

# **Proyecto** **Truequéalo**

## **Grupo** **The Red Chicken**

### **"PRUEBAS Y VALIDACIÓN"**

Hito: 4

Fecha entrega: 26-05-2017

Versión: 1

#### **Componentes:**

- Juan Francisco Bustos Correas
- Pablo Serna Martínez
- Yolanda Torregrosa Hernández
- Alejandro Torres Mateu
- Raquel Yuste Torregrosa

## **Contenido**

[Introducción](#)

[Pruebas](#)

[Estrategia y herramientas utilizadas](#)

[Flujo de trabajo](#)

[Validación](#)

## 1. Introducción

En este documento se describen los diferentes mecanismos para probar y validar el sistema realizado, tanto para la parte del front-end como en la parte del servidor.

## 2. Pruebas

### 2.1. Estrategia y herramientas utilizadas

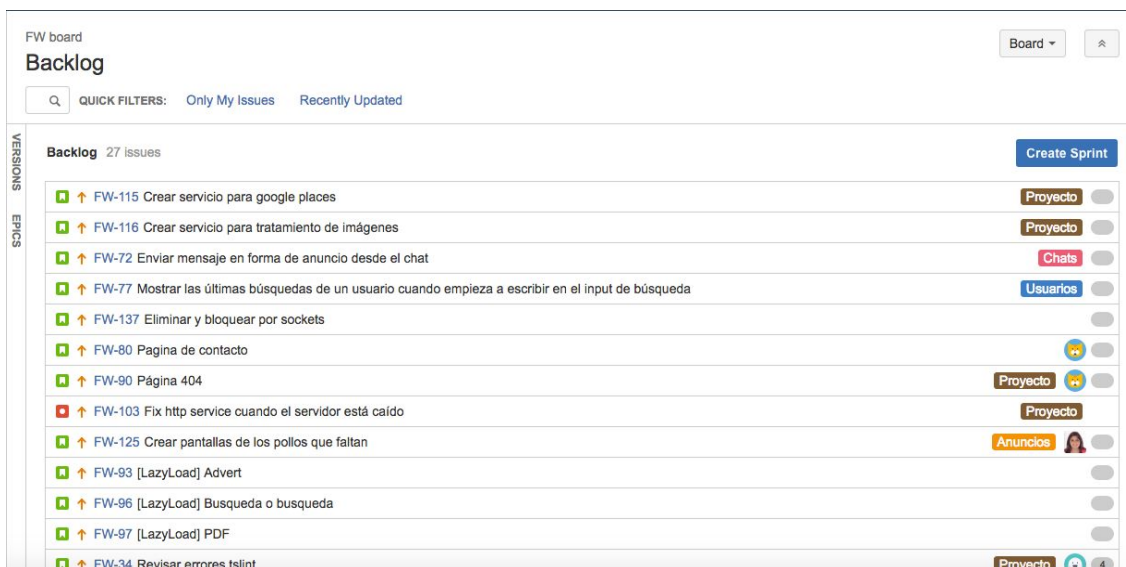
El principal objetivo de las pruebas que hemos realizado durante todo el proceso de desarrollo la aplicación es que todo el equipo de desarrollo sepa qué se está haciendo, y sobre todo, cómo se está haciendo. De esta forma todos revisamos no solo la funcionalidad, sino el propio código de cada una de las tareas que se han ido llevando a cabo.

Para llevar a cabo esta estrategia se han elegido las siguientes herramientas:

- **Wercker** para la integración continua del código a la plataforma, de forma que partimos de un prototipo al que se le van sumando funcionalidades. Cada tarea se integra por separado en la aplicación en *producción* (rama develop) una vez que ha sido validada por el equipo.
- **JIRA** para la gestión de tareas y de la metodología **SCRUM**.
- **Bitbucket** como repositorio para almacenar el código de la aplicación y relacionarlo con cada una de las tareas definidas en Jira.
- **SourceTree** como cliente de git que ofrece una interfaz gráfica para facilitar la gestión del repositorio Bitbucket. Permite hacer pull, commit y push de forma más intuitiva, así como cambiar de rama o clonarla de remoto a local.

