Diseño e implementación de una API para la conexión de Talentier con diversas ATS

Trabajo Final de Grado
Memoria

Autor: Ignasi Galofré Pujol
Director: Maarten Steurs
Ponente: Jose María Barceló Ordinas
Especialidad: Tecnologías de la Información
Fecha: 20 de Junio de 2016

Agradecimientos

Doy las gracias a mis padres Joan Galofré y Anna María Pujol por haberme dado la oportunidad de estudiar la carrera universitaria que deseaba y en la universidad que quería, ya que en Mallorca donde vivía también se daban los mismos estudios, pero aun así me apoyaron en la decisión de estudiar fuera.

A mi hermana Mariona que siempre que lo necesito me echa una mano, por ejemplo, ella me ha ayudado con la realización del proyecto con la revisión lingüística del trabajo.

También me gustaría dar las gracias a Talentier, por haberme dado mi primera experiencia laboral en el sector del desarrollo tecnológico. En especial a Maarten Steurs quien me ha enseñado muchos aspectos que voy a tener siempre en cuenta en el futuro tanto en el desarrollo de software como en la gestión de equipos.

Y a todos mis amigos más cercanos y los que he hecho durante los estudios con los que siempre paso tan buenos momentos.

Índice

1.1 Introducción

Este proyecto corresponde a un Trabajo Final de Grado en Ingeniería Informática de la Universidad Politécnica de Cataluña realizado mediante la modalidad B (proyectos realizados en empresas). La empresa en la que se realizó el proyecto es Talentier Lean Recruiting, S.L.

El autor del trabajo es de la especialidad de Tecnologías de la Información. Aunque este proyecto no tenga el foco en la especialidad realizada, sí que se aplican conocimientos adquiridos en algunas asignaturas estudiadas en dicha especialidad.

Talentier es una *startup*[1]. Fue fundada en Julio de 2015, por 3 socios fundadores y un inversor (The Venture Gang[2]). A finales de año empezaron a contratar a nuevos miembros (entre los cuales se incluye el autor de este proyecto) con el objetivo de crecer más rápidamente. El equipo actual está formado por 9 personas; el equipo desarrollador ha pasado de tener un solo programador a tener 3, además de un nuevo diseñador. También se ha formado un equipo de *marketing*. Cabe destacar que todos los nuevos miembros forman parte de convenios de distintas universidades, lo que forma un equipo joven y motivado.

[1] Compañía emergente, la cual busca arrancar, emprender o montar un nuevo negocio, que hace referencia a ideas de negocios que están empezando o están en construcción, apoyadas en la tecnología.

[2] The Venture Gang es una factory de start-up fundada en 2014. Está impulsada por un grupo inversor de profesionales de primer nivel que engloba diversos sectores (gran consumo, fármacos, alimentación, banca, consultoría, recursos humanos...), especializaciones (gestión, *marketing*, ventas, finanzas, operaciones) y tipos de empresa (desde familiar a multinacional).

1.2 Contexto

Talentier Lean Recrtuiting, S.L. pretende cambiar el paradigma de reclutamiento en España, donde el procedimiento de contratación suele realizarse mediante publicaciones de ofertas de empleo a través de distintos medios y a partir de los CV recibidos para encontrar a posibles candidatos. Su intención es que cada vez se realicen más externalizaciones a la hora de buscar posibles candidatos mediante *recruiters*[3], como ocurre en el norte de Europa, donde hay un alto porcentaje de estas prácticas.

En España suele haber desconfianza o poca comunicación entre empresas y *recruiters*. La idea de Talentier es, a través de una herramienta, hacer el proceso más eficaz y menos costoso para todos, permitiendo centralizar y gestionar los puestos de trabajo y facilitando también una buena comunicación entre empresas y *recruiters*.

Talentier es un *web service* basado en un *marketplace*, el cual conecta empresas que necesitan contratar a personas a través de consultores de selección especializados (o *recruiters*). Talentier es también una herramienta de colaboración eficiente que ayuda a las dos partes a cubrir las posiciones en menos tiempo y con menos esfuerzo.

La plataforma funciona de la siguiente manera:

1.- La empresa sube la posición o bien la importa a través de alguna ATS[4], propone una tarifa y escoge qué *recruiters* de la plataforma quiere invitar a trabajar en la posición.

2.- Los *recruiters* aceptan y a partir de entonces empiezan a proponer candidatos mediante la plataforma.

3.- El cliente paga al *recruiter* que le ha proporcionado el candidato que acaba contratando.

Antes de describir el proyecto es importante explicar que Talentier se ha asociado como *partner* con la empresa Talent Clue, una ATS. Este es un *software* de reclutamiento que permite a las empresas automatizar los procesos de selección, publicando las vacantes directamente en las redes sociales, los portales de empleo y en la propia web, y recibir todos los candidatos en un solo lugar.

[3] Consultores de selección especializados.
[4] Sistema software de seguimiento y gestión de candidatos para ofertas de empleo.

Talentier está integrado dentro de Talent Clue de la siguiente manera: Una vez creadas y publicadas las ofertas, ofrece la opción de externalizar la búsqueda de candidatos con Talentier; ahora bien, en vez de hacerse todo desde el propio Talent Clue, redirecciona a la plataforma de Talentier en la que se seleccionan a los *recruiters* y se proponen las tarifas, para después volver a Talent Clue (donde se gestionarán los CV enviados por los *recruiters*).

Por un lado, el proyecto realizado ha consistido en el diseño e implementación de una API[5] para la integración de Talentier con cualquier ATS. Por otro lado, en el proyecto se plantea sustituir la conexión actual con Talent Clue para que use la nueva API y rediseñar la integración actual (que tenía varios problemas de usabilidad) para volverla más visual e intuitiva.

Talentier tiene el propósito de crecer de forma más acelerada. Con esta intención realizó una importante inversión tanto en desarrollo como en *marketing* en busca de nuevos clientes/usuarios.

A través del departamento de *Marketing*, Talientier abrió un blog a principios de año para lograr un buen posicionamiento SEO[6] que le permitiera recibir más visitas y lograr *leads*[7]. Estos son seguidos mediante Hubspot[8] y Google Analytics[9].

En lo que respecta a la parte tecnológica (a la cual iba destinado este proyecto), mediante la creación de la API, Talentier pretende integrarse a nuevas ATS, ya que gran parte de los usuarios de la plataforma proceden de la integración con Talent Clue.

[5] Application Programming Interface - Interfaz de programación de aplicaciones.
[6] Search engine optimization - mejorar la posición en los buscadores de internet.
[7] Persona que muestra interés.
[8] Software de Marketing.
[9] Herramienta de análisis web.

1.3 Problemática a resolver

Anteriormente había dos problemas con la plataforma instalada en Talent Clue.

En primer lugar se encuentra el uso de un *pop-up*[10] que, a pesar de ser funcional (porque solucionaba el problema inicial de incorporar Talentier en la plataforma), hacía que la plataforma de Talent Clue no fuese ni consistente ni usable, ya que parecía una nueva herramienta totalmente distinta dentro de un pequeño *pop-up* comprimido.

Por otro lado, se detectó que la incorporación de Talentier a otras plataformas no era escalable, ya que para cada ATS había que aplicar adaptaciones. Así pues, la intención fue conseguir que la herramienta fuera lo más fácil y rápida de adaptar con cualquier otra ATS. Para eso se pensó en unificar todas las utilidades necesarias en una API.

[10] *Pop-up* - Ventana emergente

1.4 Actores implicados

Los actores implicados en la realización del proyecto han sido:

- **El director del proyecto:**

Maarten Steurs, CTO[11] de Talentier. Anteriormente fue el único desarrollador de la empresa y es el creador de la conexión actual con Talent Clue. Su misión fue explicar al autor del proyecto la arquitectura actual de la conexión existente y dar apoyo para establecer la arquitectura de la API a realizar.

- **El desarrollador:**

Ignasi Galofré, el autor del proyecto. Estudiante de Grado en Ingeniería Informática. Encargado de diseñar y desarrollar el API, siguiendo las pautas y tiempos planteados al principio del proyecto.

- **Talentier:**

La plataforma tiene como objetivo un crecimiento que permita integrar nuevas ATS y llegar a nuevos clientes, quienes gracias a la creación de la API tienen la posibilidad de abrir muchas ofertas de trabajo nuevas procedentes de las conexiones establecidas con otras plataformas.

- **Talent Clue:**

Ha sido la primera ATS en integrar el API de Talentier, y es con la que se han realizado las pruebas de funcionamiento y rendimiento.

- **Los usuarios:**

Los usuarios potenciales de esta integración son nuevas empresas que trabajan con las distintas ATS que podría albergar Talentier.

[11] CTO - Director de tecnología de una organización.

1.5 Posibles obstáculos

- **Proyecto acotado en el tiempo:**

El proyecto estaba limitado en el tiempo, ya que tenía una fecha máxima de entrega, por lo que se hacía necesaria una buena planificación previa. El proyecto sufrió cambios en la planificación, pero hubo tiempo suficiente para finalizarlo.

- **Desconocimiento previo de la tecnología:**

Talentier trabaja con Drupal, un *framework* con una curva de aprendizaje elevada. Esto supuso que con cada acción no realizada anteriormente se tuvieron que aprender las formas de llevarla a cabo. A parte de Drupal también se trabajó con otras tecnologías nuevas.

- **Mal diseño inicial:**

La posibilidad de plantear un mal el diseño al comienzo habría conllevado a tener que volver a dedicar tiempo al rediseño de la arquitectura, lo cual habría acortado más el tiempo disponible para llevar a cabo el proyecto.

- **Contactos con otras ATS:**

Una vez desarrollado el proyecto se desea incorporar más ATS al *web service*, puesto que este proyecto no solamente se realiza para mejorar la interacción con Talent Clue. En este sentido, un posible obstáculo era el de no poder probar el resultado en otra plataforma. Ahora, una vez terminado el proyecto, aún no se ha probado la plataforma con otra ATS, por lo que no se puede asegurar al 100% que sea funcional en cualquier plataforma.

- **Recursos humanos limitados:**

El hecho de ser el desarrollador una sola persona conllevaba más carga de trabajo que siendo un equipo. Así pues, se corría el riesgo de no tener tiempo para acabar todas las tareas planificadas en un principio (con la consecuente alteración en el resultado final que eso podía conllevar).

- **Dependencia de Talent Clue:**

Al tener que probar la API con Talent Clue, era necesaria la coordinación de las dos partes. De hecho, un ejemplo de esta problemática se manifestó al tener que aplazar la prueba final un par de días.

1.6 Motivaciones personales

A pesar de haber hecho la especialidad de Tecnologías de la Información, lo que en realidad me apasiona es el desarrollo de software y es por eso que todas las optativas que he escogido son de la especialidad de Sistemas de la Información.

Desde antes de empezar los estudios universitarios tenía claro que mi objetivo era formarme para una vez acabada la carrera empezar un proyecto por mi cuenta. Y es por eso que cuando vi la oportunidad de realizar el Trabajo de Final de Grado en una StartUp no dudé en tirar mi Curriculum en Talentier. Ya que actualmente también estoy inscrito en el concurso de emprendeduría Yuzz que lleva a cabo el Espai Emprèn.

Una vez me propusieron varios proyectos escogí este en concreto ya que era el que más complejidad en cuanto a estudio previo, diseño y desarrollo tenía. Esto me hizo pensar que podría poner en práctica buena parte de los conocimientos aprendidos en la universidad.

2.1 Estado del arte

Una vez vista la problemática principal de la escalabilidad de Talentier (que es uno de los principales inconvenientes de cualquier *startup*), se decidió idear una API que se pudiera integrar en distintas plataformas de reclutamiento.

El propósito de una API es permitir la comunicación entre componentes de *software* mediante el soporte de funciones de otros servicios, lo que permite la abstracción del código. Con la adaptación del presente proyecto se pretende conseguir que la plataforma pueda integrarse en diferentes ATS para externalizar la búsqueda de candidatos con los *recruiters* registrados en Talentier.

Para ello primero hay que estudiar la arquitectura de la conexión existente con Talent Clue. Tanto Talentier como Talent Clue trabajan con Drupal[12], factor que hace más sencilla la integración de una con la otra. Sin embargo, el objetivo es incorporar una API a la plataforma actual que no solamente funcione entre plataformas basadas en Drupal, sino que se pueda integrar con cualquier tecnología web existente.

Por el hecho de trabajar con Drupal, existen módulos que se pueden usar y modificar al gusto, de modo que se buscaron formas de realizar API con Drupal. En la actualidad existen 4 módulos: Services, RestWS, Endpoint y RESTful.

Actualmente hay infinidad de API. Un ejemplo son los servicios de Google (Google Maps, Google +, etc.). La mayoría de servicios web actualmente ofrecen su propia API para poder ser incorporados en otros servicios.

[12] Framework web open-source basado en PHP y compatible con MySQL.

2.2 Solución propuesta

La solución al problema de escalabilidad era cambiar la integración con Talent Clue construyendo una estructura que pudiese ser usada en cualquier otra plataforma ATS. La antigua integración hacía uso de un módulo que solo se puede integrar en otras plataformas Drupal (Talent Clue también usa Drupal); por eso se buscó una forma de hacer el servicio abstracto y así solucionar el problema de la escalabilidad, haciendo posible el incorporar la plataforma de Talentier a cualquier ATS. La estructura que permitía realizar este proceso era una API.

La solución al *pop-up* incorporado en Talent Clue fue el rediseño completo de este mediante la creación de un *template*[13] de Drupal para que las vistas dentro del *pop-up* fueran distintas a las de la web de Talentier. Asimismo, se aplicaron determinados cambios: se aumentó el tamaño (ya que anteriormente era relativamente pequeño) y se corrigieron errores de usabilidad en botones.

[13] Plantilla.

3.1 Fases del proyecto

El proyecto tuvo una duración aproximada de 5 meses, de finales de enero a finales de junio de 2016. El proceso empezó el 15 de enero, momento en el que el autor del proyecto entró a trabajar en la empresa Talentier. El plazo máximo de entrega era el 20 de junio.

