03


 Reunión TFG 2:00:00

 #3 Subsistema de login 1:33:01

 Obtención de datos de Morpheuz 0:51:31

 Elección del IDE 0:47:28

 Planificación inicial 0:37:29

 Other 0:16:07

### 4.1.2. Sprint 2

A lo largo del segundo Sprint, se realizaron tareas con la finalidad de obtener la funcionalidad básica del proyecto y comprobar que las tecnologías elegidas para el proyecto eran las adecuadas. Entre éstas tareas se encuentra el subsistema de login, el primer almacenamiento de datos en Firebase y un proceso para comprobar la dirección de correo de la aplicación a la espera de nuevos correos, para obtener los datos de sueño de la aplicación Morpheuz de los usuarios de la aplicación y almacenar dichos datos en Firebase.

Sin embargo, los datos de Morpheuz debían de ser procesados antes de almacenarlos, y también se llevó a cabo éste procesamiento durante el segundo Sprint. Además, se eligió el Framework de CSS usado durante el proyecto y se terminó la planificación inicial, la cual había quedado incompleta en el primer Sprint.

- Burndown:

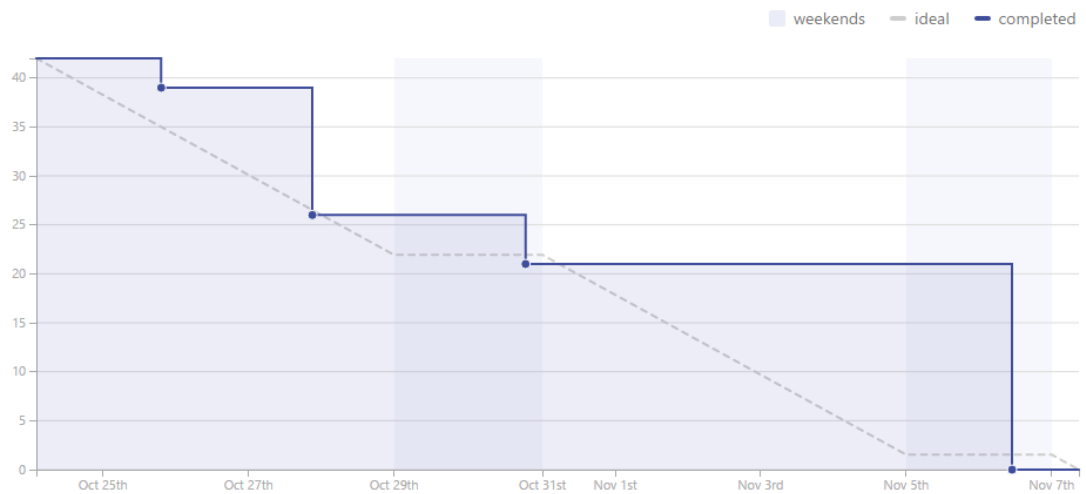


Figura 4.6: Sprint 2

- Informe de Toggl:

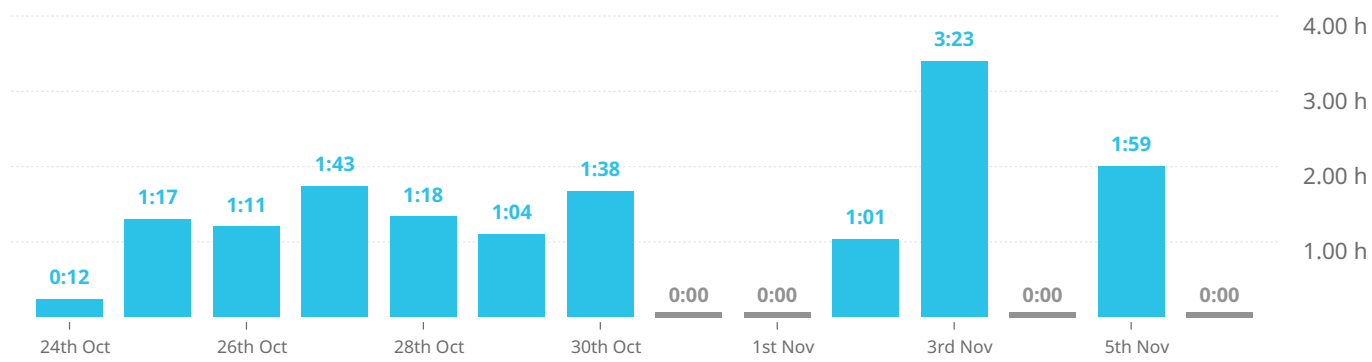
# Summary report



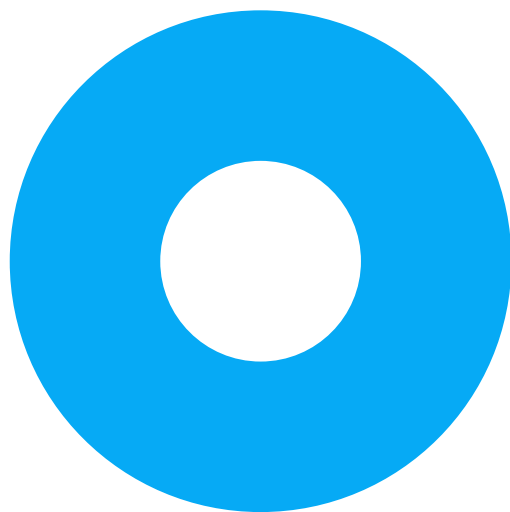
2016-10-24 - 2016-11-06

Total 14 h 51 min

DreamStill selected as projects

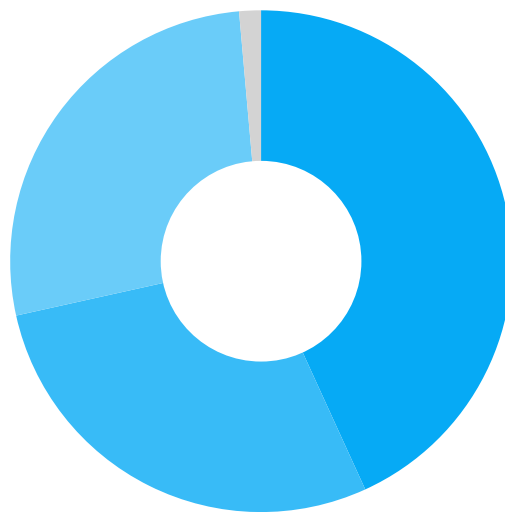






## Projects



 DreamStill 14:51:04

## Time entries



 #5 Daemon Correo y almace... 6:24:47  
 #4 Almacenamiento de dato... 4:12:31  
 #6 Procesamiento de datos... 4:01:17  
 Other 0:12:29

### 4.1.3. Sprint 3

A lo largo del tercer Sprint, se estimó los puntos de historia de las tareas anteriores y de las siguientes en éste sprint y se plantearon las historias de usuario. Además, se realizó la parte de login en el servidor, se conectó dicha parte con Firebase para almacenar los datos de acceso de los usuarios y se creó un primer gráfico representando los datos de Morpheuz.

Para terminar, en éste Sprint se diseñaron los Mockups y se incluyó el Framework de CSS elegido, que en el caso de nuestro proyecto fue “Angular Material”.

- Burndown:

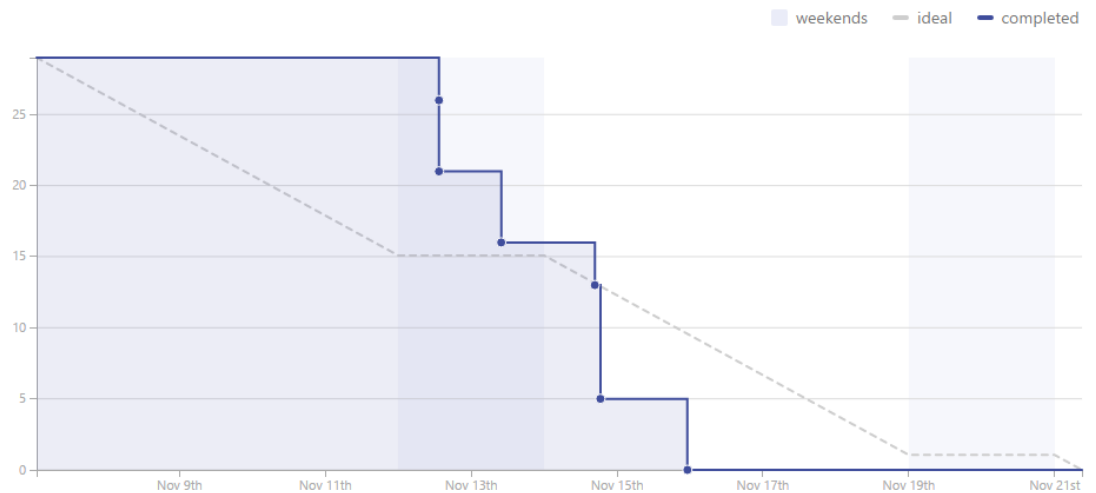


Figura 4.7: Sprint 3

- Informe de Toggl:



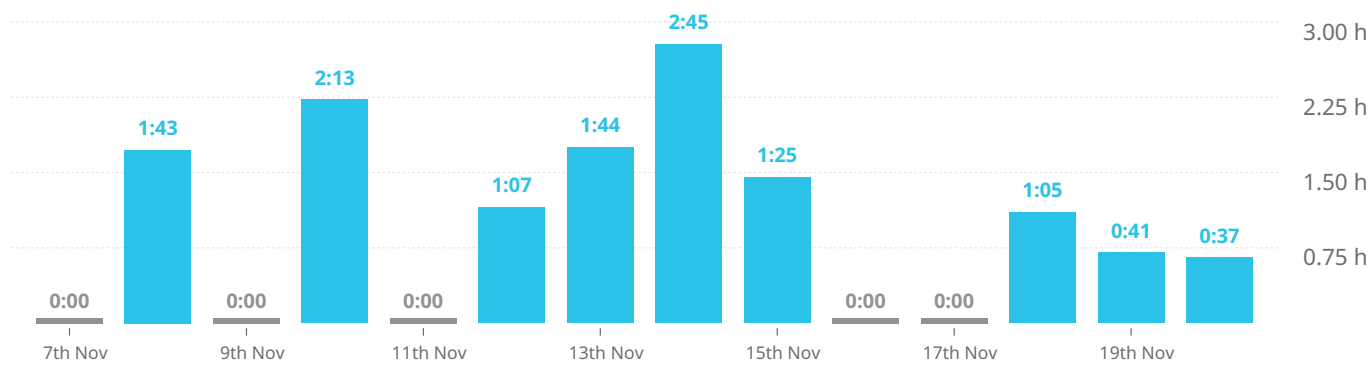
# Summary report



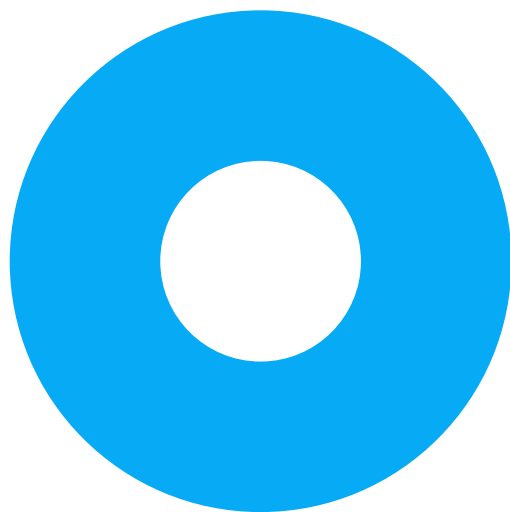
2016-11-07 - 2016-11-20

Total 13 h 24 min

DreamStill selected as projects

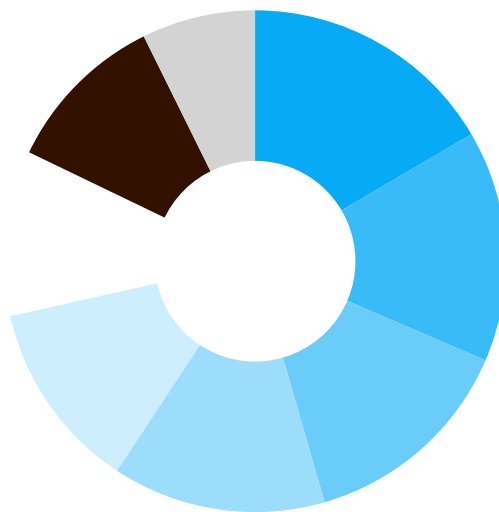










## Projects



 DreamStill 13:24:42

## Time entries



 #12 Login Back-End 2:13:13  
 Reunión TFG 2:00:00  
 #15 Mockups aplicación 1:53:09  
 #14 Dummy Chart con Movim... 1:50:36  
 #13 Almacenamiento de log... 1:38:18  
 #17 Incluir Angular Material 1:25:47  
 #20 Imagen corporativa general 1:24:47  
 Other 0:58:52

#### 4.1.4. Sprint 4

A lo largo del cuarto Sprint, se diseñó la Imagen Corporativa General de la aplicación, se añadió la seguridad a Firebase y se integró el proyecto con Travis. Además, se comenzaron a implementar las primeras historias de usuario, concretamente las tres primeras.

- Burndown:

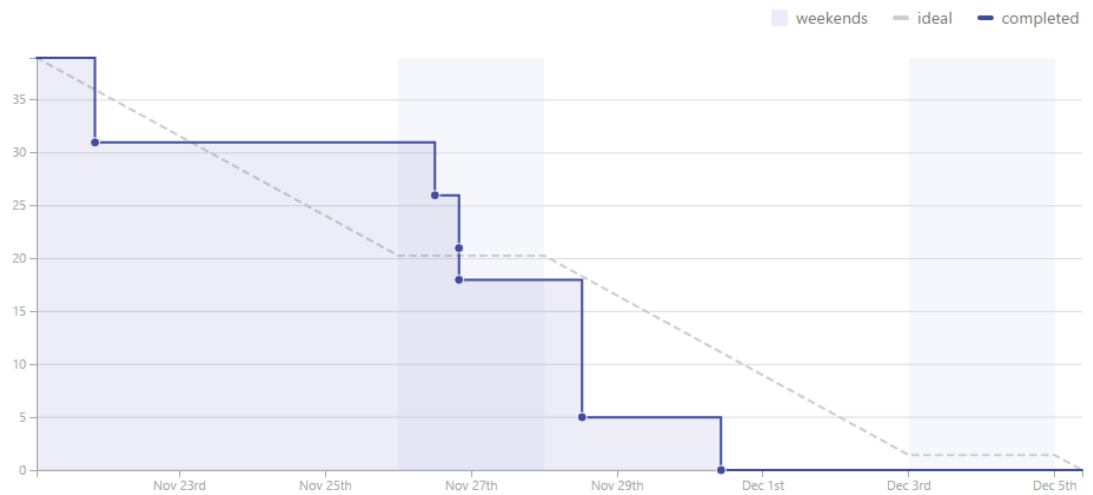


Figura 4.8: Sprint 4

- Informe de Toggl:

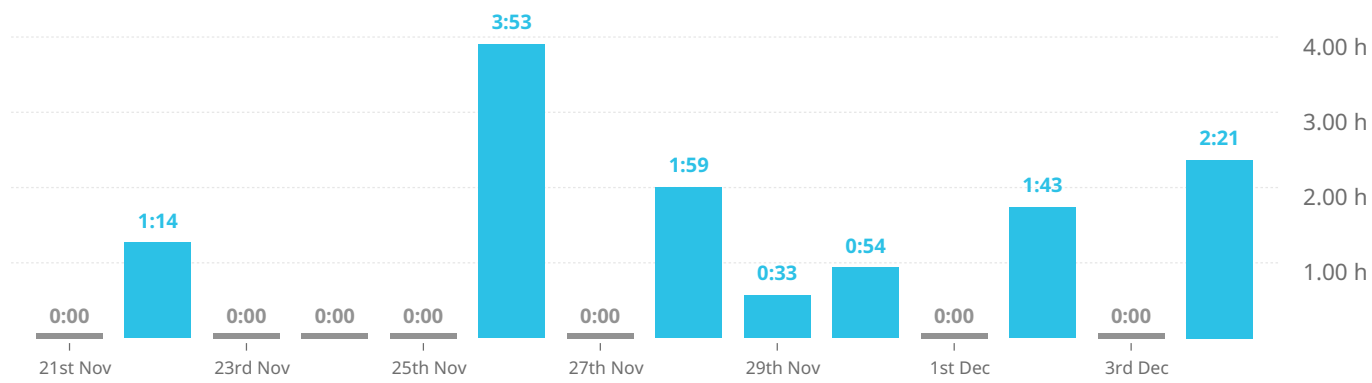
# Summary report



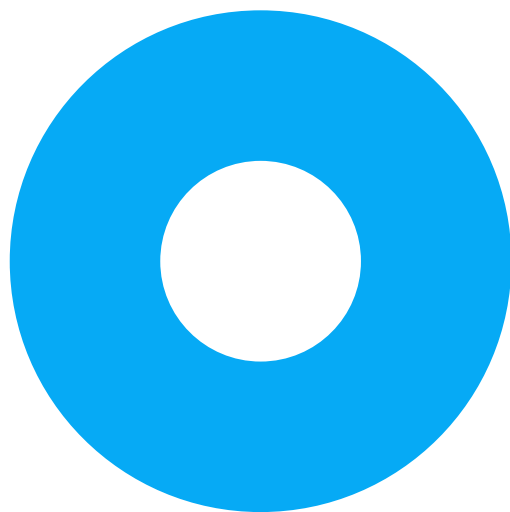
2016-11-21 - 2016-12-04

Total 12 h 40 min

DreamStill selected as projects

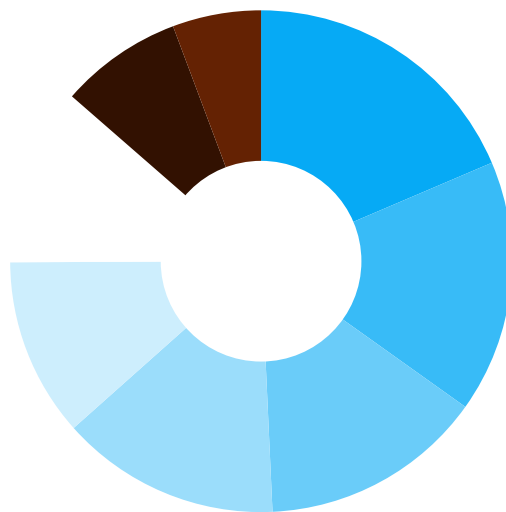



## Projects





 DreamStill 12:40:57


## Time entries




 #25 HU4: Gestión de datos... 2:21:24


 #19 Autenticación al usar... 2:03:51


 #24 Integrar el proyecto ... 1:49:41

 #21 HU1: Registro en la a... 1:47:41

 #23 HU3: Acceder a la pá... 1:27:34

 #22 HU2: Loguearse en la ... 1:27:21

 Reunión TFG 1:00:00

 Vectorizar logo 0:43:25

#### 4.1.5. Sprint 5

A lo largo del quinto Sprint, se desarrollaron las siguientes 5 historias de usuario definidas en el Sprint 3, es decir, desde la cuarta historia de usuario hasta la octava.