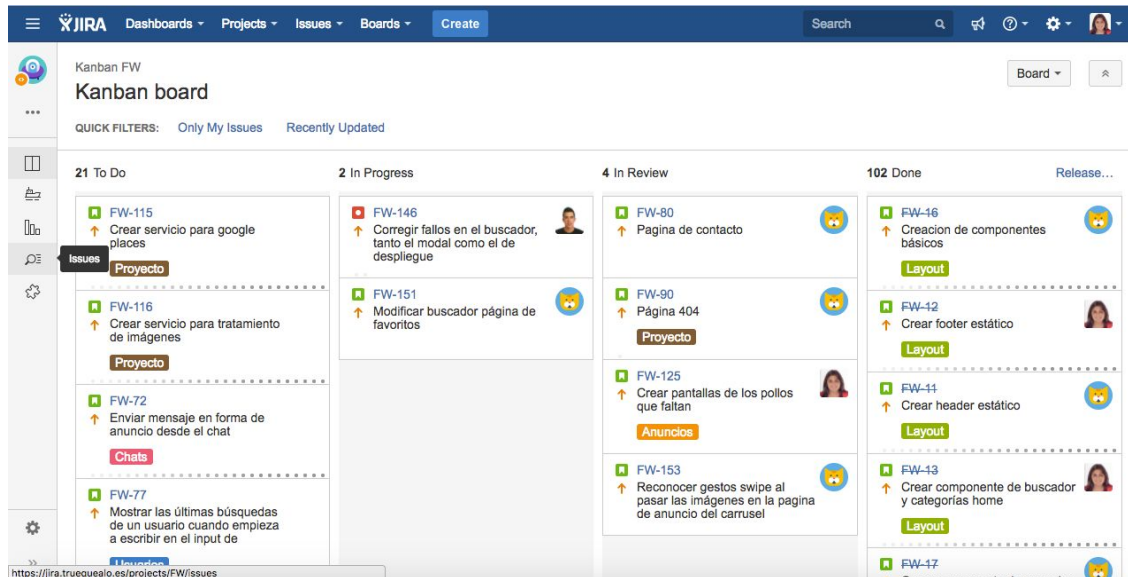
### 2.2. Flujo de trabajo

Partiendo como base de la metodología SCRUM, se ha creado una pila de backlog para el front y otra para el back-end. En esta pila se van creando todas las tareas, indicando el tipo (tarea o bug) y etiquetándolas con el topic correspondiente (Usuario, Anuncio, Chat, Documentación, etc.). De esta forma, el backlog contiene la lista de tareas pendientes por hacer y se va actualizando constantemente, cuando surgen nuevas tareas o bugs que hay que corregir.