Este proyecto se dividió en seis fases, más una fase final no contemplada al comienzo. Dichas fases fueron las siguientes:

- **Fase I:** Entornos e iniciación a Drupal[14]

Al entrar a trabajar se tenía que realizar la instalación del entorno, tanto de Talentier como de Talent Clue. Además, fue necesario un proceso de familiarización con Drupal. Por otra parte, las prácticas en la empresa incluían un trabajo paralelo en Talentier a parte de la realización del proyecto.

- **Fase II:** GEP

En esta fase hubo un paréntesis en el desarrollo del proyecto, debido a que entre el trabajo normal a realizar en Talentier se tenían que llevar a cabo 6 entregas de la asignatura GEP, planificadas en un periodo corto de tiempo (5 semanas). Por este motivo, durante las horas de trabajo en Talentier se siguió llevando a cabo la parte de aprendizaje de Drupal de la primera fase.

- **Fase III:** Estudio y diseño

A la semana siguiente de la finalización de GEP se retomó el proyecto, empezando con el estudio de la plataforma actual y el diseño del API. Esta fase

[14] Framework PHP para el desarrollo web.

consiste en definir bien todas las posibles comunicaciones entre plataformas y toda la información que se desea entregar en cada petición.

- **Fase IV:** Implementación

Una vez diseñada la API, se procedió al desarrollo de esta, siguiendo las pautas planificadas durante el estudio. Esta fase fue a la que más tiempo se dedicó, puesto que incluye el tiempo previo de estudio de nuevas tecnologías a usar más todo el tiempo necesario para conseguir los objetivos planteados.

- **Fase V:** Sustitución

Una vez implementada, se sustituyeron las comunicaciones anteriores de Talent Clue por las nuevas peticiones a la API. Además del rediseño del *pop-up* que integra a Talentier en la plataforma de Talent Clue.

- **Fase VI:** *Testing*[15] y *debugging*[16]

La fase final consistió en comprobar que todas las comunicaciones y procesos que se realizan entre plataformas siguieran funcionando correctamente y también en corregir cualquier fallo encontrado.

- **Fase Final:** Entrega

Esta fase no fue contemplada al inicio, pero por la necesidad de realizar este paso y por la cantidad de tiempo necesario, se incluyó como una fase más del proyecto. Esta consta de dos tareas distintas: la redacción de la memoria y la preparación de la exposición final.

[15] Testing - Pruebas del software.
[16] Debugging - Depuración del software.

3.2 Metodologías de trabajo

Cuando el desarrollador de proyecto (el estudiante) entró a trabajar en Talentier, se usaba la metodología ágil SCRUM (consistente en planificar tareas más cortas y concretas, en contraste con el planteamiento de fases más largas y complejas). Estas pequeñas tareas se reparten en *sprints* de una o dos semanas, dependiendo de la cantidad de trabajo que lleve cada una y del objetivo concreto que se persiga en cada caso.

3.2.1 Scrum

Semanalmente se realizaban las distintas reuniones:

- ***Sprint Focus*:**

  Donde se decidía a qué se enfocaría el siguiente *sprint*.

- ***Ticket Definition*:**

  Se definían las tareas y se asignaba a cada una un tiempo estimado de realización.

- ***Sprint Planning*:**

  Se asignaban los *tickets*, es decir, las tareas a realizar en el siguiente *sprint*.

- ***Sprint Review*:**

  Observación del trabajo completado hasta el momento que permitió pequeños cambios en los *tickets* en los casos en los que no se había calculado con precisión la inversión de tiempo que suponían.

- ***Sprint Retrospective*:**

  Al final del *sprint* se hacía una reunión con el equipo durante la cual los miembros compartían impresiones con relación a tareas determinadas.

- ***Demo*:**

  Al final del *sprint* se mostraba el resultado al equipo no técnico.

A pesar de seguir una metodología rigurosa, a mediados de abril el equipo de desarrollo detectó que no se acababan de cumplir los plazos establecidos. Se encontraron tres problemas principales: mala definición de las tareas, tareas con un alcance demasiado grande y dificultades para establecer plazos. De hecho, hasta ese momento, se habían alargado los *sprints* hasta el punto de que se había dejado de seguir la metodología SCRUM.

A raíz de esta problemática, el equipo decidió crear una metodología propia, que es la que se está usando actualmente. Hay que decir que el grupo ha notado una diferencia de productividad una vez instaurada esta nueva metodología.

3.2.2 SCRUM + Kanban

A principios de mayo se puso en marcha la nueva metodología. Se basa en los siguientes aspectos:

- No hay plazo para actualizar los datos para la sección de producción; cada día a primera hora se suben las novedades del día anterior.

- Los *tickets* no pueden llegar 8 *story points*[17] (8 se considera más de un día de desarrollo). Por esta razón, las tareas grandes se tienen que dividir en *subtickets*.

- Todos los desarrolladores se encargan de desarrollar, revisar el código de otros y testear los *tickets* de otros.

- Se tiene que comunicar al equipo rápidamente cualquier error que se encuentre en un *ticket*, ya sea un error de definición o un problema en el desarrollo. De este modo se pretende resolver fallos cuanto antes mejor y evitar los que puedan producirse más adelante.

Además de estos puntos básicos también se realizan las siguientes reuniones:

- *Daily Stand-up*: Cada día a las 9.30 se celebra una reunión de un máximo de 15 minutos en la que se revisan las métricas para ver si hay alguna tarea que bloquee el proceso y tomar medidas rápidamente.

[17] Tiempo de finalización estimado.

- *Refinement*: Cada lunes se definen o redefinen los *tickets* del *backlog* y se priorizan. Así, cada vez que se termina una tarea el grupo sabe cuál es la siguiente.

- *Retrospective*: Cada dos semanas se convoca a todo el equipo para hacer comprobaciones relativas a la metodología y al trabajo hecho. De esta manera se detecta fácilmente la necesidad de aplicar cambios.

3.3 Métodos de validación

Con la metodología SCRUM, al final de cada *sprint* se hacía la comprobación de todas las tareas que se habían realizado y, en caso de que alguna no hubiera sido terminada —o empezada—, se estudiaba la razón y se analizaban las prioridades, bien para asignar dicha tarea al siguiente *sprint*, bien para modificar el deadline del presente *sprint*.

Además de las comprobaciones, al final de los *sprints* se realizaban reuniones con el director cada una o dos semanas para hablar del estado del proyecto y de los avances observados, y también para verificar que el procedimiento se estaba desarrollando sobre el tiempo programado.

Con SCRUM + Kanban se realiza un gráfico que se modifica diariamente antes del *Daily Stand-up*. Ello permite hacer la validación diaria del trabajo realizado.

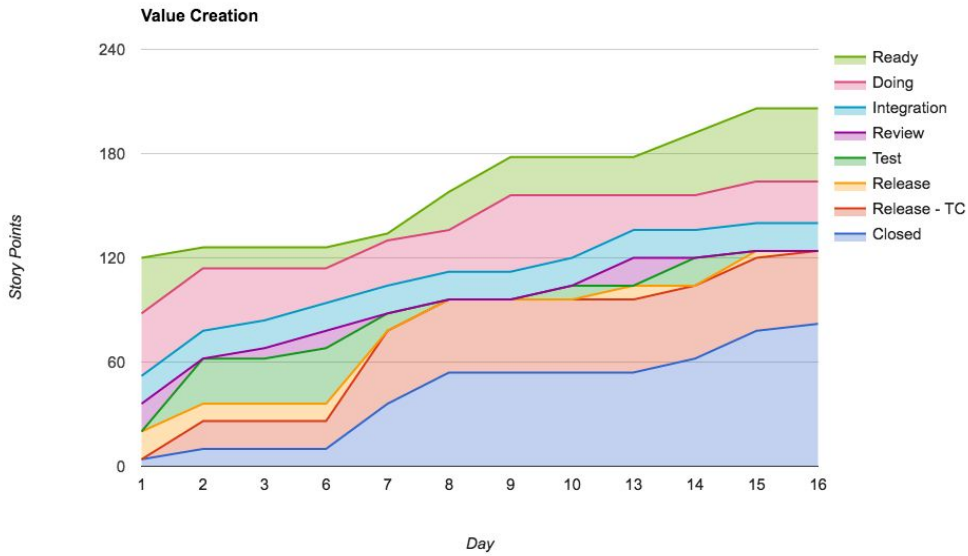
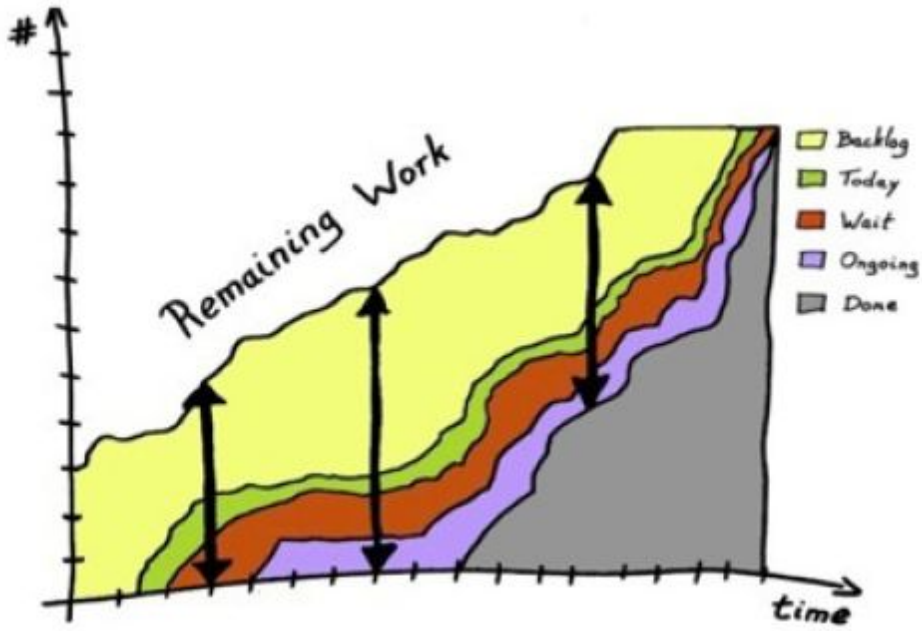


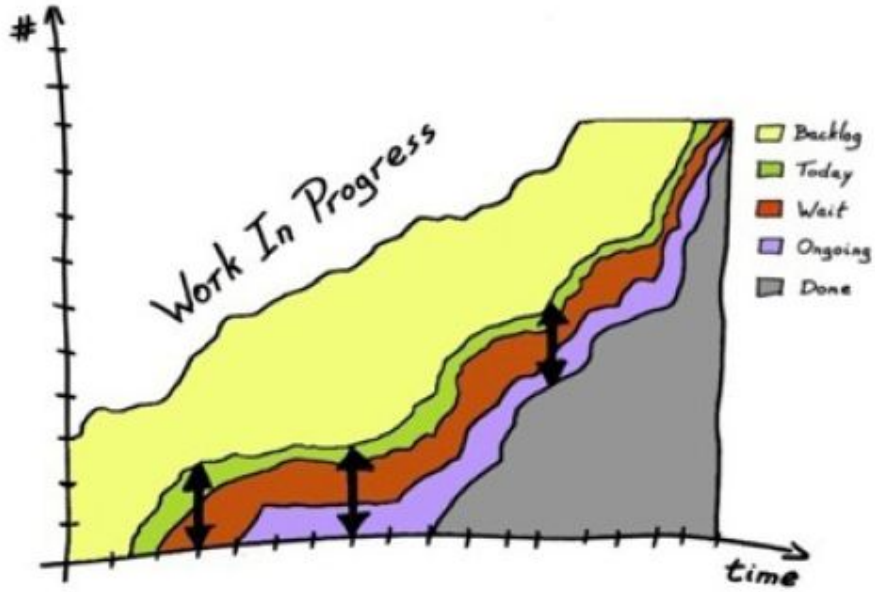
Gráfico con las estadísticas de SCRUM + Kanban

- *Closed*: Indica los *story points* de los *tickets* que ya se han subido a producción.
- *Release - TC*: Indica los *tickets* que están listos para ser subidos a la producción de Talent Clue.
- *Release*: Indica los *tickets* que están listos para ser subidos a la producción de Talentier.
- *Test*: Muestra los *tickets* que están pendientes de ser testeados.
- *Review*: Muestra los *tickets* que están pendientes de ser revisados.
- *Integration*: Indica los *tickets* que están pendientes de otros *tickets* para ser realizados.
- *Doing*: Son los *tickets* que se están realizando en ese momento.
- *Ready*: Son todos los *tickets* del *backlog* que ya están definidos y listos para ser desarrollados.

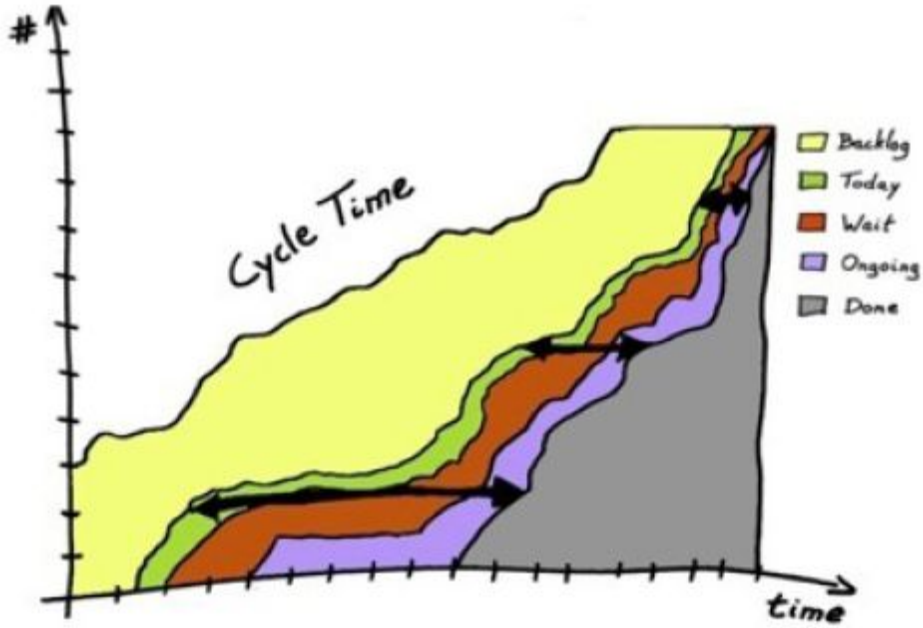
Con esta gráfica se pueden realizar las siguientes validaciones:



Trabajo pendiente por hacer



Trabajo haciéndose actualmente.