- Burndown:

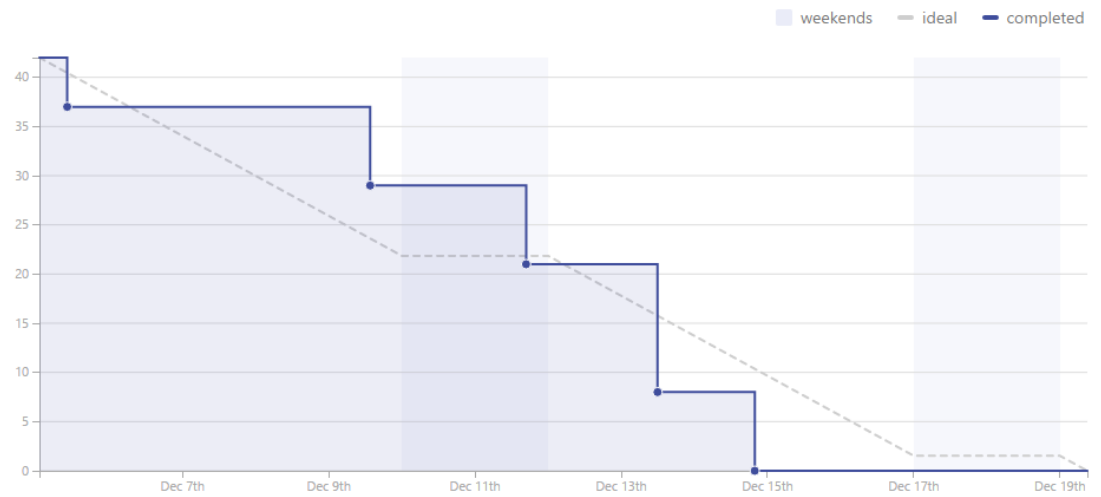


Figura 4.9: Sprint 5

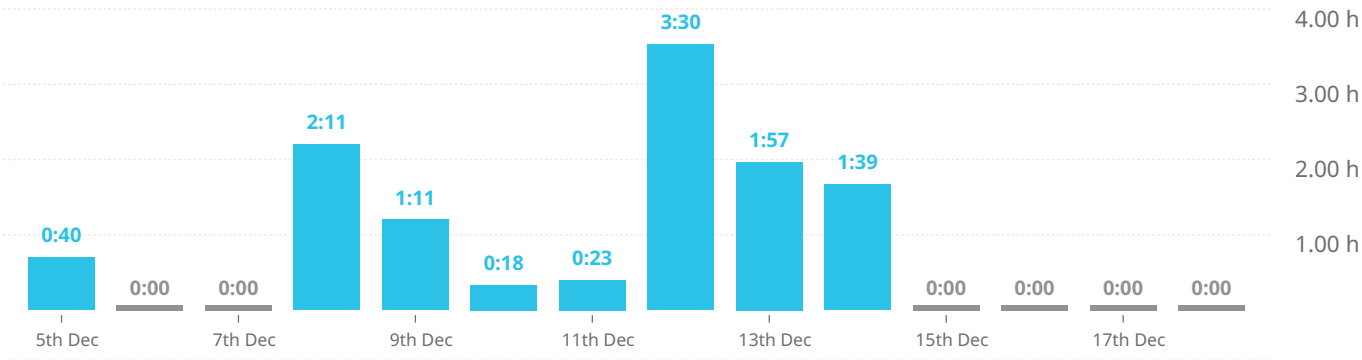
- Informe de Toggl:

# Summary report

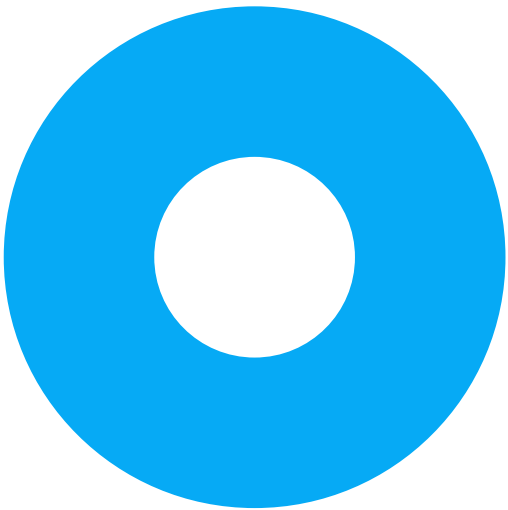
2016-12-05 - 2016-12-18

Total 11 h 51 min

DreamStill selected as projects

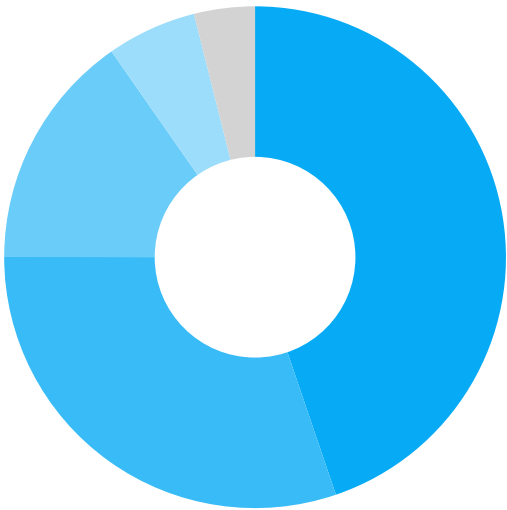







Projects



 DreamStill 11:51:27

Time entries



 #28 HU7: Representación d... 5:18:34  
 #26 HU5: Representación d... 3:35:08  
 #29 HU8: Terminar el proc... 1:48:46  
 #27 HU6: Integración de i... 0:41:06  
 Other 0:27:53

#### 4.1.6. Sprint 6

A lo largo del sexto Sprint, se comenzó la primera fase de la integración con las APIs de GoogleFit y Fitbit. Éstas APIs, son tecnologías que ofrecen muchas de las empresas del sector tecnológico actualmente y que permiten a otras aplicaciones obtener datos de sus bases de datos a través de llamadas realizadas bajo protocolos de Internet.

Debido a nuevas ideas que surgieron durante la ejecución del presente Sprint, se llevó a cabo una actualización de las historias de usuario anteriormente planteadas.

- Burndown:

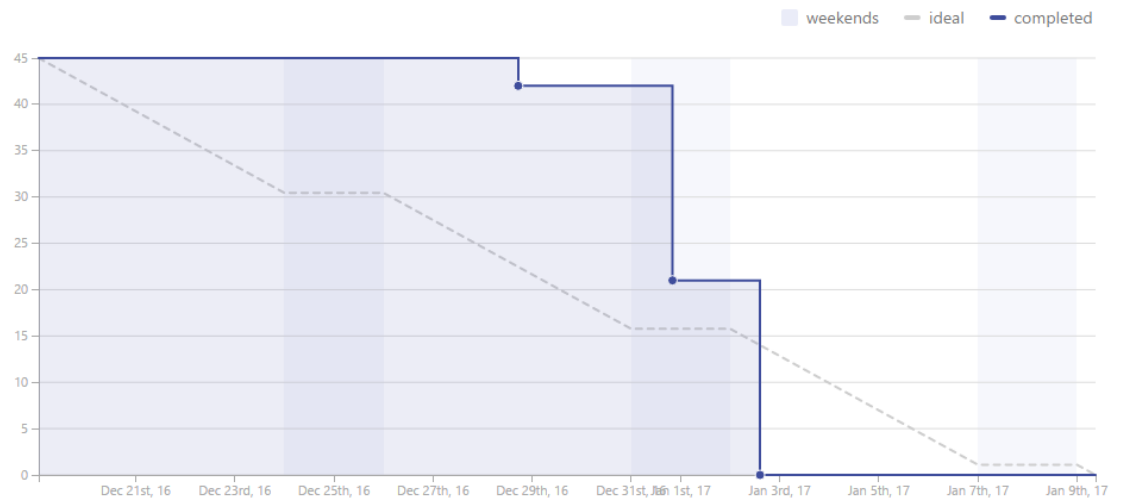


Figura 4.10: Sprint 6

- Informe de Toggl:

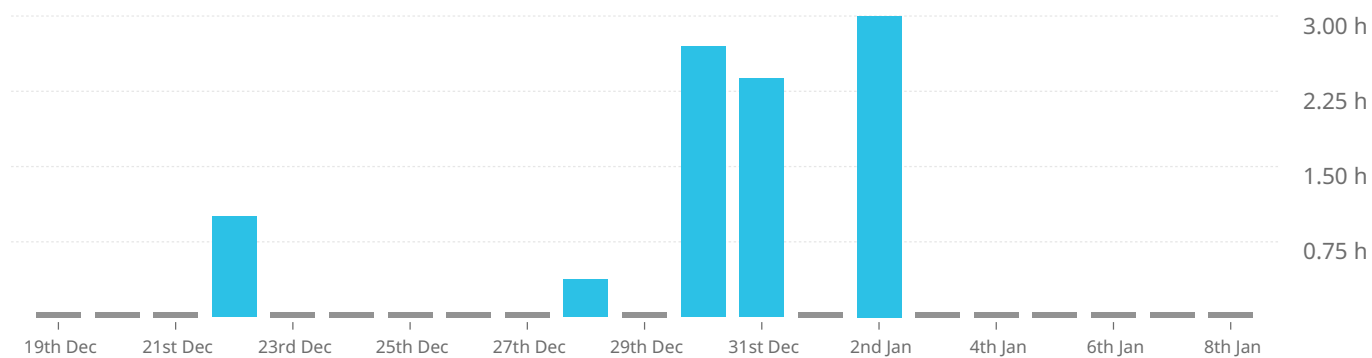
# Summary report



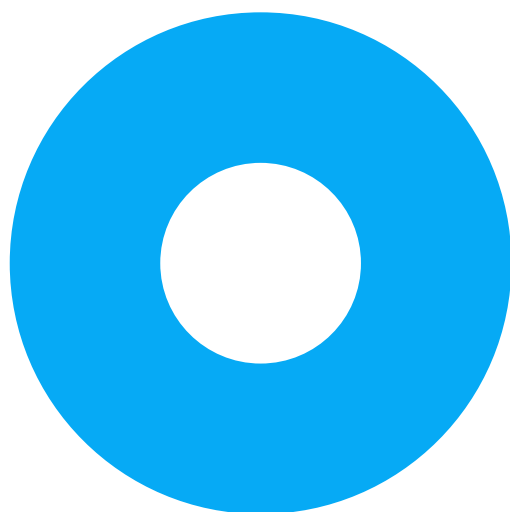
2016-12-19 - 2017-01-08

Total 09 h 25 min

DreamStill selected as projects

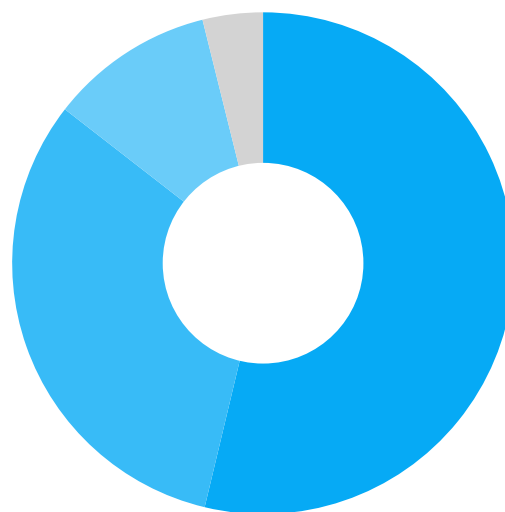






## Projects



 DreamStill 9:25:34

## Time entries



 #33 Integración con la AP... 5:03:56  
 #34 Integración con la AP... 2:59:41  
 Reunión TFG 1:00:00  
 Other 0:21:57

#### 4.1.7. Sprint 7

A lo largo del séptimo Sprint, se continuó la fase de integración con las APIs de GoogleFit y Fitbit y se comenzó con los primeros tests funcionales de la aplicación.

- Burndown:

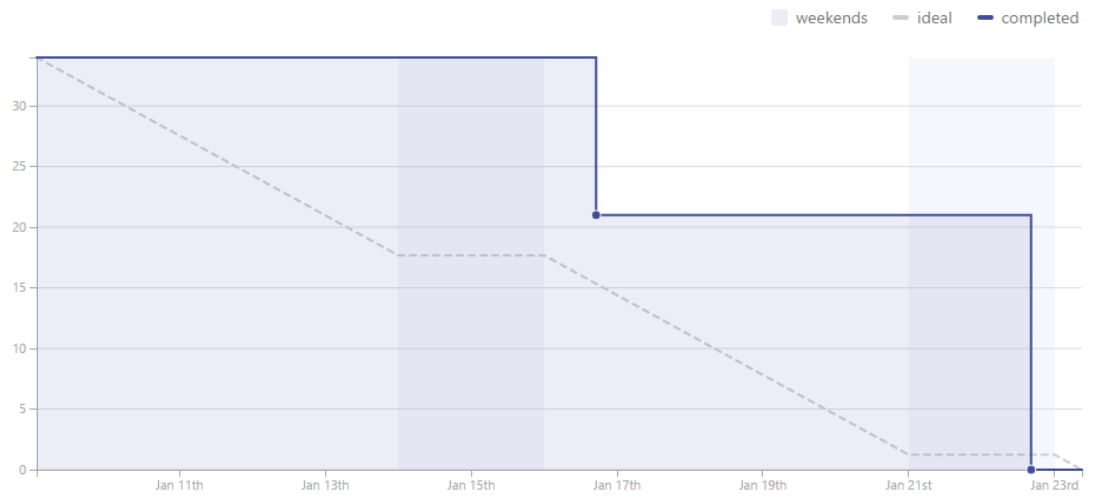


Figura 4.11: Sprint 7

- Informe de Toggl:



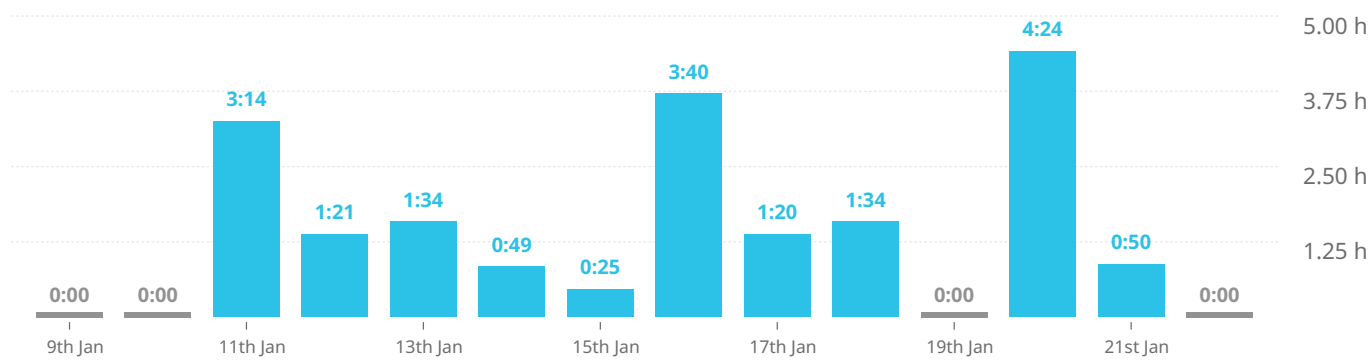
# Summary report



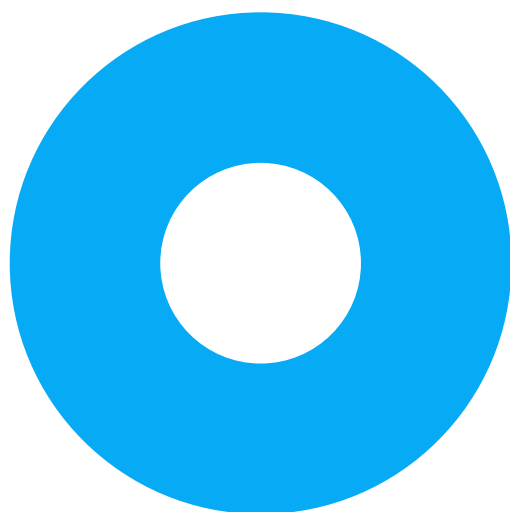
2017-01-09 - 2017-01-22

Total 19 h 15 min

DreamStill selected as projects

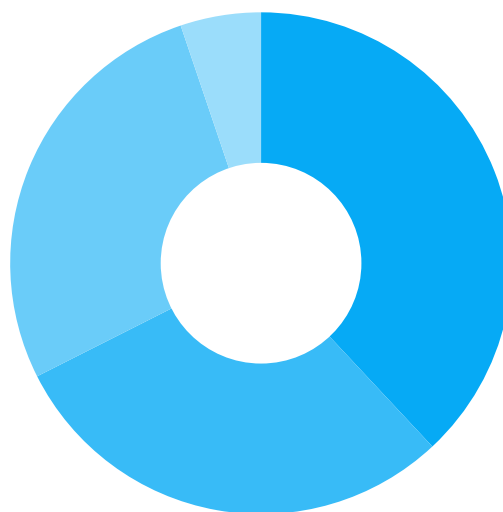






## Projects



 DreamStill 19:15:54

## Time entries



 #35 Finalizar integración... 7:19:41  
 #36 Finalizar y corregir ... 5:41:12  
 #37 Testear aplicación 5:15:01  
 Reunión TFG 1:00:00

#### 4.1.8. Sprint 8

A lo largo del octavo Sprint, se desarrollaron el resto de test funcionales de la aplicación, se creó una página para que los usuarios conocieran los pasos a seguir para enlazar su cuenta de DreamStill con la cuenta de Morpheuz y se estudió la posible integración con la API de Sleep As Android. Respecto a ésta integración, se llegó a la conclusión de que no era una buena alternativa a seguir.