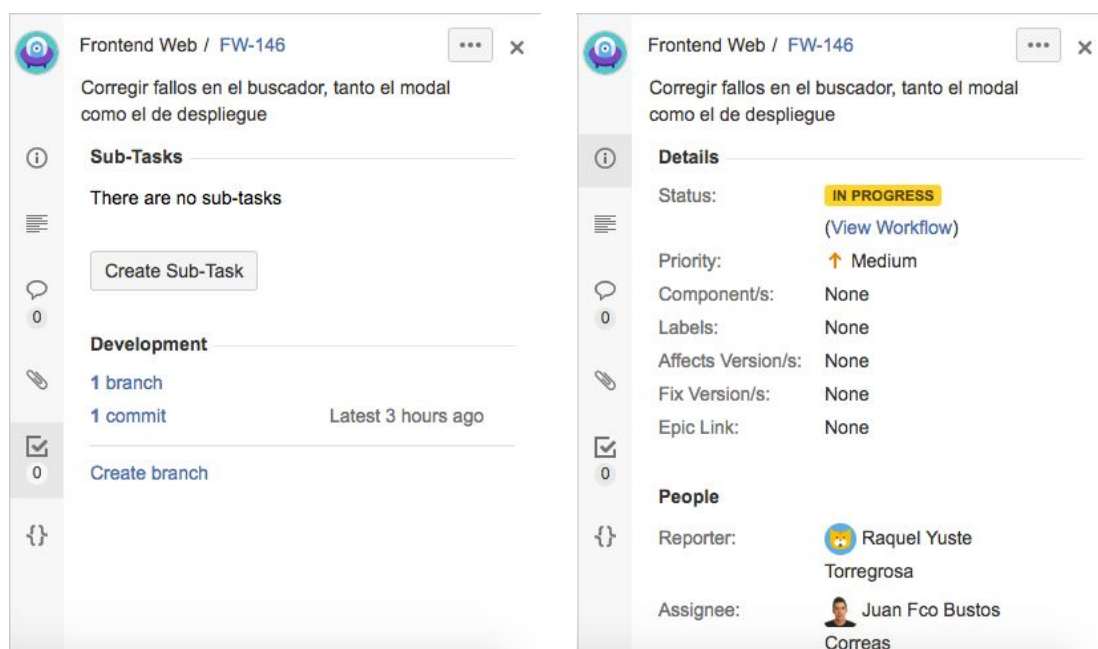


En cada *planning meeting*, se crea el sprint correspondiente y se introducen las tareas del backlog que ha acordado realizar en dicha iteración. Es entonces cuando se asigna cada tarea a uno de los miembros del equipo y se realiza conjuntamente una estimación de la misma.

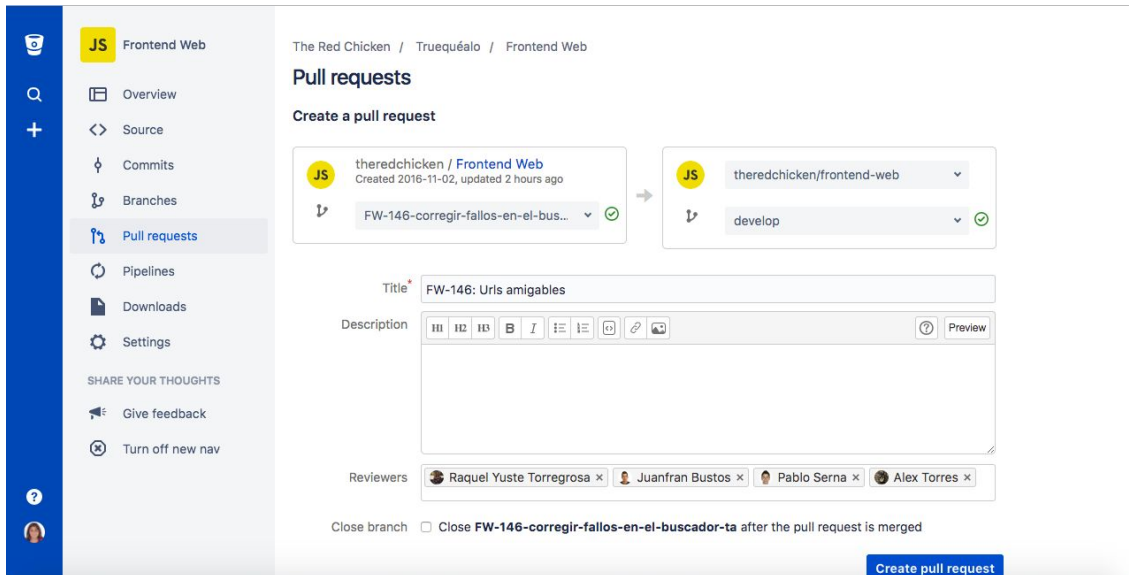
Una vez tenemos el sprint definido, se nos presenta en forma de kanban con 4 columnas: *To Do*, *In Progress*, *In Review* y *Done*.