Tiempo entre comienzo y fin de las tareas.

3.4 Alcance del proyecto

El alcance del proyecto se divide en dos partes.

Para empezar, se partió de un estudio del estado inicial de la plataforma, un análisis de la misma y una definición de la arquitectura desde cero, teniendo en cuenta el nivel de abstracción necesario y la escalabilidad del sistema. Después se realizaron el diseño y la implementación de la API para que pudiera ser usada en diferentes ATS.

Por otro lado, se planteó la incorporación del uso de la API en Talent Clue, junto con el rediseño de la integración vía *pop-up* en la plataforma de Talent Clue para corregir los problemas de usabilidad existentes.

3.5 Competencias técnicas

CTI1.1: Demostrar comprensión del entorno de una organización y de sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.

Actualmente la empresa está en crecimiento y se está planteando el cambio del servicio de *hosting*, lo que será uno de los aspectos a tener en cuenta antes de la implementación del proyecto.

CTI2.2: Administrar y mantener aplicaciones, sistemas informáticos y redes de computadores.

Al ser una organización pequeña, entre todos los componentes del equipo de desarrollo se mantiene el entorno de trabajo de toda la empresa. Además, durante el proceso se ha creado un entorno con Docker para futuros integrantes del equipo de desarrollo.

CTI2.3: Demostrar comprensión, aplicar y gestionar la garantía y la seguridad de los sistemas informáticos.

Durante el desarrollo del proyecto, en todo momento se contemplan cuestiones de seguridad.

CTI3.1: Concebir sistemas, aplicaciones y servicios basados en tecnologías de red, teniendo en cuenta Internet, web, comercio electrónico, multimedia, servicios interactivos y computación ubicua.

El proyecto consiste en crear un *web-service*, por lo que se refleja la integración de esta competencia en profundidad.

CTI4: Usar metodologías centradas en el usuario y la organización para el desarrollo, la evaluación y la gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, la ergonomía y la usabilidad de los sistemas.

En Talentier se usa la metodología ágil SCRUM, lo que facilita y agiliza mucho el trabajo al equipo de desarrollo. Por otra parte, la función de este equipo es la realización de un producto definido previamente en un proceso dividido en breves tareas.

4.1 Planificación temporal

La planificación temporal sufrió pequeños cambios respecto a lo planteado al comienzo del proyecto. En este apartado veremos las desviaciones ocurridas, las causas de estas y los tiempos utilizados en cada fase.

4.2 Planificación inicial

Al comienzo del proyecto se hizo la planificación que se muestra en la **figura 1** y la **figura 2.** En la primera figura se puede ver el diagrama de Gantt de las dos primeras fases (la barra temporal gris indica el trabajo de desarrollador adicional que se realiza en la empresa Talentier).

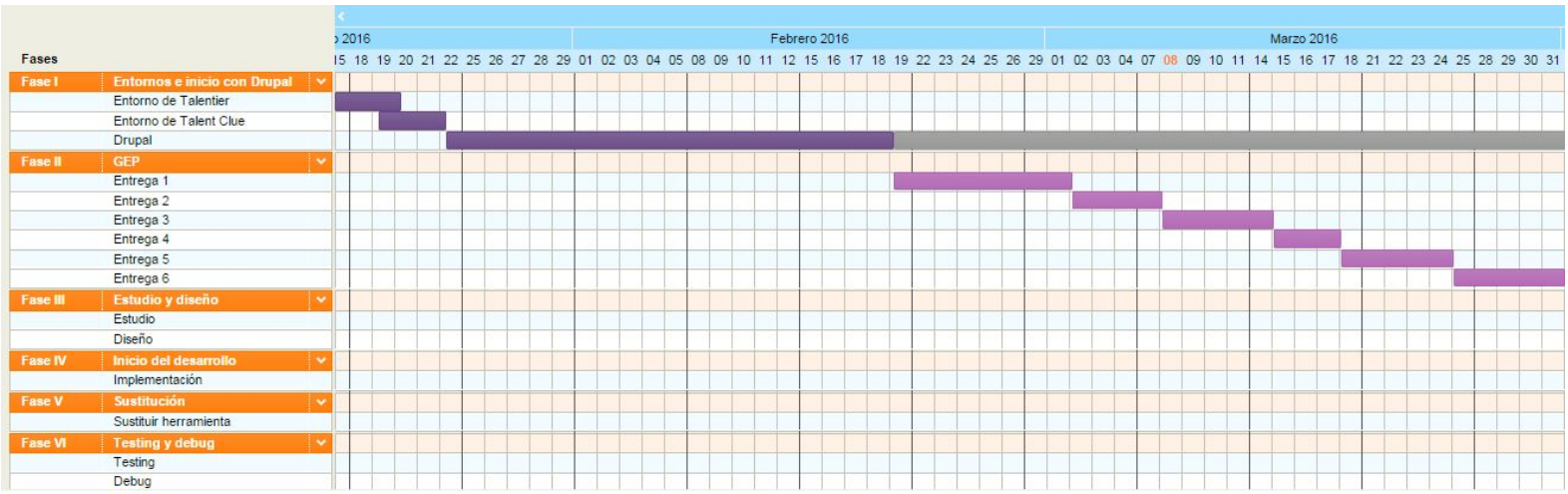


Figura 1. Gantt Fase I y Fase II

En la segunda se vuelve a apreciar la barra gris, junto a la planificación estimada para el resto de fases.

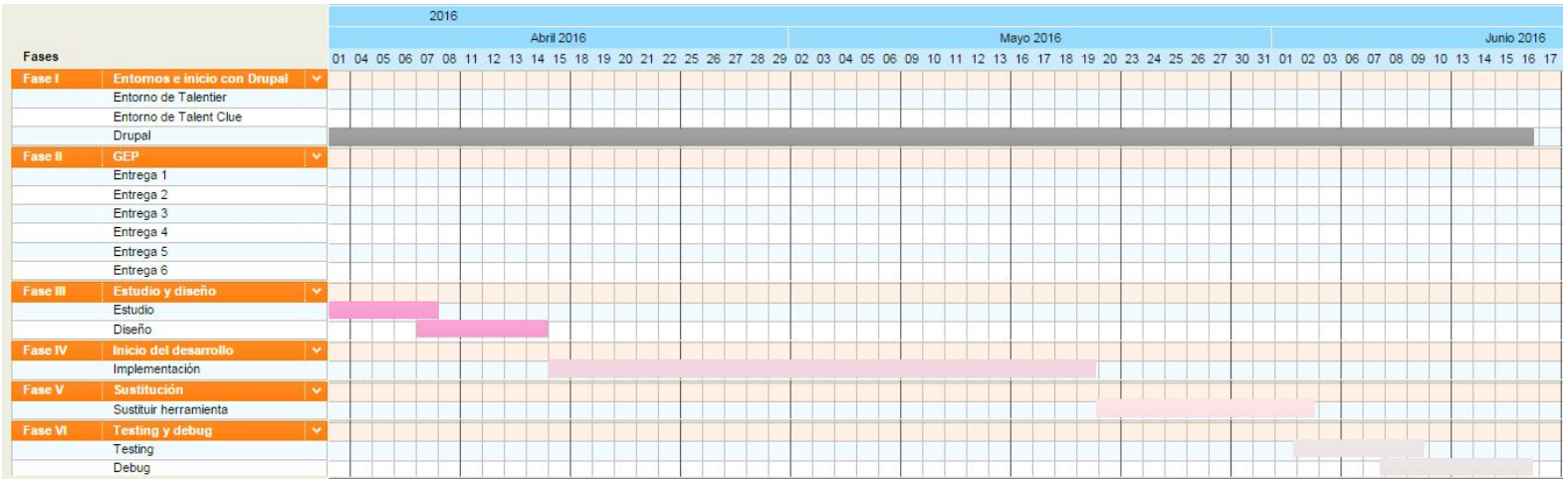


Figura 2. Gantt Fases III, IV, V y VI

4.3 Desarrollo temporal

A continuación, en la **figura 3** se pueden ver los tiempos reales del desarrollo del proyecto. Las dos primeras fases siguieron la planificación estimada, pero fue a partir de la fase 3 cuando la planificación sufrió desviaciones.

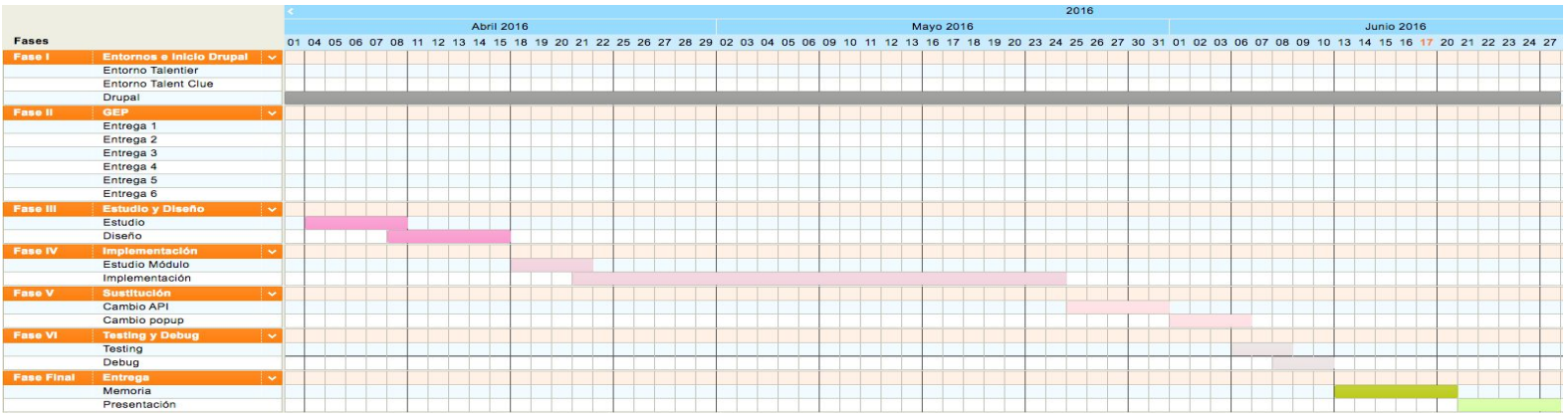


Figura 3. Gantt real de la planificación del proyecto.

La tercera fase requirió tres días más de lo planeado, lo que se transmitió al resto de la planificación.

En la cuarta fase surgió la necesidad del estudio de un módulo de Drupal para ayudar a la realización del API. La implementación terminó una semana más tarde de lo planeado, lo que supuso hacer uso de un plan de acción (el cual consistió en la inversión de más horas en las tareas que faltaban).

La quinta fase se realizó con el tiempo previsto, pero acumulando una semana de retraso, lo que hizo que la sexta fase se realizara en menos tiempo del previsto. Sin embargo, durante la planificación inicial no se tuvo en cuenta que con la metodología de trabajo *scrum-canvas*, el testeo y el *debugging* de los *tickets* se realizaban antes de considerarlos terminados. Eso permitió que la sexta fase se pudiera reducir en tiempo.

Por último, se tuvo que añadir una fase que no se había planteado al inicio, que ocupó todo el tiempo restante hasta la fecha de entrega. Dicha fase es la que consistió en documentar el proyecto y preparar la presentación del mismo.

4.4 Horas dedicadas

Fase I: Entornos	Inicio	Fin	Horas
Entrono de Talentier	15 de enero	19 de enero	21 horas
Entorno de Talent Clue	19 de enero	21 de enero	15 horas
Drupal	22 de enero	18 de febrero	120 horas

Horas por tarea de la Fase I

Fase II: GEP	Inicio	Fin	Horas
Entrega 1	19 de febrero	29 de febrero	20 horas
Entrega 2	1 de marzo	7 de marzo	15 horas
Entrega 3	8 de marzo	14 de marzo	15 horas
Entrega 4	15 de marzo	17 de marzo	5 horas
Entrega 5	18 de marzo	24 de marzo	15 horas
Entrega 6	25 de marzo	31 de marzo	25 horas

Horas por tarea de la Fase II

Fase III: Estudio y Diseño	Inicio	Fin	Horas
Estudio	4 de abril	8 de abril	15 horas
Diseño	8 de abril	15 de abril	35 horas

Horas por tarea de la Fase III

Fase IV: Implementación	Inicio	Fin	Horas
Estudio módulo	18 de abril	21 de abril	25 horas
Implementación	21 de abril	24 de mayo	125 horas

Horas por tarea de la Fase IV

Fase V: Sustitución	Inicio	Fin	Horas
Cambio API	25 de mayo	31 de mayo	8 horas
Cambio *pop-up*	1 de junio	6 de junio	20 horas

Horas por tarea de la FaseV

Fase VI: Testing y debug	Inicio	Fin	Horas
Testing	6 de junio	8 de junio	5 horas
De-bug	8 de junio	10 de junio	20 horas

Horas por tarea de la Fase VI

Fase Final: Entrega	Inicio	Fin	Horas
Memoria	11 de junio	20 de junio	50 horas
Presentación	21 de junio	27 de junio	15 horas

Horas por tarea de la Fase Final

Fase	Horas
Fase I	156 horas
Fase II	95 horas
Fase III	50 horas
Fase IV	150 horas
Fase V	28 horas
Fase VI	25 horas
Fase Final	65 horas
Total	**569 horas**

Horas totales por fases

5.1 Tecnologías usadas

Las tecnologías usadas para desarrollar el proyecto fueron las siguientes:

5.1.1 Drupal 7 (PHP)

La herramienta de Talentier está desarrollada con Drupal 7, el cual es un *framework* para la realización de páginas web basado en PHP.