Para terminar, en éste Sprint se añadió la opción de recuperación de contraseñas de la aplicación, para aquellos usuarios que olvidaran su contraseña y no pudieran acceder.

- Burndown:

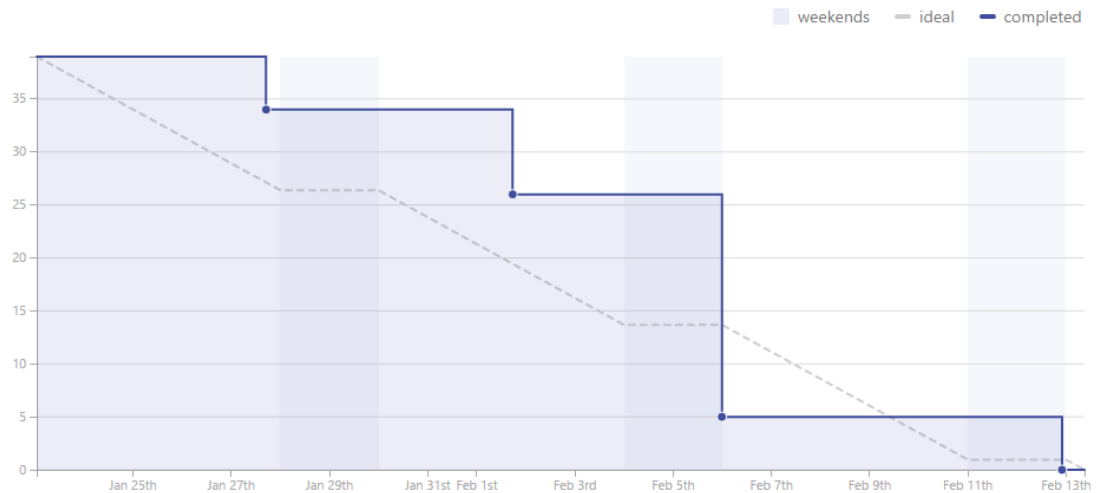


Figura 4.12: Sprint 8

- Informe de Toggl:

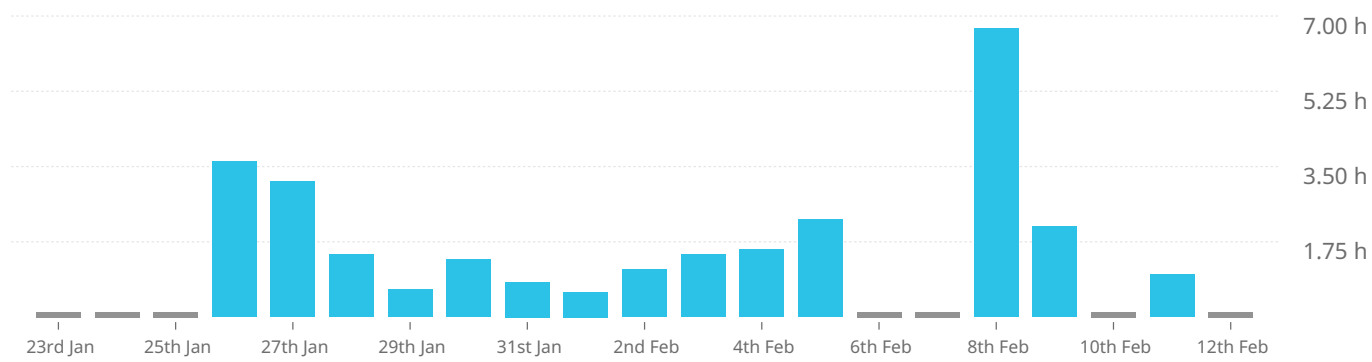
# Summary report



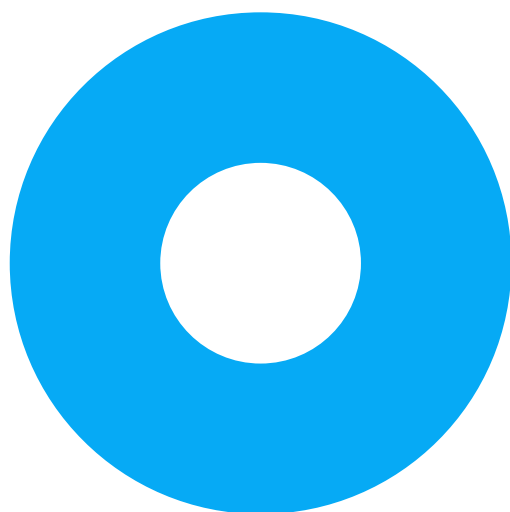
2017-01-23 - 2017-02-12

Total 27 h 26 min

DreamStill selected as projects

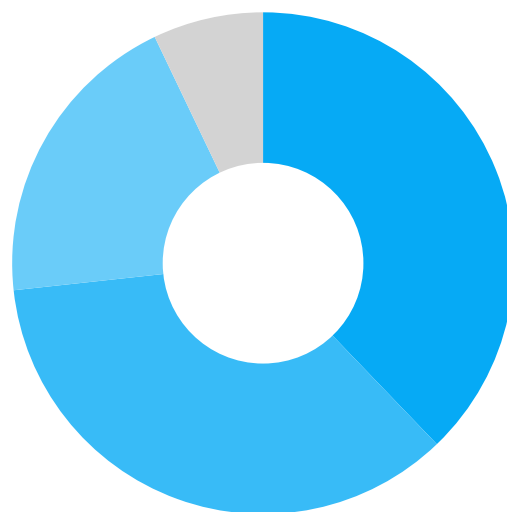






## Projects



 DreamStill 27:26:29

## Time entries



 #41 Continuar con los Tests 10:22:44  
 #43 Añadir opción para re... 9:43:48  
 #40 Crear página para que... 5:22:56  
 Other 1:57:01

#### 4.1.9. Sprint 9

A lo largo del noveno Sprint, se comenzó con la documentación de la presente memoria. Concretamente, se introdujeron los capítulos 1, 2 y 4.

- Burndown:

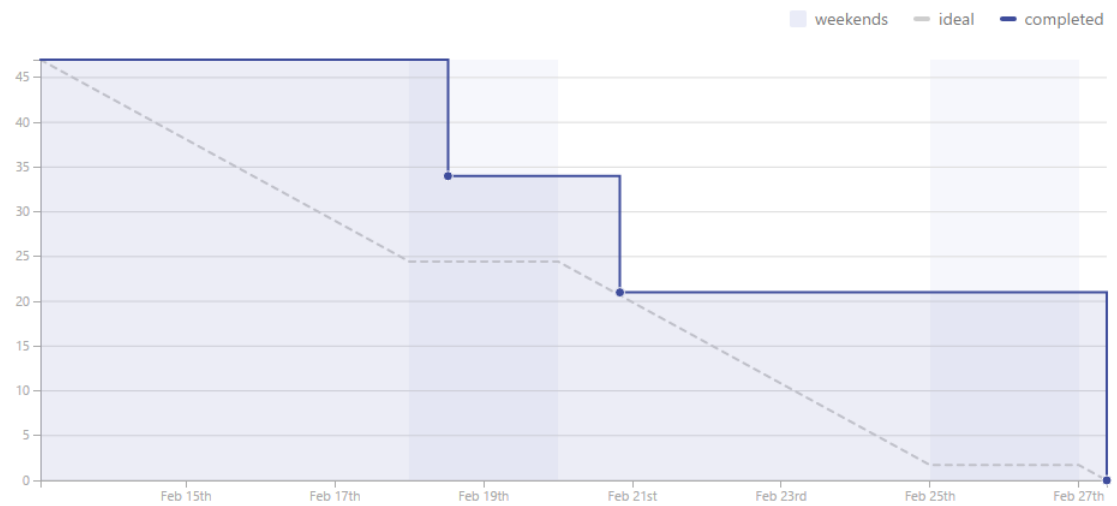


Figura 4.13: Sprint 9

- Informe de Toggl:

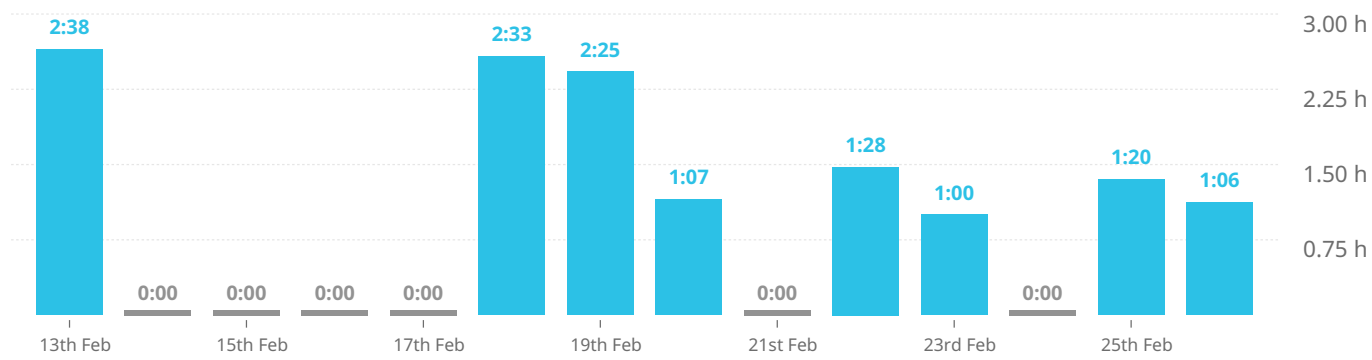
# Summary report



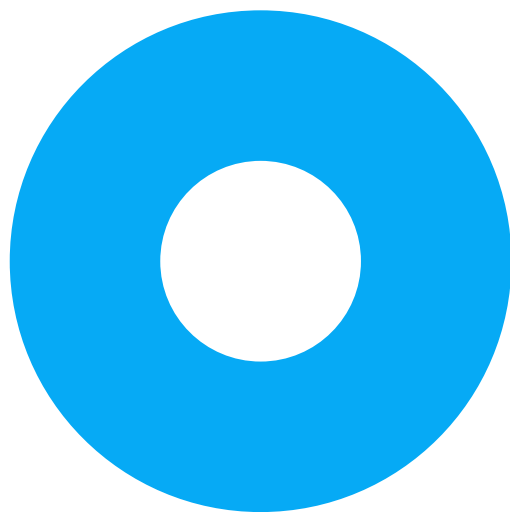
2017-02-13 - 2017-02-26

Total 13 h 39 min

DreamStill selected as projects

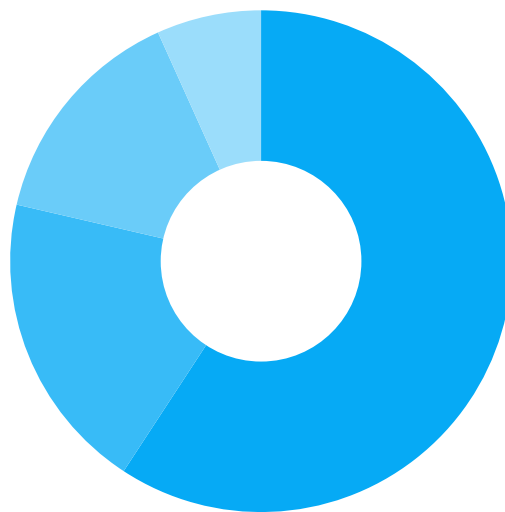






## Projects



 DreamStill 13:39:40

## Time entries



 #46 Documentación Inicial... 8:05:26  
 #45 Documentación Inicial... 2:39:00  
 Reunión TFG 2:00:00  
 #47 Documentación Inicial... 0:55:14

#### 4.1.10. Sprint 10

A lo largo del décimo Sprint, se revisaron los capítulos comenzados en el Sprint anterior y se añadió un sistema de alertas, para que el usuario pudiera activar unas alertas que le informaran de de carencias en su sueño durante un período determinado.

- Burndown:

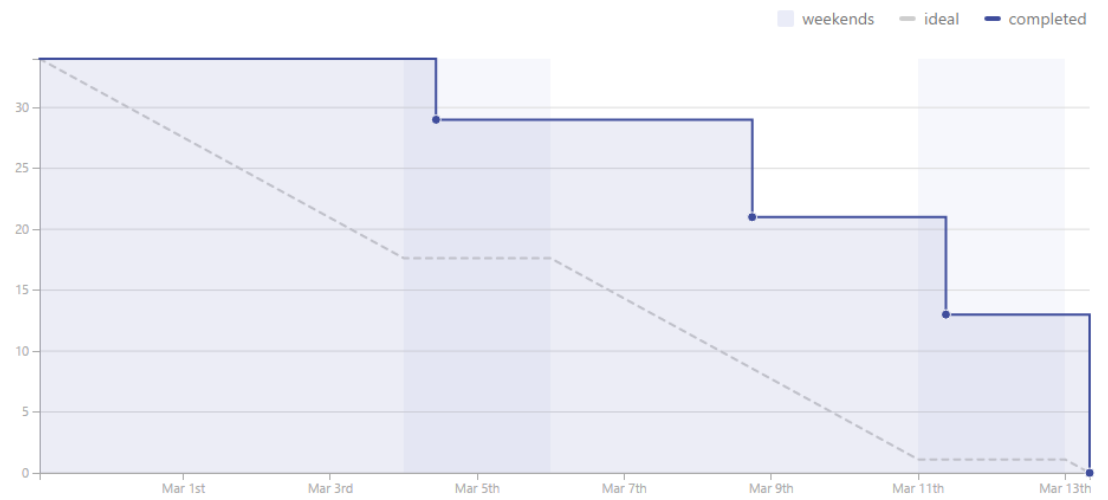


Figura 4.14: Sprint 10

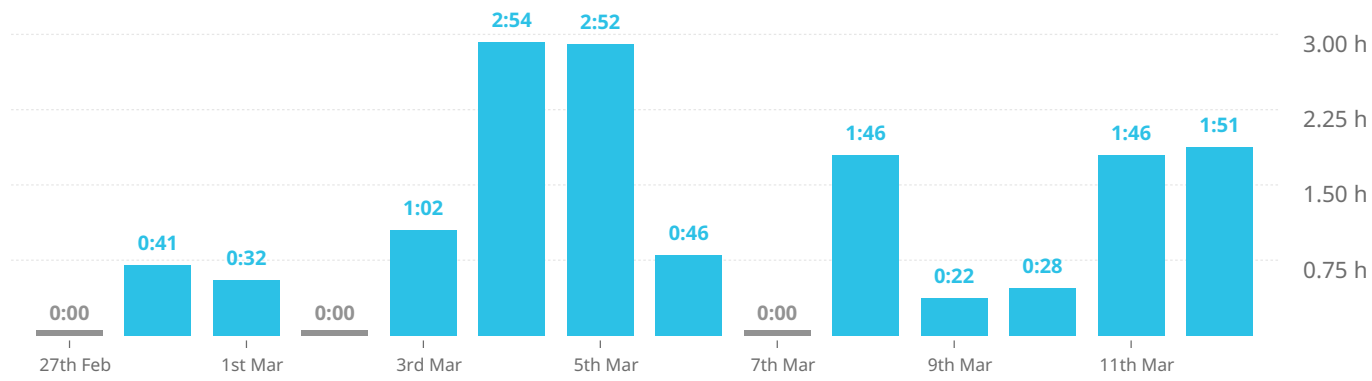
- Informe de Toggl:

# Summary report

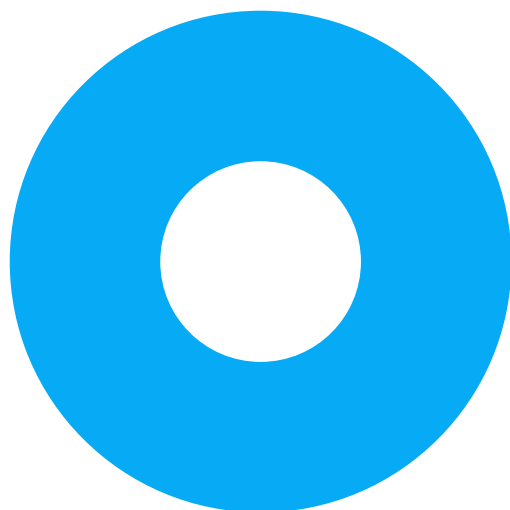


2017-02-27 - 2017-03-12

Total 15 h 05 min

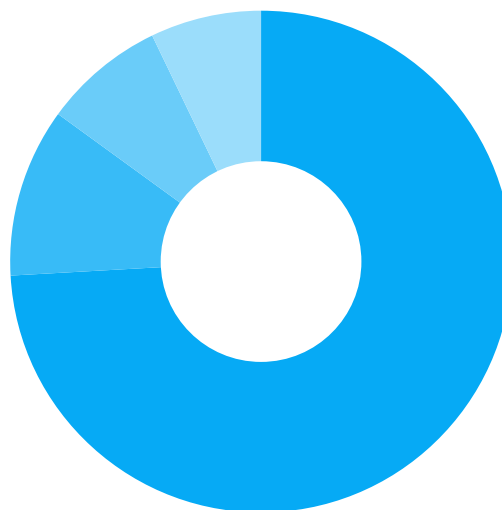






## Projects



 DreamStill 15:05:52

## Time entries



 #51 Añadir sistema de alertas 11:11:20  
 #50 Revisar y corregir ap... 1:38:40  
 #48 Revisar y corregir ap... 1:11:08  
 #49 Revisar y corregir ap... 1:04:44

#### 4.1.11. Sprint 11

A lo largo del undécimo Sprint, se finalizó la documentación de los capítulos 1, 2 y 4 de la presente memoria y se introdujo la documentación de los apartados 1 y 7 de la presente memoria.