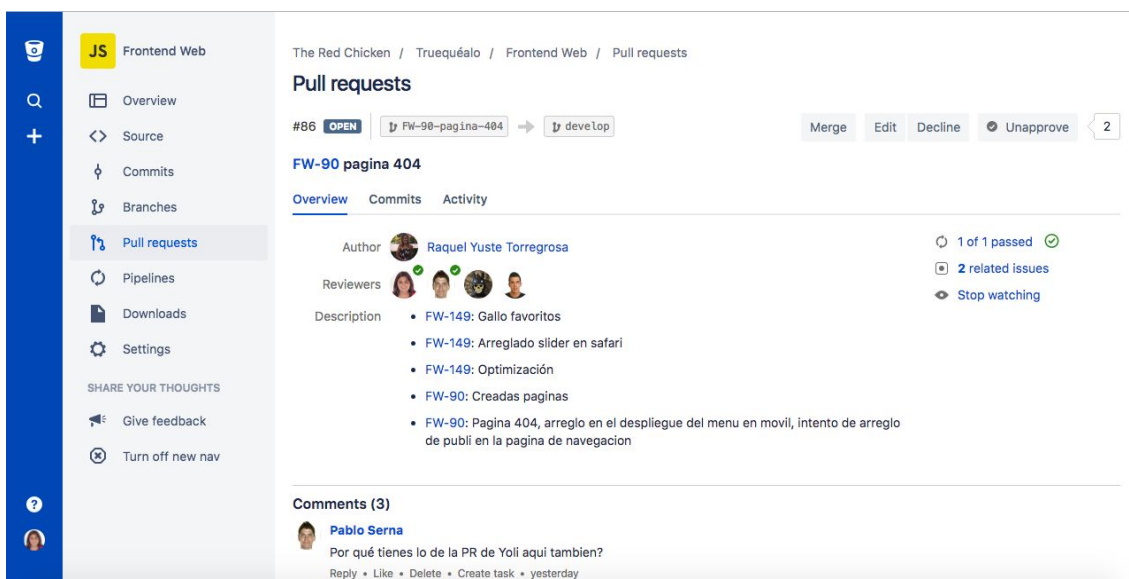
Durante el sprint, las tareas irán pasando por las diferentes columnas. Cuando un miembro del equipo comienza a hacer una tarea la mueve a *In Progress* y, desde el propio Jira, crea una rama del repositorio de Bitbucket para dicha tarea.



Una vez el usuario ha finalizado la tarea y ha hecho el commit de todo su trabajo, desde la propia tarea en Jira crea una *pull request* (en adelante, PR) para dicha rama y mueve la tarea a *In Review*. Esta PR es una petición para unir el código de su rama a la rama develop, donde se encuentra el código en producción.



Cuando un usuario crea una PR, el resto del equipo es notificado vía email con un enlace a la página donde aparecen un extracto de lo que ha realizado en la tarea: los commits realizados, los ficheros modificados, añadidos o eliminados y las líneas concretas que se han tocado. Estas modificaciones han de ser revisadas por el resto de miembros del equipo, pudiendo dejar comentarios relativos a archivos o líneas concretos si tienen alguna duda o consideran que algo no es correcto.



El autor puede contestar los comentarios y hacer modificaciones en su rama, actualizando la PR. Por último, cuando la PR ha conseguido un número mínimo de aprobados (2 en el caso del repositorio de la API y 3 en el front-end), puede unir (*merge*) su rama a la rama develop.

The screenshot shows a GitHub pull request for the file `src/app/components/advert-card/advert-card.component.ts`. The interface includes tabs for 'Side-by-side diff', 'View file', 'Comment', and a menu icon. Three comments are visible:

- Pablo Serna**: "Todo esto de favoritos lo metería en un servicio porque se utiliza en varios sitios pero bueno que no es necesario ahora".
- Raquel Yuste Torregrosa** (AUTHOR): "Puede ser, aunque en realidad ya est'a como englobado en el servicio de advert, lo unico que se repiten son las llamadas al servicio ese y el a;adir y quitar a la variable de numero de favoritos".
- Yolanda Torregrosa**: "yo lo veo bien así, porque ya está en advert".

Below the comments is a code diff showing changes to the `advert-card.component.ts` file. The diff highlights the removal of the `Router` import and the addition of `ActivatedRoute`, `Params`, and `Router` imports from `@angular/router`. Other imports from `@angular/material` and `moment` are also shown.

Aquí es cuando Wercker se encarga de subir las nuevas actualizaciones de develop a producción (a `truequealo.es` o `api.truequealo.es`) y compilar de nuevo el código.

The screenshot shows the Wercker CI/CD pipeline interface. The top navigation bar includes 'ORACLE + wercker', 'Registry', 'Applications (w)', and a '+ Create' button. The main area displays a list of pipeline runs for the application `FW-80-pagina-de-contacto`. Each run shows the user, the action performed, the status, and the time taken.


ID	Action	Status	Time
#3001ceb	contact.component.html edited online with Bitbucket	build	5 hours ago
#da8c179	Merged in FW-148-cambios-en-la-pagina-de-busqueda (pull request #85)	build, deploy	6 hours ago
#193013e	contact.component.html edited online with Bitbucket	build	6 hours ago
#99c3591	contact.component.scss edited online with Bitbucket	build	6 hours ago
#ee63a84	not-found.component.scss edited online with Bitbucket	build	6 hours ago

Además, Wercker hace sus propias pruebas en cada PR compilando el código de la rama, lo nos permite ver en la página de la misma si la *build* de esa rama es correcta o hay algún error.