Drupal es de código abierto; la primera versión se lanzó en 2001. Esto hace que sea una herramienta muy madura y con una comunidad muy grande de personas trabajando en ella. Actualmente Talentier usa la 7ª versión, a pesar de ya existe una versión más reciente. Uno de los planes futuros es hacer la migración a dicha versión.

Al ser de código abierto la comunidad va mejorando la herramienta. Esta se divide en dos partes: el *core* (incluída por defecto, se instala con cada versión de Drupal) y los proyectos (módulos aparte que la gente comparte para que los pueda usar toda la comunidad).

El core va sufriendo actualizaciones de seguridad regularmente; este es uno de los puntos fuertes de usar un *framework*. Otro es el poder hacer uso de módulos existentes para no tener que empezar desde cero tareas más complejas.

En Talentier, como ya hemos dicho, se usa el *core* de Drupal 7, y en la actualidad se trabaja en cerca de 100 proyectos de la comunidad, además de unos 30 módulos propios. Para la realización del proyecto se usó uno de los módulos de la comunidad que ayuda a la realización de API. Actualmente existen 5 proyectos distintos que ayudan al desarrollo de API.

Otras de las ventajas de Drupal es el manejo del contenido creado en la plataforma, ya que lo hace modulable y actualizable. Podemos destacar, además, sus características en cuanto al control y gestión de usuarios y roles o rutas y alias.

Cabe indicar que Talent Clue también usa la 7ª versión de Drupal.

5.1.2 MySQL

MySQL es la tecnología de bases de datos que usan tanto Talentier como Talent Clue, ya que es una de las soportadas por Drupal.

Este sistema de gestión de bases de datos sigue un modelo relacional, es decir, se basa en diversas relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados *tuplas*. Cada relación funciona como una tabla compuesta por registros (filas) y campos (columnas).

Las principales ventajas de las bases de datos relacionales son que evitan la duplicación de tablas y que garantizan la integridad referencial. En otras palabras, si se elimina un registro de una tabla se elimina en todos los demás registros dependientes.

Como principal desventaja frente a los modelos no relaciones es que su recorrido es más costoso y lento si la base es de un tamaño considerable.

Para usar MySQL en Drupal, existen métodos propios para hacer consultas, inserciones o modificaciones. Todos ellos están debidamente expuestos en la documentación de Drupal.

5.1.3 HTML/CSS/JavaScript/JQuery

Además de PHP se han usado las tecnologías comunes del desarrollo web, puesto que Talentier es una herramienta web.

El uso de estas tecnologías es constante en el desarrollo con Drupal, pero el principal uso surgió en el momento de sustituir el *pop-up* existente al comienzo del proyecto por el actual.

Más adelante, con la explicación del desarrollo de esta parte, entramos en detalle sobre la tecnología que se usó en cada momento. Aun así, a grandes rasgos, podemos decir que: HTML es la tecnología usada para definir los elementos; JavaScript y JQuery establecen las formas de actuar de los elementos definidos por el HTML, y el CSS es la herramienta para dar forma a dichos elementos.

5.1.4 JSON

Es la tecnología usada para hacer el intercambio de datos en las peticiones de la API entre plataformas (Talentier y otras ATS).

JSON, es el acrónimo de JavaScript Object Notation. Es un formato de texto ligero para el intercambio de datos. Este es un subconjunto de la notación literal de objetos de JavaScript.

Permite el intercambio de números (positivos, negativos o decimales), cadenas de caracteres, *booleanos*, valores nulos (*null*), vectores y objetos. Gracias al envío de este tipo de datos a través de la API se puede hacer el uso que se encuentre conveniente (ya sea realizar consultas, crear nuevos objetos o insertar bases de datos). Esto es posible gracias al hecho de que las tecnologías actuales son capaces de traducir los objetos en formato JSON.

Durante el proyecto, para hacer las peticiones y traducciones del API, se ha usado cURL[18]. Este permite añadir a las peticiones cabeceras y parámetros para luego traducir la respuesta de JSON a *arrays* de PHP.

[18] Este es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos.

5.1.5 Git

Git es la tecnología usada por el equipo de desarrollo para la gestión de versiones del repositorio. Es decir, durante el desarrollo cada desarrollador trabaja en una tarea distinta pero para la misma herramienta. En consecuencia, los participantes no saben exactamente qué parte del código están modificando los demás, y es por eso que se necesita un mecanismo de control para que en caso de que dos desarrolladores modifiquen a la vez la misma parte de código, los participantes sean conscientes del conflicto y puedan resolverlo.

Tanto el equipo de Talentier como Talent Clue usan git, pero los repositorios están alojados en distintos servicios de revisiones. En el caso de Talentier se usa Github y Talent Clue utiliza BitBucket.

5.2 Herramientas usadas

El equipo usado para este proyecto ha sido un Macbook Pro del 2010, con las siguientes características:

Las herramientas utilizadas durante el desarrollo del proyecto fueron:

5.2.1 Drush

Drush es una interfaz de línea de comandos del *shell*[19] para Drupal. Simplifica y acelera el uso de Drupal; mediante una serie de comandos, permite desarrollar acciones que se realizarían mediante la navegación de la interfaz de administrador en Drupal.

[19] Es el término usado en informática para referirse a un intérprete de comandos.

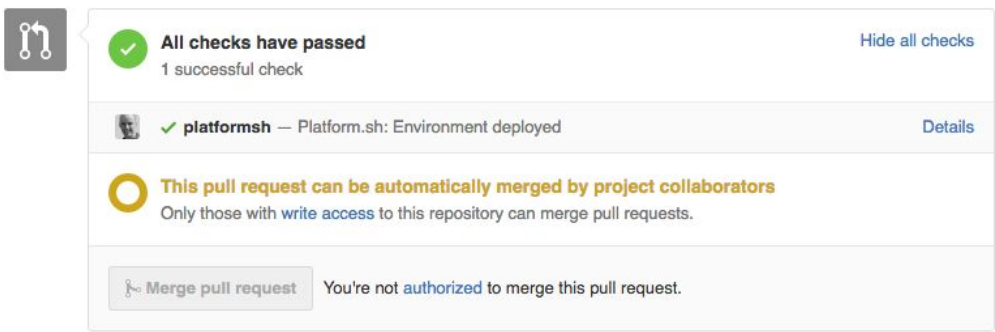
5.2.2 Pantheon/Platform:

Pantheon y Platform.sh son los dos servicios de *hosting* que se han usado en Talentier durante el desarrollo del proyecto. Ambos servicios ofrecen compatibilidad con Drupal.

Al empezar en Talentier el sistema usado era Pantheon, pero a principios de abril se decidió cambiar de servicio, ya que este ofrecía muchas mejores prestaciones.

Pantheon solo disponía de servidores en Estados Unidos, lo que hacía que la navegabilidad en la web fuese lenta. Esta fue la principal causa del cambio, pero no fue la única, ya que por ejemplo Platform permite gestionar las versiones del repositorio en pocos clicks.

Otro beneficio clave de este sistema es la creación de entornos por cada *Pull Request* creado desde Github, que permite al equipo de desarrollo hacer el testeo de las nuevas funcionalidades sin necesidad de bajar ningún código a la máquina local ni de tener que cambiar las bases de datos.

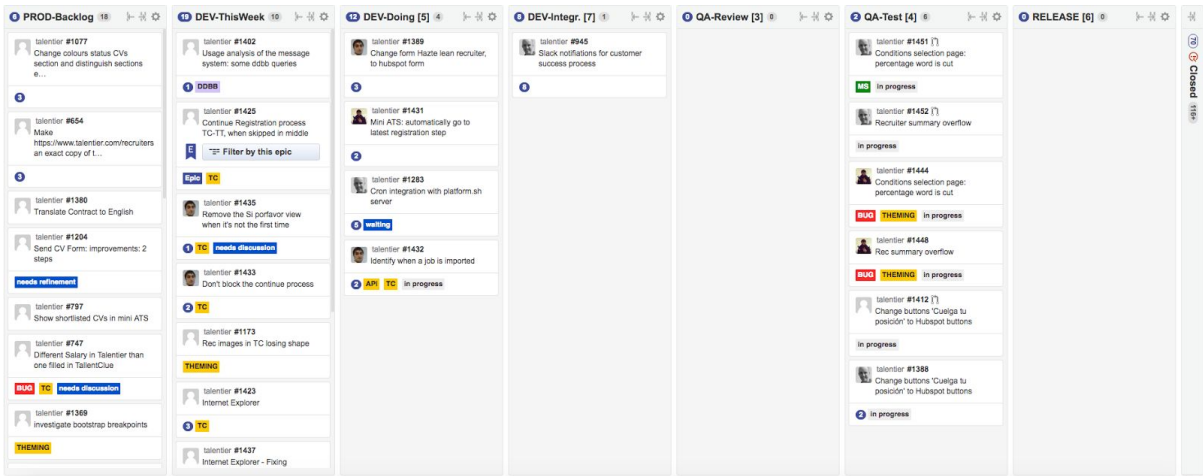


En la imagen se ve uno de los entornos que se crean en Github cuando se hace un *Pull Request*.

5.2.3 Github + ZenHub

Github es el servicio web dónde está subido el repositorio de desarrollo y ZenHub es una extensión de Github que permite crear *tickets* y gestionarlos entre el equipo de desarrollo.

Gracias a la combinación de GitHub y ZenHub, el equipo de desarrollo se organiza a diario, permitiendo visualizar la metodología de trabajo SCRUM - Canvas.



Como ya se ha explicado, la metodología consiste en que el curso del proyecto siempre fluya. La forma de representarlo es la siguiente: los *tickets* (tareas por hacer) se mueven de la columna de más a la izquierda hacia la derecha, desde *Backlog* a *Closed*.

En la Columna de *Backlog* se almacenan todos los *tickets* pendientes por hacer, ordenados de arriba abajo, de más prioritarios a menos prioritarios. En la siguiente columna (*This Week*), estarían los *tickets* previstos de realizarse esa semana (lo que equivaldría a un *Sprint* en SCRUM).

En la siguiente columna (*Doing*) ya se indica el desarrollo de los *tickets.* A partir de esta columna hay un número que indica la cantidad máxima de *tickets* que pueden estar abiertos al mismo tiempo para evitar que el trabajo se estanque en algún punto. En el caso de *Doing* solo puede haber 5 tareas como máximo en desarrollo. Esto significa que hasta que no se acabe alguno de los *tickets* ningún desarrollador puede empezar a desarrollar uno nuevo, por lo que tendría varias opciones a escoger: o bien revisar el código de algún *ticket* ya desarrollado, o bien hacer el *testing,* o, por último echar una mano en alguna tarea en la que se necesite ayuda.

Todos los tickets están diseñados para que se trate de tareas cortas (menos de un día de duración). Por lo tanto, no suele ocurrir que el flujo de trabajo se estanque en ningún punto.

También existe la columna *Integrate*, que contiene aquellos *tickets* que forman parte de una tarea más grande. Así pues, estos necesitan de más de un *ticket* para poder ser revisados y testeados.

La siguiente parte del flujo, una vez terminado el desarrollo de un *ticket*, la observamos en la columna *Review*, donde se revisa el código (estándares de lenguaje y otros) para después poderlos desplazar hasta *Testing.* en caso de no pasar el *review,* se asignará una etiqueta, como pueden ser "Necesita más calidad" o "Revisión". Los *tickets* marcados con estas etiquetas se modifican sin moverse de la columna *Review*, ya que los *tickets* solo se pueden mover de izquierda a derecha para no modificar el flujo. Una vez corregidos los fallos, se elimina la etiqueta y cuando se pasa el *Review* son movidos a *Testing*.

En la columna *Testing*, se mueve el *ticket* junto al PR[20], ya que platform.sh crea entornos de los PR, en los cuales se puede hacer el *testing* del *ticket*. En caso de encontrar algún error, se volverá a añadir una etiqueta que indique que no funciona y que se necesita revisión. Una vez corregido y pasado el *testing* se puede mover a la columna *Release*.

En la columna *Release* están todos los *tickets* terminados que pueden incorporarse a la plataforma a la que tienen acceso todos los usuarios. Los cambios siempre son subidos a primera hora de la mañana. Una vez subidos los *tickets* se cierran y quedan registrados en la última columna de la derecha (*Closed*).

[20] *Pull Request.*

5.2.4 SourceTree

Es el *software* que usa el equipo de desarrollo para gestionar git con una interfaz gráfica. Este permite varios repositorios al mismo tiempo, tanto de Github como de BitBucket.

Es la forma visual de crear *branches, pushes* o *pulls* del repositorio o de resolver *merge conflicts*.

Una funcionalidad práctica que tiene es la posibilidad de crear *stashes*, una forma de guardar modificaciones de código sin subir para poder cambiar de *branch* y poder recuperar dichos cambios más adelante.

5.2.5 Docker

Docker es un sistema para configurar entornos seguros y reproducir las mismas condiciones de producción para todos los desarrolladores, reduciendo así tiempos de pruebas y adaptaciones en el entorno de producción.

Docker es un proyecto de código abierto con el que se pueden crear "contenedores" (máquinas virtuales[21] ligeras). Estos contenedores también se pueden descargar de un repositorio general. Por ejemplo, en Talentier no se ha creado ninguno nuevo, sino que se usan contenedores creados previamente.

El uso de Docker se automatizó creando unas *scripts*[22] que se ejecutan al iniciar los equipos.

[21] Software que simula un equipo informático.
[22] Archivo que contiene comandos.

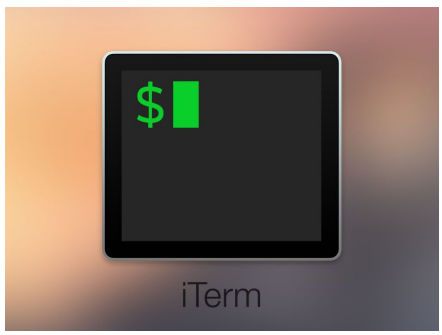
5.2.6 MySQL Workbench

Es el software para gestionar la base de datos con interfaz gráfica. Permite gestionar tanto la base de datos de Talentier como la de Talent Clue a la vez. A pesar de que se pueda modificar vía la aplicación, muy pocas veces modifica manualmente.