- Burndown:

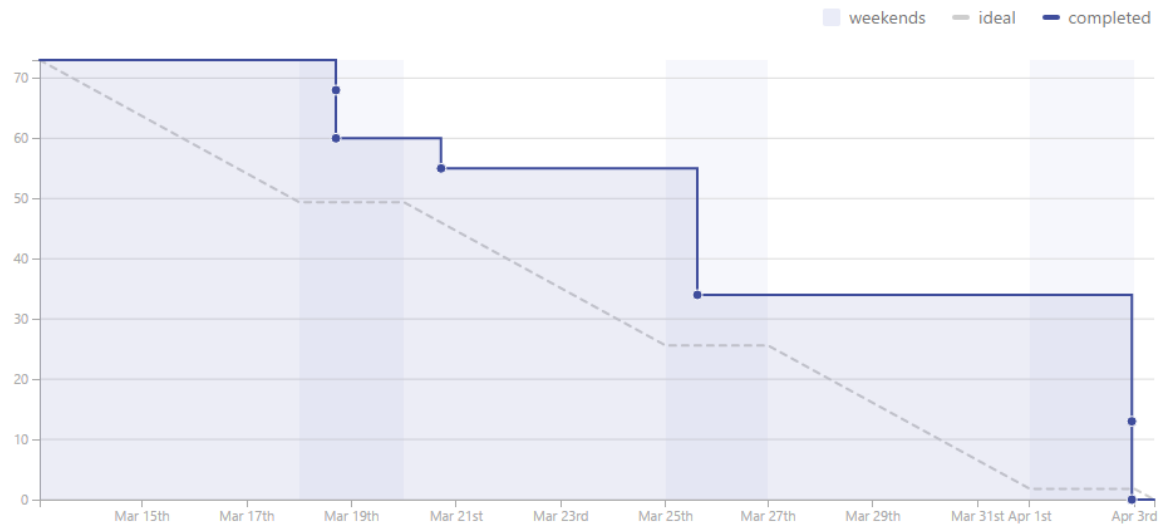


Figura 4.15: Sprint 11

- Informe de Toggl:



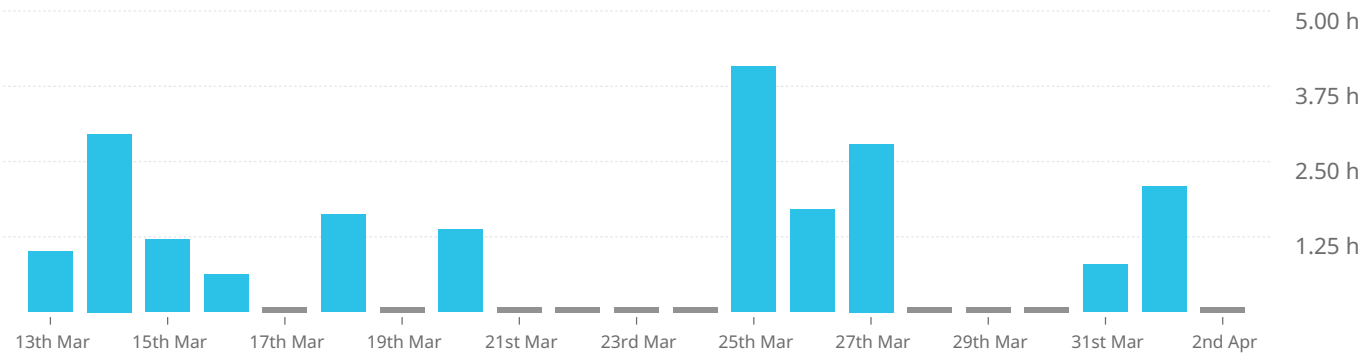
# Summary report



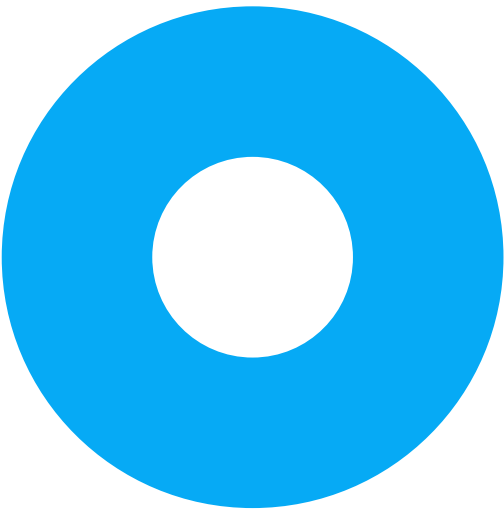
2017-03-13 - 2017-04-02

Total 20 h 01 min

DreamStill selected as projects

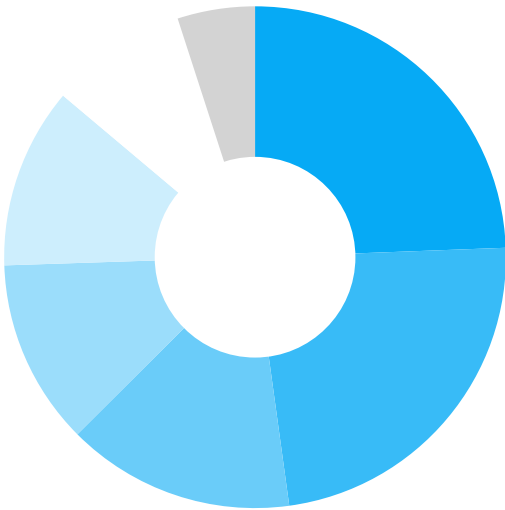









Projects



 DreamStill 20:01:58

Time entries



-  #58 Almacenar y mostrar l... 4:53:17
-  #56 Documentación Apartado 3 4:41:31
-  #53 Finalizar el apartado... 2:56:44
-  #57 Documentación Apartado 7 2:23:41
-  #55 Finalizar el apartado... 2:20:04
-  #54 Finalizar el apartado... 1:46:41
-  Other 1:00:00

#### 4.1.12. Sprint 12

A lo largo del duodécimo Sprint, se finalizó la customización del sistema de alertas. Está customización consiste en ofrecer la posibilidad al usuario de elegir durante cuanto tiempo ha de incumplir un mínimo de horas durmiendo para que se le avise.

- Burndown:

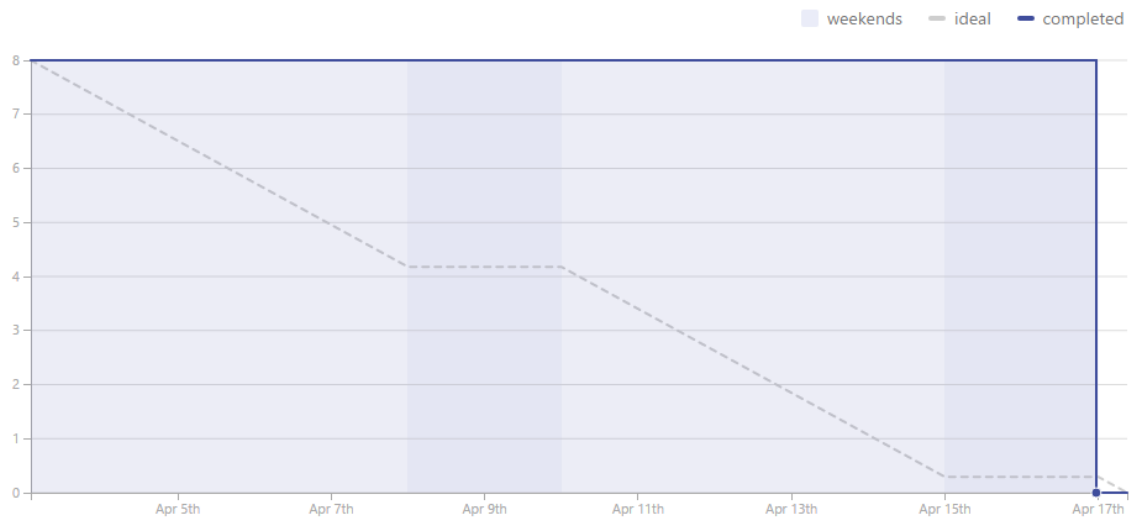


Figura 4.16: Sprint 12

- Informe de Toggl:

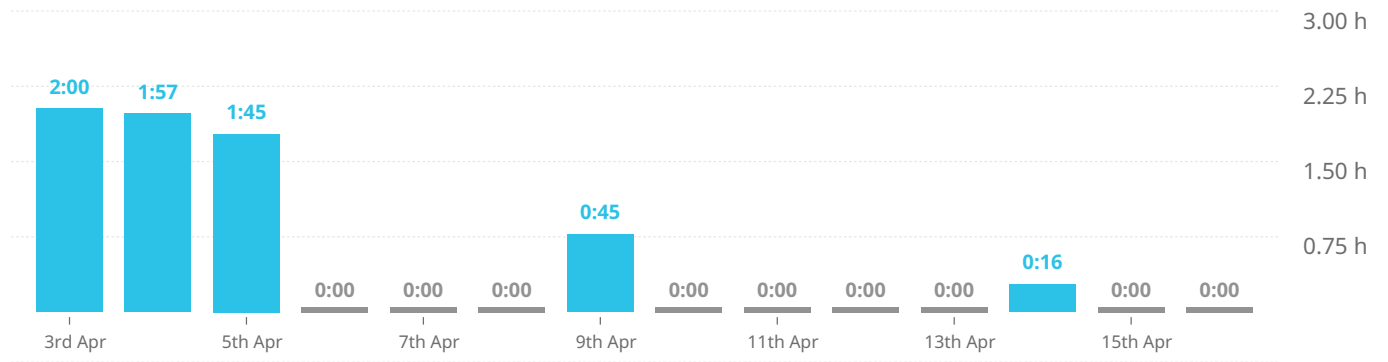
# Summary report



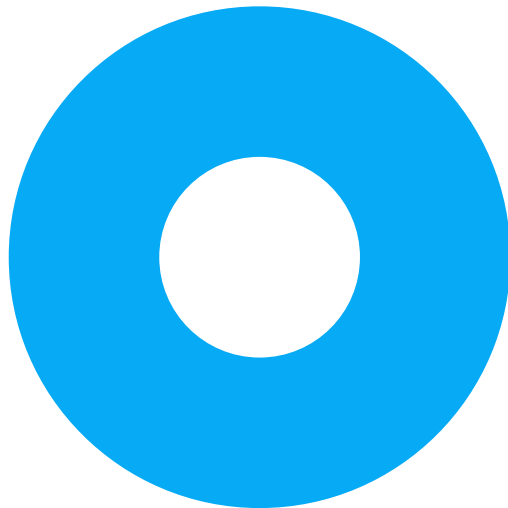
2017-04-03 - 2017-04-16

Total 06 h 45 min

DreamStill selected as projects

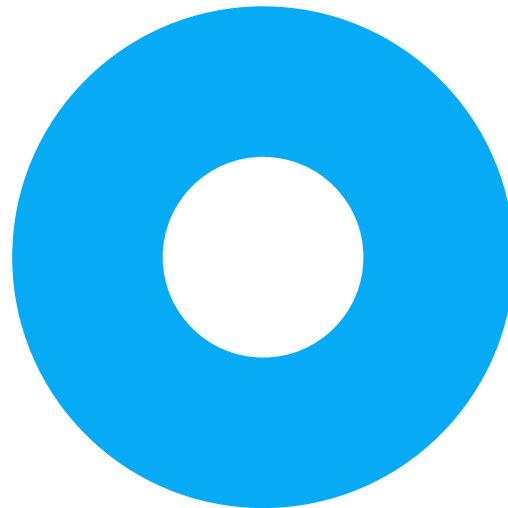


## Projects



● DreamStill 6:45:03

## Time entries



● #60 Añadir parámetros par... 6:45:03

### 4.1.13. Sprint 13

A lo largo del decimotercero Sprint, se finalizó con los apartados 6 y 8 de la documentación. Además, se corrigieron errores que se habían cometido en la documentación en las tareas anteriores.

- Burndown:

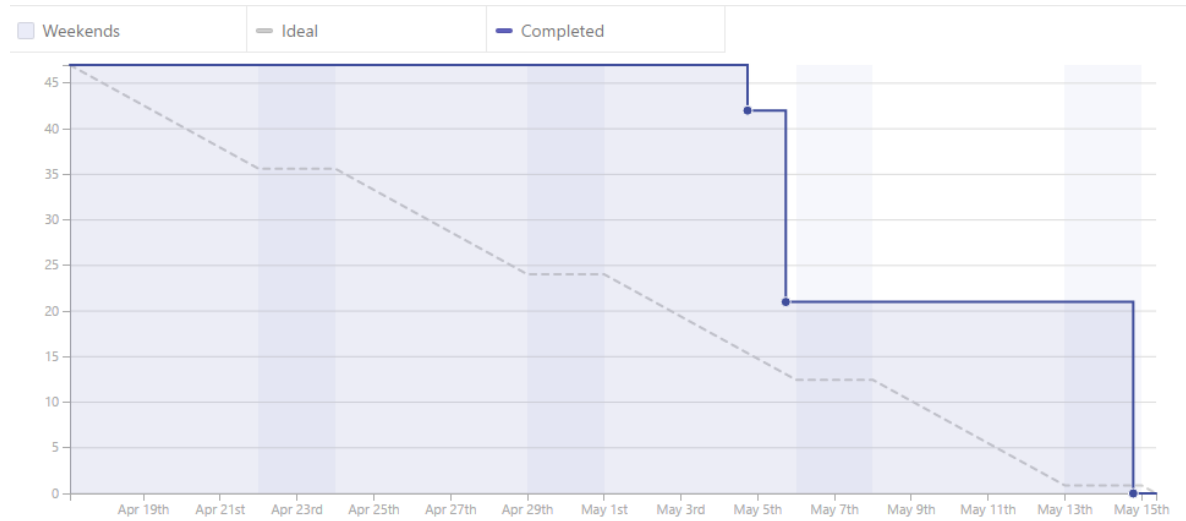


Figura 4.17: Sprint 13

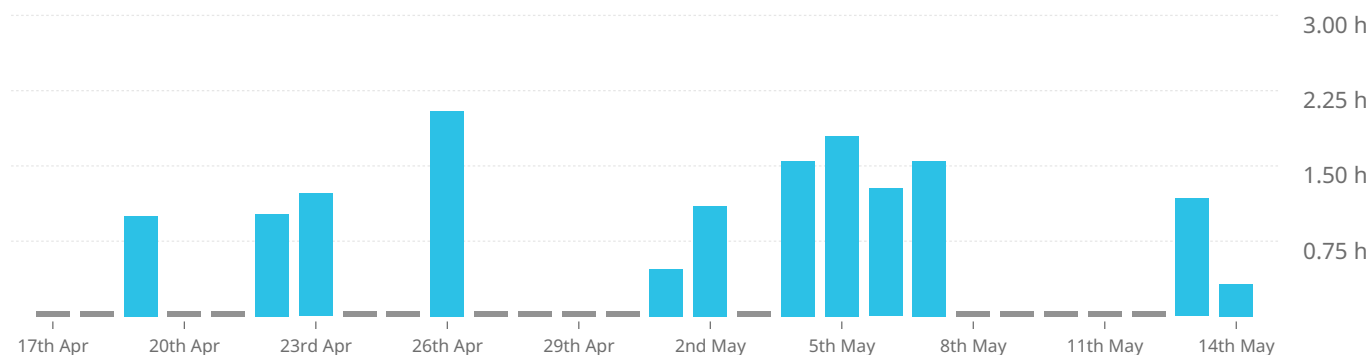
- Informe de Toggl:

# Summary report

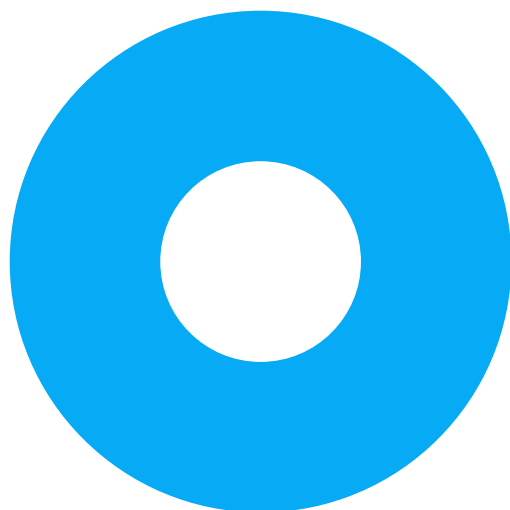


2017-04-17 - 2017-05-14

Total 14 h 24 min

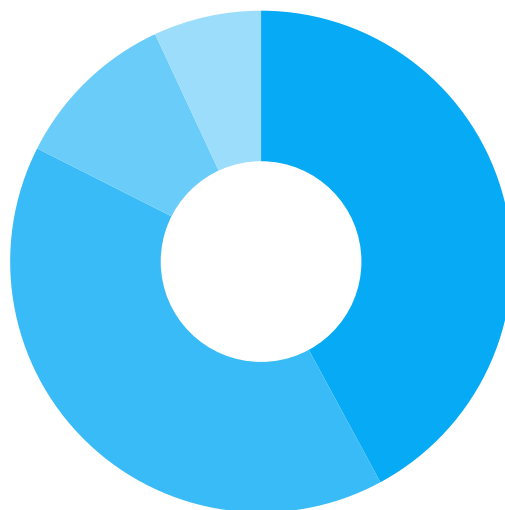






## Projects



 DreamStill 14:24:48

## Time entries



 #64 Documentación Apartado 8 6:04:10  
 #62 Corregir errores de l... 5:48:49  
 #63 Documentación Apartado 6 1:31:49  
 Reunión TFG 1:00:00

## 4.2. Presupuesto

invoice

# Initech Inc.

---

123 Broadway  
City, State 12345

(000) 111-1111  
john@smith.com

**Invoice To:**  
James Smith  
Generic Corporation

**Date:**  
28 de mayo de 2017

Consulting Services

October 3, 2012812 October 4, 20126.512 October 5, 20125.2512  
October 10, 20129.7520 October 11, 2012512.51

Accounting Services

October 10, 2012280 October 11, 2012180

Hosting Expenses

Web Hosting: October, 201260