5.2.7 Sublime Text

Es el editor de texto usado, que permite instalar extensiones que facilitan el desarrollo (como por ejemplo ver los cambios realizados en el *branch* o seleccionar el lenguaje del archivo para poder diferenciar mejor por colores las clases, objetos, *strings*, etc.).

5.2.8 iTerm

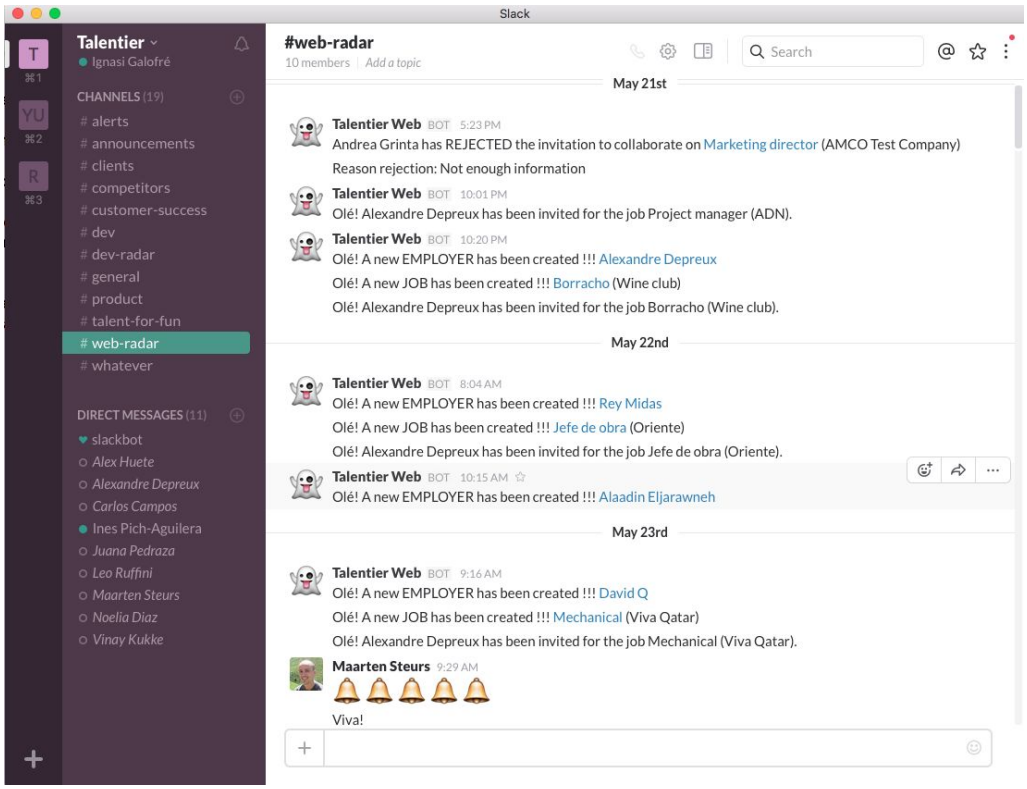


Software de terminal que facilita el funcionamiento con diversos *shortcuts* y estilos.

5.2.9 Slack

Software de comunicación que se usa en Talentier para hablar todo el equipo, mediante distintos grupos de trabajo (*Marketing*, Dev, etc.) además de comunicación uno a uno con cada miembro de la empresa.

Además este servicio permite integrar triggers[23] en la web que te comunica avisos como por ejemplo, cuando se registra un usuario o se crea una oferta de trabajo.



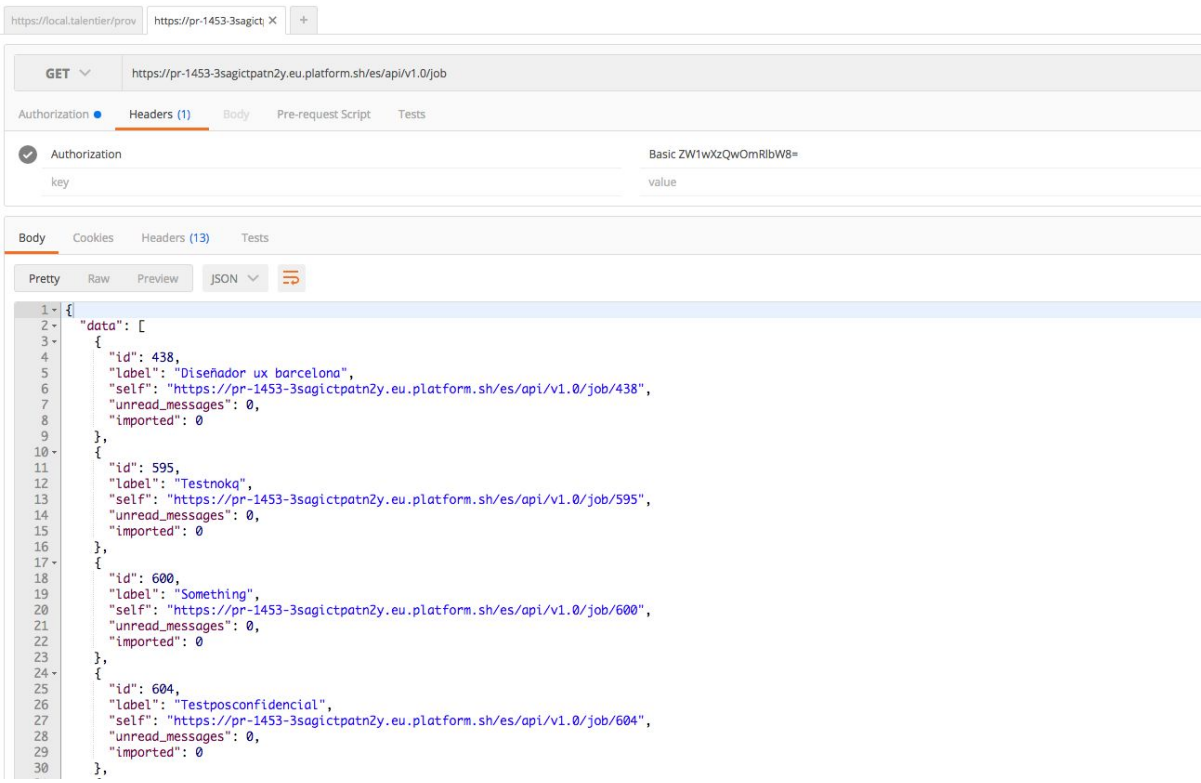
Ejemplo de una vista del Slack de Talentier.

Slack también permite usar *addons*[24], como por ejemplo la creación de videollamadas a través de Appear[25].

[23] Desencadenante.
[24] Extensiones de software.
[25] Software online de videollamadas.

5.2.10 Postman

Postman sirve para testear los *enpoints* de la API y ayudar a documentarla. Con este *software* se puede probar cualquier tipo de petición (GET, POST, PUT, PATCH, DELETE). También permite añadir los *headers* y parámetros POST que se deseen.



Vista de un ejemplo de un endpoint.

6.1 Desarrollo de la Fase I

El desarrollo del proyecto se llevó a cabo en un entorno Mac. Para empezar a trabajar en Talentier, y en el proyecto en sí, se debían preparar los entornos de trabajo, tanto el del propio de Talentier como el de Talent Clue, porque el equipo de Talentier tiene acceso al repositorio de Talent Clue para hacer modificaciones en la integración con la otra plataforma.

Es importante destacar que ambas compañías usan 4 entornos distintos para el desarrollo de las plataformas. Estas son:

- **LIVE:** El entorno de producción, donde tienen acceso los clientes/usuarios de las plataformas.

- **DEV:** En este entorno es donde se hacen las pruebas de todas las nuevas funcionalidades antes subirlas a producción.

- **TEST:** De este entorno solo hace uso Talentier, a pesar de que Talent Clue también lo tiene activo. En Talentier se usa para crear *demos* para los clientes.

Los 3 entornos mencionados son configurados en el servidor donde se hospeda el servicio web.

- **LOCAL:** Este es el entorno donde trabajan los desarrolladores y tiene que ser configurado antes de empezar el desarrollo.

6.1.1 Entorno de Talentier.

Para preparar el entorno de Talentier se debe instalar Drush en primer lugar, crear una carpeta y descargar Drupal 7 con el comando:

```
drush dl drupal-7.x
```

Después, con el archivo Makefile del proyecto de Talentier, se debe ejecutar el siguiente comando, *drush*:

```
drush make --no-core talentier.make
```

Con estos comandos se consigue generar el código de los módulos propios de Drupal y los proyectos de la comunidad. Para añadir los módulos personalizados hay que descargar el repositorio de Github. Antes, sin embargo, hay que descargar el software Source Tree, que es el sistema de Git que se ha usado para el desarrollo del proyecto. Una vez descargado se debe sincronizar el repositorio en la carpeta /sites/default.

A continuación hay que instalar un *docker* para configurar los entornos de la máquina local para que coincidan con los del entorno de producción.

Una vez instalado el *docker*, copiamos el archivo fig.talentier.yml (archivo que contiene la configuración del entorno) también en /sites/default.

Para arrancar los entornos es necesario crear una *script*, debido a que cada vez que se enciende el equipo se debe arrancar el entorno. El contenido de este script es el siguiente:

```
#!/bin/bash

cd Talentier/httpdocs/sites/default

docker-machine start tt

eval $(docker-machine env tt)

docker-compose -p tt -f fig.talentier.yml up -d

docker run --add-host local.talentclue:192.168.99.100 referup/apache cat /etc/hosts
```

El siguiente paso es crear una base de datos en el entorno creado por el *docker*. Para eso primero se debe entrar en el container mysql, crear la base de datos y copiar una base de datos de producción. Antes de esto se debe copiar el archivo talentier.sql en la carpeta docker/data/mysql/master. Seguidamente, hay que entrar en el container mysql con el comando:

```
docker exec -it tt_mysql_1 bash
```

Una vez dentro creamos la tabla con los comandos:

```
TERM=linux mysql -uroot -p
create database talentier;
exit;
TERM=linux mysql -uroot -p talentier < talentier.sql
```

Para acabar la instalación del entorno hay que añadir en el archivo *hosts* la siguiente línea:

```
192.168.99.101  local.talentier
```

6.1.2 Entorno de Talent Clue

Para añadir el entorno de Talent Clue se debe seguir el mismo procedimiento que para el de Talentier, pero con los archivos propios de Talent Clue, módulos, *docker* y base de datos.

6.1.3 Drupal

Anteriormente nunca había trabajado con Drupal así que le tuve que dedicar bastantes horas de aprendizaje, para el aprendizaje usé la documentación de Drupal y un libro proporcionado por el director, además de tener el soporte del director para cualquier duda.

Durante este aprendizaje se estudió desde como crear usuarios o añadir rutas a cómo editar los módulos de la comunidad o crear módulos propios.

Una vez hecho el estudio preliminar de Drupal que duró unas dos semanas, se empezó a navegar por los menús de las plataformas de Talentier y Talent Clue, también a hacer el primer vistazo al código de estas. Hasta la fecha no había entrado en detalle con respecto a las características de la plataforma. Unos días más tarde se empezaron a realizar las primeras tareas sencillas, como modificar formularios o campos de perfiles de usuario. Otra semana más tarde ya se empezó con el trabajo de desarrollo, como el que se está realizando actualmente, ya entrando en la metodología de trabajo del equipo de desarrollo.

Como ya hemos mencionado anteriormente, Drupal tiene una curva de aprendizaje elevada. Por esta razón conviene empezar desde lo básico y avanzar poco a poco antes de entrar en la creación de contenido.

6.2 Desarrollo de la Fase II

Esta fase fue dividida en 6 partes, todas ellas definidas por la asignatura GEP. La realización de esta iba a servir para hacer un primer análisis de tiempos, costes y un primer *overview* del proyecto, que ayudaría a la fase final de presentación del proyecto, ya que se podría hacer una comparación real de lo planificado y lo ocurrido realmente.

GEP está dividida en la siguientes entregas:

- Entrega 1: Definición del alcance y contextualización.
- Entrega 2: Planificación temporal.
- Entrega 3: Gestión económica.
- Entrega 4: Presentación preliminar.
- Entrega 5: Especialidad.
- Entrega 6: Documento Final + Presentación.

6.3 Desarrollo de la Fase II

La tercera fase fue dividida en dos partes diferenciadas. La primera iba enfocada al estudio actual del sistema, los mecanismos que se usaban para realizar la integración con Talent Clue y cuál tenía que ser el resultado del proyecto y con qué fin. Una vez terminado el estudio se tenía que buscar la mejor solución y definir paso a paso qué acciones se tenían que realizar durante la fase de implementación.

Para entender el propósito de la funcionalidad del API es importante entender el proceso al externalizar una búsqueda de candidatos con Talentier a través de un ATS. En la plataforma del ATS se crea la oferta de trabajo, una vez creada, si se desea buscar candidatos a través de Talentier se debe exportar la oferta a Talentier, para que los recruiters puedan ver todas las condiciones y requisitos necesarios. Una vez los recruiters envían currículums de candidatos, es importante saber el estado de estos, es decir, si se han descartado como candidatos o si se ha considerado como seleccionado, por esto otra de las funcionalidades a tener es saber el estado de estos. Por último si quieren contratar a uno de los candidatos propuestos por un recruiter, es necesario saber cuál será el sueldo para saber el porcentaje que se lleva el recruiter, por lo que también se tiene que exportar la oferta para que el recruiter pueda aceptar o no las condiciones desde Talentier.

Una vez descrito el proceso de externalización ya se puede entrar en más detalle de cuáles eran los puntos a tener en cuenta durante el estudio.

6.3.1 Estudio

Para empezar la fase del estudio se hizo una reunión con el director, este fue el encargado de implementar la anterior versión de la comunicación de Talentier con Talent Clue. Esta reunión sirvió para saber cómo estaba hecho anteriormente y hacia donde tenía que ir el desarrollo.

Durante el proceso de externalización se identifican los tres resources[26] en los que se centra el API, estos son los Jobs (ofertas de trabajo que son publicadas), CV (curriculums enviados por los recruiters) y CP (Candidate Placement, proposición de contratación de una empresa a un candidato del recruiter).

En la integración anterior se usaban dos módulos de Drupal (Services y Migrate), uno de los problemas era que Migrate estaba configurado para que sólo funcionase con la conexión de Talent Clue, lo que no hacía que fuera escalable con otras plataformas. Por otra parte se podría simplificar la conexión usando un solo módulo.

El estudio se empezó buscando módulos de Drupal que ayudarán en el desarrollo de APIs, los diferentes módulos encontrados fueron:

- Services: Era el que se usaba anteriormente, es el más usado por la comunidad. Este usa el protocolo SOAP[27] para responder las peticiones, donde las respuestas son más largos que los usados por REST[28]. Este fue el punto flojo que tenía usar el módulo ya que con tanto apoyo de la comunidad dispone de una buena documentación.

- RestWS: Este módulo era poco usado por la comunidad y su última actualización era muy antigua. Además de tener poca documentación, lo que iba a suponer una difícil implementación.

- Endpoint: Endpoint tenía pocas funcionalidades y poca documentación, por ejemplo no estaba documentado cómo comprobar autenticaciones entre otras cosas.

- RESTful: La opción escogida. Tenía muy buena documentación, usable para cualquier plataforma y buen rendimiento gracias al uso de la arquitectura REST. Además era el segundo módulo más extendido en la comunidad para la realización de APIs. Por otra parte este módulo es moderno y dispone de actualizaciones más recientes.

[26] Objetos con tipo y datos asociados sobre los que actúa el API.
[27] SOAP es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP.
[28] (Representational State Transfer), es un estilo de arquitectura de software dirigidos a sistemas distribuidos.

Cabe indicar que Drupal 8 ya dispone de soporte REST y documentación para realizar APIs personalizadas.

Una vez decidido el módulo que se iba usar, se enumeraron todas las funcionalidades que tenía anteriormente el API más nuevas que se podían incorporar. Después de enumerar las funcionalidades y teniendo los resources claros ya se podía pasar a la fase de diseño del API.

En cuanto a los problemas de usabilidad del popup, también se enumeraron para tenerlos en cuenta y diseñar una solución. Los problemas que se tuvieron en cuenta fueron; el tamaño, ya que era solamente un cuadro central relativamente pequeño. la aparición del navbar[29] de Talentier, este quitaba espacio al resto del popup. No era fijo, es decir que si se hacía scroll hasta abajo luego se podía seguir haciendo scroll en la pantalla del background y por último a veces no se veía el botón de cerrar, ya que era pequeño y a veces no se identificaba suficiente.

6.3.2 Diseño

Al comienzo de esta tarea se volvió a hacer una reunión con el director para verificar que la idea pensada le parecía bien al equipo de Talentier. Durante el estudio se contemplaron muchas funcionalidades que podrían ser añadidas, cómo obtener listas de recruiters, listas de recruiters filtradas por países en los que trabajan o sectores que dominan. O listas de posiciones también con filtros y muchas más. Peró se decidió centrarse en que fuese funcional para la integración con otras ATS y que fuese actualizable en un futuro.

La primera tarea antes de definir los endpoints fue decidir que tipo de autentificación se necesitaría para poder usar el API. Cómo el módulo RESTful ofrecía la posibilidad de usar autenticación básica (username + password) o token de sesión que éste a su vez necesita de autenticación básica, se decidió usar el token, ya que era un poco más seguro.

[29] Plantilla web que permite añadir enlaces.

Justo después de tener claro el punto de la autentificación se pasó a definir los endpoints del API que se quería diseñar:

Endpoints para el resource Job:

- Crear una posición:
    - Justificación: Este endpoint sería usado para exportar las ofertas de empleo de una ATS a Talentier, para que los recruiters pudieran ver las condiciones de la oferta y aceptar o rechazar la invitación.
    - Tipo: POST. Debido a que tendría que crear un nuevo objeto en la plataforma.
    - Datos de entrada: Objeto con los datos de la posición.
    - Datos de salida: Identificador de la posición otorgado en Talentier.
    - Autenticación: Token.

- Actualizar una posición:
    - Justificación: Si desde el ATS se hace alguna modificación como fecha límite o descripción de la posición, es importante que el recruiter sepa de los cambios.
    - Tipo: PATCH. Debido a que se modifica un objeto ya existente.
    - Datos de entrada: Identificador de la posición, más un objeto con los datos que se desea modificar.
    - Datos de salida: No.
    - Autenticación: Token

- Obtener los mensajes no leídos de una posición:
    - Justificación: Esta fue una de las funcionalidades extras añadidas, permitía poder saber si se tenían mensajes sin leer relacionados con la posición.
    - Tipo: GET. Esta es solo una consulta que te devuelve el número de mensajes no leídos.

    - Datos de entrada: Identificador de la posición.
    - Datos de salida: Número de mensajes sin leer de la posición.
    - Autenticación: Token.

Endpoints para el resource CV:

- Actualizar estado de un CV:
    - Justificación: Es necesario para un recruiter saber el estado de los candidatos para saber cómo avanza el proceso.
    - Tipo: PATCH. Este modifica el estado (aceptado, rechazado, etc.) del CV.
    - Datos de entrada: Identificador del CV, más nuevo estado del CV.
    - Datos de salida: No.
    - Autenticación: Token.

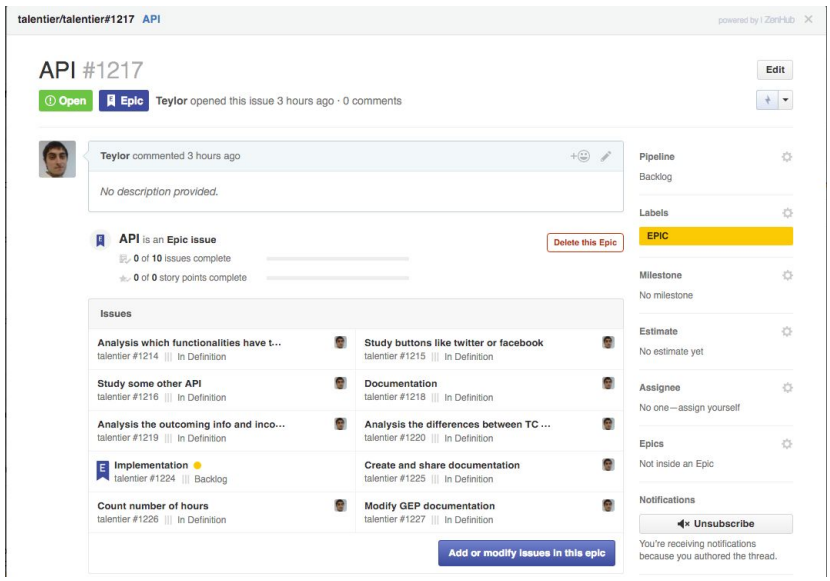
Endpoints para el resource CP:

- Crear un CP:
    - Justificación: La estructura de Talentier guarda las proposiciones de candidatos como un nodo de Drupal, por lo que se tiene que tratar por separado de la posición. Y sirve para que el recruiter acepte o rechace la oferta de la empresa al contratar uno de sus candidatos.
    - Tipo: POST. Crea un nuevo objeto CP, con los datos pasados por parámetro que contienen el sueldo propuesto por la empresa a el candidato propuesto por el recruiter.
    - Datos de entrada: Identificador de la posición, más identificador del CV y los datos de la contratación propuestos.
    - Datos de salida: Identificador del CP otorgado en Talentier.
    - Autenticación: Token.

- Actualizar un CP:
    - Justificación: Si un recruiter rechazara las condiciones de contratación estas podrían sufrir modificaciones.
    - Tipo: PATCH. Modificación del objeto CP con los nuevos datos.
    - Datos de entrada: Identificador del CP, más los datos de contratación.
    - Datos de salida: No.
    - Autenticación: Token.

Solamente se diseñaron y desarrollaron las funcionalidades necesarios para realizar el proceso de externalización, todas las posibles funcionalidades propias de un API como por ejemplo, consultas de posiciones creadas en Talentier, lista de recruiters disponibles etc. no se contemplaron para este proyecto, pero no se descarta de ser añadidas en un futuro.

Una vez definidos los endpoints se creó en ZenHub un Epic[30] con los tickets de lo que sería el desarrollo del API, definiendo los detalles de cada tarea a realizar.



Primeros tickets creados en GitHub

[30] Ticket de GitHub que contiene Subtickets.

En cuanto al diseño del nuevo *pop-up*, para corregir los problemas de usabilidad se diseñó una solución que corregía todos los problemas encontrados. Esta consistía en crear un *template* personalizado que eliminara el *navbar* y para que se pudiesen tratar los elementos por separado, además de crear un nuevo *iframe*[31] más grande siempre ajustado a cualquier monitor con el fondo gris y el botón de cerrar fuera de este.

[31] Elemento web que permite cargar otra página web en su interior.

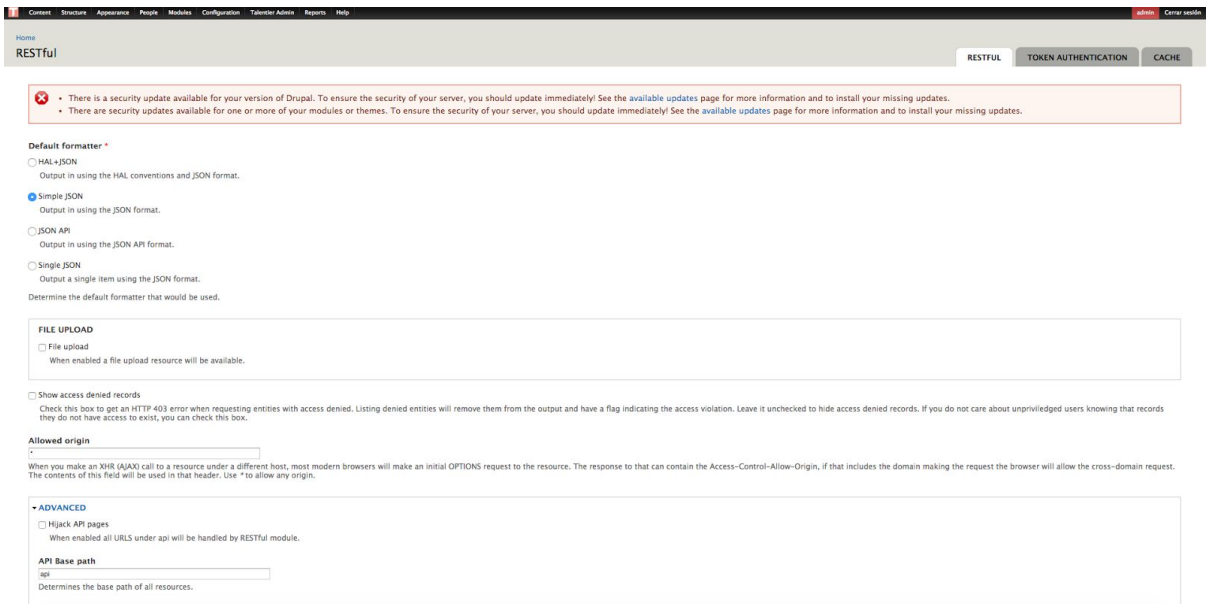
6.4 Desarrollo de la Fase IV

Una vez documentada la API y definidos los tickets era momento de empezar la implementación. Para eso se empezó aprendiendo a usar el módulo RESTful de Drupal, ya que este disponía de una buena documentación.

6.4.1Módulo

La instalación del módulo se realizó con Drush, éste a su vez instala todas las dependencias con otros módulos. También se añadió el módulo al Makefile de Talentier y al hook[32] update, para que al ser mergeado en el repositorio se instalará automáticamente en el entorno de producción.

Una vez instalado este habilita un nuevo menú, sólo accesible en modo administrador, para poder configurar cosas como la ruta base del API o que tipo de peticiones aceptar.



Lo primero que se estudió del uso del módulo era como crear resources y endpoints. Para esto se practicó el simple hecho de crear un endpoint de un

[32] Recurso software usado en Drupal para hacer de trigger.

resource Job sin ningún tipo de autentificación. Lo que llevó a entrar al estudio de la autentificación.

Para autenticar el uso del API, el módulo RESTful daba soporte a autenticación básica, token o cookie. Para hacer uso de la autenticación por token se tiene que hacer uso de un endpoint que te ofrece por defecto el módulo RESTful el cual necesita la autenticación básica de un usuario registrado, una vez estudiado el funcionamiento básico se pasó a la implementación empezando por el tema de la autenticación.

6.4.2 Implementación

Una vez probado el módulo y comprobar que todo lo planificado era posible de realizarse con él, se procedió a implementar la API diseñada.

El primer punto a tratar fue el tema de la autenticación, ya que con el estudio previo del módulo no se acabó de probar la autenticación por token, que fue la escogida en el diseño. Su implementación constó de tres partes; primero se creó un rol de usuario llamado api, luego un usuario llamado api y tercero se le dió acceso a los resources Job, CV y CP. Este es el usuario que se tiene que añadir en las ATS para tener acceso al API de Talentier.

Aunque la solución fue sencilla, el tema de la autenticación llevó mucho más tiempo del previsto, ya que en un principio no se contempló la posibilidad de crear un usuario para el API, y se llegó a pensar volver a la fase de diseño para crear nuevos endpoints para registrar nuevos usuarios a la plataforma de Talentier y estos tener acceso a sus propios resources, pero al final se dió con la solución que a su vez se considera la más replicable posible.

Una vez superado el problema de la autenticación ya se comenzó con el desarrollo de los endpoints. Se siguió el orden del proceso de externalización

primero la exportación y modificación de un Job, cambiar los estados de los CV, creación y modificación del CP y por último la funcionalidad extra de los mensajes.

La realización de los endpoints fue sencilla gracias a que el módulo permitía crear resources propios o usar nodos de drupal, cómo los tres resources ya formaban parte de la estructura de nodos de Talentier fue más sencillo. Fue simplemente implementar las funciones y asociarlas a cada método del endpoint.

Con la realización de la funcionalidad extra para ver los mensajes sin leer se añadió una notificación en forma de globo al botón de mensajes con el número de mensajes sin leer.

Funcionalidad extra de mensajes no leídos

Una vez acabados los tickets del API, se empezó con la reforma del popup, para esto lo primero que se hizo fue crear un template personalizado, este lo se asignó a cualquier ruta de la web que acabara con /ttbox, lo que permitía tratar cualquier elemento por separado, simplemente comprobando el último elemento de la ruta del navegador. Esto conllevaba cambiar todos los enlaces en el código de Talent Clue, tarea que se realizó en el momento de la sustitución. Cómo el popup no permitía volver a la pantalla anterior también se integró el botón atrás en el template.

Con estos cambios no se corregían todos los problemas de usabilidad del popup, pero por cuestión de deadline se decidió pasar a la fase de sustitución y si durante la sustitución daba tiempo a cambiar más cosas se harían durante esa.

6.5 Desarrollo de la Fase V

En la fase de sustitución no solo se sustituyó el anterior sistema de comunicación por la API, sino que también se terminaron de corregir los errores de usabilidad del popup integrado en Talent Clue.

6.5.1 API

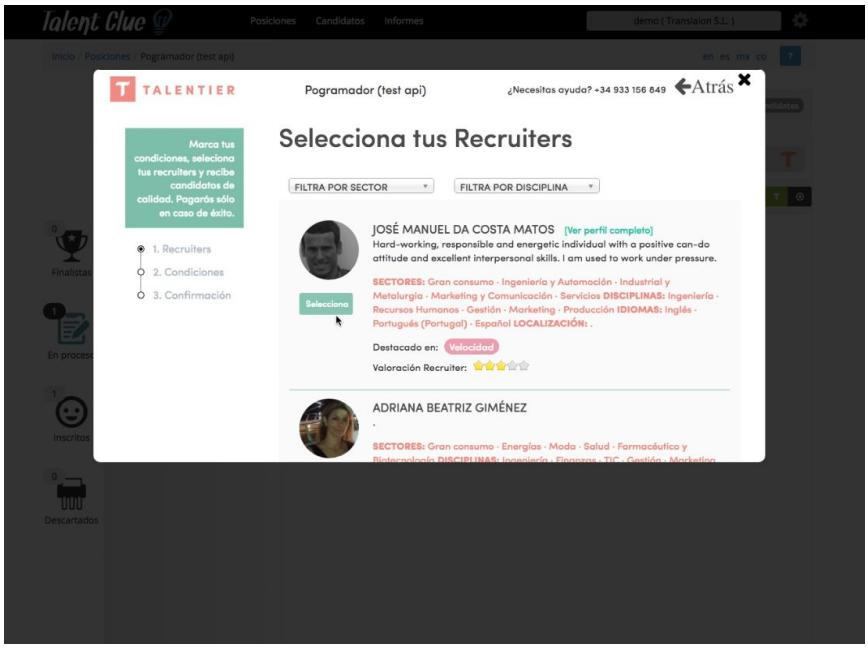
El primer paso que se siguió para hacer uso del API desde Talent Clue fue añadir el password del usuario api en un archivo de configuración del servidor, para poder hacer la autentificación vía token.

Una vez incorporada la autenticación se procedió a hacer una prueba de una llamada cualquiera en el entorno de producción. Una vez comprobado que los resultados eran los deseados, se fueron buscando y sustituyendo una a una las llamadas de la integración anterior por las llamadas a la nueva API. La tarea de testeo de estos cambios no fueron probados hasta la sexta fase del proyecto, ya que formaban parte del último ticket del proyecto.

Cómo Talent Clue usa Drupal, es decir PHP se optó por usar cURL, para hacer las llamadas al API. Cabe indicar que cURl está disponible en 30 lenguajes de programación.

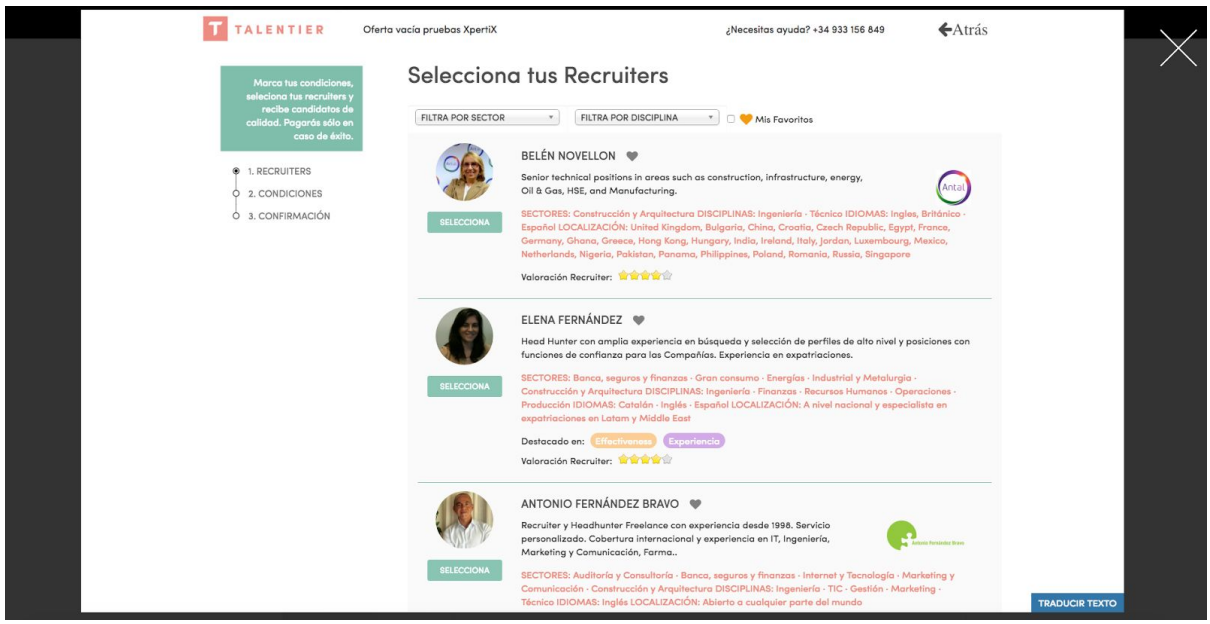
6.5.2 *Pop-up*

Como la sustitución del API no costó mucho tiempo y el testeo y debug de la sexta fase iba a ser corto, se pudo acabar de arreglar los últimos fallos de usabilidad encontrados anteriormente en esta fase.



Popup antes de mejoras

Para ello se modificó el tamaño por uno más grande, lo que hacía que los elementos no estuvieran tan compactos. Además el botón de cerrar que anteriormente era pequeño y a veces poco identificable, se trasladó fuera del popup y también aumentó el tamaño, volviéndolo más visible. También se aprovechó para añadir un efecto para saber que se estaba cargando el popup, ya que anteriormente se quedaba de color blanco mientras se esperaba.



Popup después de las mejoras

6.6 Desarrollo de la Fase VI

Como se ha mencionado con la metodología de trabajo se testeaban los tickets antes de considerarlos terminados, no fue hasta el último ticket de la sustitución que se consideró fase de testeo y debug, ya que además se iba a probar todo el proceso completo muchas y muchas veces, con todos los navegadores y versiones posibles, ya que anteriormente desarrollando otra cosa nos surgieron problemas de compatibilidad con Internet Explorer.

Testeando la parte del API solamente surgió un pequeño error que fue solucionado rápidamente, el error era fue que no cambiaba bien el estado del CV cuando se deseaba descartar, pero simplemente fue cambiar los parámetros POST que no estaban correctamente.

Finalmente testeando el popup si que se encontraron varios errores que se tuvieron que corregir, como por ejemplo que una vez cerrado no se podía abrir hasta que se refrescara la página. Pero todos los errores encontrados fueron corregidos correctamente y a tiempo.

6.7 Desarrollo de la Fase Final

La fase final no se incluye en el desarrollo, pero sí que iba a suponer un gran tiempo de realización, así que se ha considerado como una fase más del proyecto a realizar.

7.1 Gestión económica

En esta sección se explicará cómo se calculó el presupuesto estimado y se desglosará el precio real del proyecto.

Para realizar el presupuesto estimado se partió de dos premisas, intentar no superar los 10.000€ y usar el máximo de herramientas gratuitas posibles, para abaratar costes, ya que Talentier es una startup que a pesar de que ya genera ingresos todavía es una empresa pequeña y no consta de mucho presupuesto.

7.2 Identificación de recursos

Los recursos necesarios se dividieron en los siguientes grupos:

- **Recursos humanos:**

  Son las personas que tenían que llevar a cabo el proyecto.

  - Diseñador e implementador: Sueldo por trabajar con el convenio de la Universidad Politécnica de Cataluña. Encargado de diseñar e implementar la API.

- **Oficinas:**

  Lugares dónde se llevó a cabo el proyecto, ambás oficinas incluyan los costes de electricidad y de conexión a internet.

  - Oficina de Talentier: Lugar donde trabaja todo el equipo de Talentier, esta està en La Salle Bonanova.

- Oficina de TalentClue: Desde el mes de Abril se tenía pensado alquilar dos mesas en las oficinas de Talent Clue, por si se necesitaba de cooperación o reuniones con ellos.

- **Hardware[33]:**

  Serían todos los recursos hardware necesarios para llevar a cabo el proyecto.

- Ordenador portátil: La herramienta para llevar a cabo el desarrollo del proyecto.

- Servidor: El alquiler del servidor donde se hostea la web de Talentier, esta tendría un coste mensual.

- **Software[34]:**

- Drupal: Framework con el que se ha desarrollado la web de Talentier.

- Drush: interfaz de línea de comandos del shell para Drupal.

- GitHub: Servicio web en el que está el repositorio de Talentier.

- ZenHub: Extensión de GitHub usado para llevar a cabo la metodología de trabajo.

- Source Tree: Software usado para manejar git con una interfaz gráfica.

- MySQL Workbench: Software para gestionar la base de datos con interfaz gráfica.

- Sublime: Editor de texto usado.

[33] Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.
[34] Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

- iTerm: Software de terminal que facilita el funcionamiento con diversos shortcuts.

- Slack: Software de comunicación que se usa en Talentier.

- Postman: Software para hacer pruebas del API.

- Docker: Software para crear entornos de desarrollo similar al de producción.

- **Imprevistos:**

Serían los costes no tenidos en cuenta en un inicio, el presupuesto estimado para los imprevistos es la diferencia del total menos el presupuesto máximo establecido en unos 10.000€.

7.3 Estimación de los costes

- **Recursos humanos:**

    - Sueldo del desarrollador: Según el convenio realizado entre Talentier y la Universidad Politécnica de Cataluña, este cobrará 960€ mensuales.

- **Oficinas:**

    - Oficina de Talentier: El coste de esta oficina es de 800€ al mes.

    - Oficina de Talent Clue: El alquiler de dos mesas por las tardes a partir del mes de Abril es de 150€ al mes.

- **Hardware:**

    - Equipo de desarrollo: Este es un Macbook Pro que pertenece al estudiante y no supone ningún coste al proyecto.

    - Servidor: El servidor de Pantheon tiene un precio mensual de 100€ al mes.

- **Software:**

Todo el software usado es gratuito, Sublime dispone de versión de pago, pero no se adquirió la licencia.

En la siguiente tabla se puede ver coste previsto del proyecto:

Recursos humanos	**Sueldo** 960€ (5 meses)
Oficinas	**Talentier** 800€ (5 meses) + **Talent Clue** 150€ (3 meses)
Hardware	**Servidor** 100€ (5 meses)
Software	0€
Total	**9.750€**

Aproximadamente 250 € para imprevistos.

7.4 Coste del proyecto

Una vez finalizado el proyecto se ha podido calcular el coste total de este. Este se ha desglosado por precio al mes. El único imprevisto no calculado al principio del proyecto fue el cambio del servidor, pero este resultó ser más barato. Por lo tanto el precio total fue menor del pensado inicialmente.

Febrero:

Recursos humanos	**Sueldo** 960€
Oficinas	**Talentier** 800€
Hardware	**Servidor** 100€
Software	0€
Total	**1.860€**

Marzo:

Recursos humanos	**Sueldo** 960€
Oficinas	**Talentier** 800€
Hardware	**Servidor** 100€
Software	0€
Total	**1.860€**

Abril: En el mes de abril se hizo el cambio de servidor. Pero durante este mes se pagó la licencia de ambos meses por si se tuviese que volver al servicio anterior. Además se empezó a ir cada dos semanas a Talent Clue, y se alquilaron dos mesas, una para el director y otra para el estudiante.

Recursos humanos	**Sueldo** 960€
Oficinas	**Talentier** 800€ + **Talent Clue** 150€
Hardware	**Servidores** 140€
Software	0€
Total	**2.050€**

Mayo: Cómo el servidor dió muy buenos resultados de rendimiento y nos ofrecía nuevas funcionalidades muy útiles se decidió darse de baja del otro servicio de hosting.

Recursos humanos	**Sueldo** 960€
Oficinas	**Talentier** 800€ + **Talent Clue** 150€
Hardware	**Servidores** 40€
Software	0€
Total	**1.950€**

Junio:

Recursos humanos	**Sueldo** 960€
Oficinas	**Talentier** 800€ + **Talent Clue** 150€
Hardware	**Servidores** 40€
Software	0€
Total	**1.950€**

Al final se pudieron conseguir las dos premisas en las que se basó el presupuesto del proyecto, usar el máximo de software gratuito y no superar los 10.000€. El coste total del proyecto desglosado mes a mes es el siguiente:

Total:

Febrero	**1.860€**
Marzo	**1.860€**
Abril	**2.050€**
Mayo	**1.950€**
Junio	**1.950€**
Total	**9.670€**

7.5 Viabilidad económica

Este proyecto surgió de la necesidad económica de la empresa Talentier, debido a que una parte de ingresos procede de usuarios que externaliza sus procesos de selección con Talent Clue, permitiéndole en un futuro incorporarse en más ATS aumentando así el número de usuarios, lo que significaba mayor número de ingresos.

Si Talentier consigue integrarse en alguna nueva ATS se podrá amortizar el precio pagado para el desarrollo de este proyecto en un breve periodo de tiempo, ya que una vez realizado el proyecto no significará más coste económico.

Por otro lado el coste de este proyecto no es muy elevado, teniendo en cuenta que la mayor parte del presupuesto se pagaría de todos modos si no se realizará este proyecto, por lo que se puede considerar que es viable económicamente.

8.1 Matriz de sostenibilidad

La siguiente matriz muestra el impacto del proyecto sobre la sostenibilidad.

	PPP[35]	Vida Útil	Riesgos
Ambiental	**Consumo del diseño**	**Huella Ecológica**	**Riesgos Ambientales**
	3	**18**	**-3**
Económico	**Factura**	**Plan de Viabilidad**	**Riesgos Económicos**
	0	**15**	**-3**
Social	**Impacto Personal**	**Impacto Social**	**Riesgos sociales**
	8	**5**	**0**
Rango Sostenibilidad	**11**	**38**	**-6**
	43		

8.2 Sostenibilidad ambiental

El impacto ambiental en cuanto a coste energético ha sido el consumo eléctrico producido por el equipo de desarrollo, el servidor donde se hostea la web y el producido en las oficinas, teniendo en cuenta que el ordenador no es un ordenador de altas prestaciones y las horas de trabajo eran en horario diurno no ha sido un gasto excesivo. En cuanto al servidor del servicio de hosting no se ha conseguido saber el consumo, pero teniendo en cuenta que está siempre en funcionamiento incluso una vez terminado el proyecto se puede considerar un poco costoso energéticamente.

En cuanto a la fabricación previa de el equipo de desarrollo usado, se sabe que produce residuos que afectan al medio ambiente, al haber sido solo un equipo no se puede considerar un gran impacto ambiental, además de que no se dispone solo para la realización del proyecto, el equipo ya tiene 6 años de antigüedad. A la hora de cambiar el equipo se buscará la mejor manera para que tenga el menor impacto sobre el medio ambiente.

[35] Proyecto Puesto en Producción

Al ser un proyecto software este no produce un coste medio ambiental directo, además de que uno de los puntos clave del proyecto es que es replicable, por lo que el número de usuarios que usen la API no afectará directamente al medio ambiente.

8.3 Sostenibilidad económica

El coste del proyecto no se puede considerar caro, debido a que la gran mayoría de los fondos se hubieran consumido igualmente si no se hubiera realizado este proyecto, ya que el estudiante que empezó a trabajar en Talentier no entró a trabajar para realizar este proyecto, sino que fue propuesto posteriormente por la necesidad de éste de realizar un proyecto final de grado.

Al ser Talentier una empresa pequeña con menos de un año de vida y de momento no contar de grandes ingresos, esta tiene la mentalidad de ahorrar el máximo posible, es por esto que todo el software usado durante el proyecto era o gratuito o open source.

Por otro lado, el API realizado ha sido realizado para unas funcionalidades muy concretas, por lo que esta podría ser ampliada en un futuro, en caso de querer invertir en mejorarla con nuevos desarrolladores podría suponer más coste económico al proyecto.

8.4 Sostenibilidad social

Este proyecto está destinado a poder integrar Talentier en cualquier ATS que lo desee, esto supone ayudar a empresas que publiquen ofertas de trabajo mediante alguna ATS, a encontrar mejores candidatos para los puestos propuestos. Otro punto favorable para los usuarios (empresas) que usen Talentier a través del API es que es de coste gratuito, ya que Talentier cobra un porcentaje a los recruiters no a las empresas. Y a las ATS que lo integren les generaría ingresos, ya que Talent Clue también cobra un porcentaje de los beneficios de Talentier si la oferta de trabajo procedía de Talent Clue.

Por último al autor del proyecto la realización de este le ha supuesto una gran primera experiencia laboral en el desarrollo de software, además de aportarle grandes conocimientos en varias tecnologías diferentes, que le servirán para el futuro.

9. Referencias

[1] Documentación de Drupal - Página oficial de Drupal [Online]
<https://www.drupal.org/documentation>
[Consultado día 22 de Enero 2016]

[2] Drupal - Libro de desarrollo de Drupal [Libro]
Pro Drupal 7 Development - Todd Tomlison & John VanDyk
[Consultado día 25 de Enero 2016]

[3] SCRUM - Libro sobre la metodología ágil SCRUM [Libro]
Agile Project Managment with SCRUM - Ken Schwaber
[Consultado día 16 de Enero 2016]

[4] Gantt - Aplicación para hacer diagramas de Gantt [Online]
<http://www.tomsplanner.es/>
[Consultado día 6 de Marzo 2016]
[Consultado día 14 de Junio 2016]

[5] RESTful - Documentación RESTful [Online]
<https://github.com/RESTful-Drupal/restful/wiki>
[Primera consulta día 6 de Abril]

[6] Kanban - Personal Kanban in a Nutshell [Libro]
Personal Kanban in a Nutshell - Jurgen De Smet
[Consultado día 4 de Mayo 2016]

[7] W3Schools - Documentación HTML/CSS/JavaScript [Online]
<http://www.w3schools.com/>
[Consultado desde la fase de desarrollo]

10.1 Anexo A: Documentación del API de Talentier

Default endpoints:

- Get the access-token:

Request URI: https://talentier.com/api/login-token
Request Method: GET
Content-Type: application/json
Authorization: *Basic*

Job endpoints:

- Create new job:

Request URI: https://talentier.com/api/v1.0/Job/new
Request Method: POST
Content-Type: application/json
acces-token: *Token*
Input Data: {“title”:”job_title”, “description”:”job_description”, “location”:”job_location”, “industry”:”job_industry”, “sector”:”job_sector”, “salary_min”: “job_salary_min”, “salary_max”:”job_salary_max”, “vacancies”:”job_vacancies”}

- Update a job:

Request URI: https://talentier.com/api/v1.0/Job/update
Request Method: PATCH
Content-Type: application/json
acces-token: *Token*
Input Data: {“id”:”job_id”, “title”:”job_title”, “description”:”job_description”, “location”:”job_location”, “industry”:”job_industry”, “sector”:”job_sector”, “salary_min”: “job_salary_min”, “salary_max”:”job_salary_max”, “vacancies”:”job_vacancies”}

- Get unread messages of a job:

Request URI: https://talentier.com/api/v1.0/Job/unread_messages
Request Method: GET
Content-Type: application/json
acces-token: *Token*

Input Data: {“id”:”job_id”}

CV endpoints:

- Update CV status:

Request URI: https://talentier.com/api/v1.0/CV/update
Request Method: PATCH
Content-Type: application/json
acces-token: *Token*
Input Data: {“id”:”cv_id”, “status”:”cv_status”}

CP endpoints:

- Create new CP:

Request URI: https://talentier.com/api/v1.0/CP/new
Request Method: POST
Content-Type: application/json
acces-token: *Token*
Input Data: {“jid”:”job_id”, “cvid”:”cv_id”, “salary_annual”:”cp_salary_annual”, “salary_var”:”cp_salary_var”}

- Update a CP:

Request URI: https://talentier.com/api/v1.0/CP/update
Request Method: POST
Content-Type: application/json
acces-token: *Token*
Input Data: {“id”:”cv_id”, “salary_annual”:”cp_salary_annual”, “salary_var”:”cp_salary_var”}

API codes:

Code	Text	Description
200	OK	Success!
400	Bad Request	The request was invalid or cannot be otherwise served
401	Unauthorized	Authentication credentials were missing or incorrect
403	Forbidden	The request is understood, but it has been refused or access is not allowed
404	Not Found	The URI requested is invalid or the resource requested, such as a user, does not exists
500	Internal Server Error	Something is broken

10.2 Anexo B: Normas de la metodología Scrum + Kanban

Daily Standup

At 9h30, keep it SHORT (15min), FOCUS on creation of VALUE + REMOVING BLOCKAGES
Coordinator: SCRUM MASTER

1. UPDATE metrics and graphs
    a. By scrum master, before meeting
2. OBSERVE and ANALYZE metrics and graphs
    a. Why certain changes in velocity, queue volume, etc?
    b. What are the current blockers? Where is the flow obstructed?
    c. What needs to be done to unblock flow?
3. ORGANIZE the day
    a. What are the current priorities?
    b. Who needs help with certain issues?
    c. Who will help out with them, and when?
    d. Who depends on who for the development of certain tickets?
4. COMMIT
    a. Express personal commitments, so as a team we know we can count on you

Release Plan

1. Prepare RELEASE - every morning (between 8h and 9h15)
    a. If there is a dependency with TC, remove the 'Closes' tag in ticket description, so that ticket will remain open until the TC depending part has been deployed
    b. Merge the PRs into development which are in Release column
    c. If there is a MERGE CONFLICT
        i. Move PR to Release column and mark **Merge Conflict**
    d. Create snapshot of master branch in Platform.sh
2. Do RELEASE - every morning (between 8h and 9h15)
    a. git fetch github
    b. pull github development
    c. push platform master
3. Finish RELEASE - every morning (between 8h and 9h15)
    a. Clean up Release column (if any ticket left over and not closed automatically)
    b. Add released total number of Story Points to DEV Metrics
    c. Pull code & database to TEST and to DEV
4. Tag WEEKLY Code Change - every monday morning (just after release)
    a. Create a PR in github (development into master)
    b. Merge PR
    c. Create tag with number v1.x.[weeknumber]

d. Add Description: overview of features

5 Simple Flow Rules

1. Always **FINISH** the task or micro-task you are currently **FOCUSSED ON**

2. If you are STUCK, **LABEL** your ticket
   a. **ORK**: estimation will not be met, important unexpected problems have arisen
   b. **DRAGON**: completely lost in problems, you don't know what to do next anymore
   c. Communicate your status: right now through Slack, and during next daily

3. If you are **READY** with what you were doing, **DO** next item using the following **RULES**
   a. Take a **BUG** ticket if there is one
   b. **TEST** tickets if there are any (not your own tickets), add labels
   c. FIX / IMPROVE QUALITY of your code if a ticket of yours is **not working**, **needs changes**, or **needs more quality**
   d. Give SUPPORT on any **ORK** or **DRAGON** ticket
   e. PULL FIRST ticket of the BACKLOG

4. Except, if there is a **HIGH PRIORITY** ticket on your name
   a. Immediately **STOP** what you are currently doing
   b. Coordinate with the team, who takes responsibility to solve the ISSUE
   c. Communicate any problems you are facing, do not post-pone asking for help

5. **RELEASE** will be done, if there is code to be released, **DAILY** between 8h and 9h15

Daily Schedule

- 8h - 9h15 Release
- 9h30 - 9h45 Daily Standup
- 9h45 - 10h Tech Support
- 10h - 12h30 **Development**
- 12h30 - 13h30 Review MS (if anything to review), and Tech Support (on demand)

LUNCH

- 14h30 -> afternoon Fix code, and finish your stuff
  (*Taking always into account, the agenda of other Devs, so if you want your ticket to be retested, finish it asap.)*

Definition of Ready

1. DEFINED
    a. New features
        i. Why? Why important
        ii. Who? Who is the user, who will this deliver value to
        iii. What? Expected outcome
            1. Short description
            2. Mock-up or sketch
            3. Design (if new feature and special design needed)
    b. Bugs (what to do to reproduce?)
        i. URL + Screenshot
        ii. Expected result

2. ANALYZED
    a. Split in very small chunks of independent working functionalities
    b. Implementation Plan defined
        i. Cleared out all technical doubts and questions

3. ESTIMATED
    a. Estimations agreed on with whole team (estimation = READY)
    b. No more than 5 story points per ticket

4. CLEAR
    a. Developers understand expected outcome

5. LABEL a ticket if NOT READY
    a. If a ticket is not ready, because something is missing, assign the appropriate label (blue labels are good candidates)

Definition of Done

1. WORKING as DEFINED
    a. Tested and approved (by -internal- client if relevant, during development process)
    b. Tested and approved (by another developer)
2. QUALITY ASSURED
    a. Coding standards respected
    b. Readable and extendable
    c. No manual database changes needed (or very very exceptionally, using label)
3. MERGED
    a. Into github development branch
    b. No dependencies on other tickets
4. RELEASED
    a. Translations, and other manual changes, executed
    b. Available in LIVE
    c. Available in TEST & DEV

LABELS

1. AREA of DEVELOPMENT
    a. **THEMING**
    b. **jQuery / AJAX**
    c. **TC** (TalentClue)
    d. **API**

2. NOT READY status
    a. **needs design**
        i. A visual design is needed for this ticket
    b. **needs discussion**
        i. A discussion is needed between stakeholders, to take decisions
    c. **needs refinement**
        i. More details are needed, by stakeholders

3. FLOW status
    a. **waiting**
        i. Depending on external input or action
    b. **needs more quality**
        i. Code Review not passed, code needs quality improvements
    c. **needs changes**
        i. Code Review not passed, code needs structural changes
    d. **not working**
        i. Testing not passed, not working as defined

4. SPECIAL ATTENTION FLOW status

    a. **BUG**
    b. **HIGH Priority**
    c. *** ORK ***
    d. *** DRAGON ***

5. DEVELOPERS labels
    a. **IG**
    b. **VK**
    c. **MS**

6. CLOSING reason
    a. **In progress**
    b. **INVALID**
    c. **DUPLICATE**
    d. **CAN'T REPLICATE**

SCRUM Meetings

1. **Daily StandUp**
    a. When? EVERY DAY at 9h30
    b. 15min max
    c. Objective:
        i. Focus on Value and removal of blockages
        ii. Organize and coordinate the day
    d. Coordinator: Scrum Master

2. **Refinement Meeting** (grooming)
    a. When ? Every TUESDAY, fixed time, 1h
    b. Backlog items that have no estimations
    c. Objective:
        i. Split work in small parts
        ii. Discuss technical issues
        iii. Create implementation plan
        iv. Estimate
    d. Coordinator: Team Lead

3. **Retrospective Meeting**
    a. When ? Every two weeks (FRIDAY), 1.5h
    b. Objective:
        i. Learn from past 2 weeks
        ii. Share and propose improvements
        iii. Concrete proposals for change
    c. Coordinator: Scrum Master