	Pablo Serna	<a href="#">10bd323</a>	FW-90 fix ng build	23 hours ago	
	Pablo Serna	<a href="#">f1391fc</a>	FW-90: publicidad arreglada	yesterday	
	Raquel Yuste T...	<a href="#">4ac19c6</a>	not-found.component.scss edited online with Bitbucket	yesterday	
	Raquel Yuste T...	<a href="#">1649ae4</a>	favorite.component.scss edited online with Bitbucket	yesterday	
	Raquel Yuste T...	<a href="#">df5e5bb</a>	favorite.component.html edited online with Bitbucket	yesterday	

Una vez se ha mergeado la PR, se mueve la tarea a la columna *Done* y se indica el número de horas reales empleadas. Se repite el mismo proceso con cada tarea hasta que ya no queden tareas por hacer, en cuyo caso, habrá finalizado el sprint.


Close Issue



Closing an issue indicates that there is no more work to be done on it, and that it has been verified as complete.

Resolution\*  ?

Fix Version/s **None**

Assignee  Yolanda Torregrosa Hernández  
[Assign to me](#)

Time Spent  (eg. 3w 4d 12h) ?

Date Started

Remaining Estimate ☒ Adjust automatically  
☐ Leave estimate unset  
☐ Set to  (eg. 3w 4d 12h)  
☐ Reduce by  (eg. 3w 4d 12h)

Close Issue

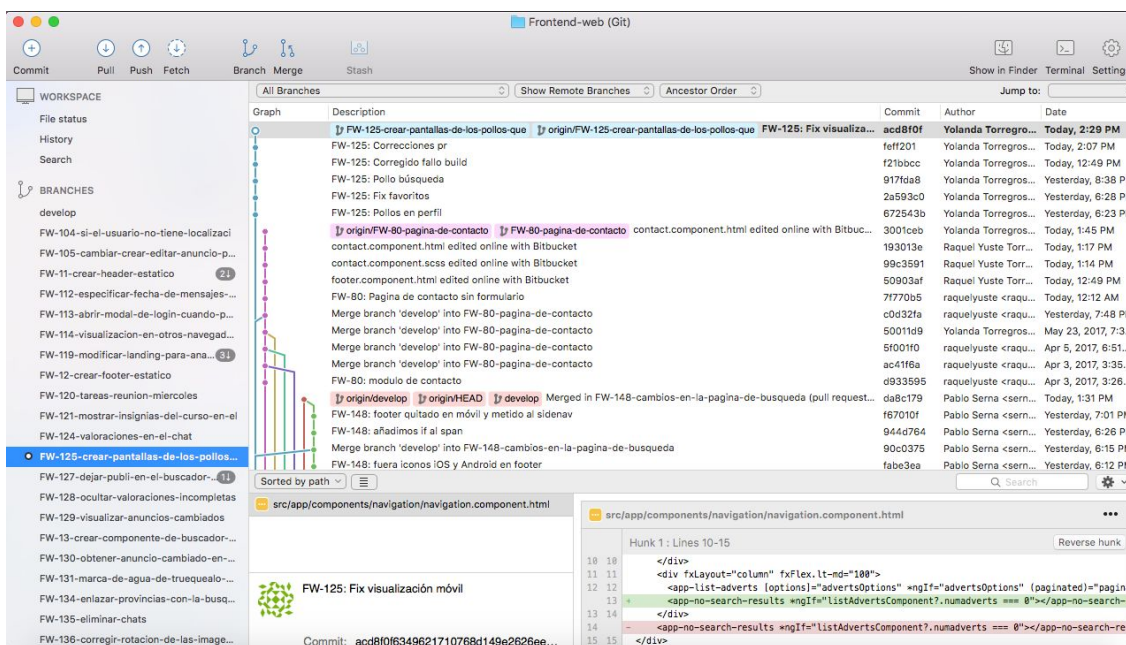
Cancel



### 3. Validación

Este flujo de trabajo que permite probar cada una de las funcionalidades del proyecto también permite la validación del mismo, pues las revisiones y pruebas por parte del equipo no solo son funcionales y comprueban que no hay fallos de compilación, sino que sirven también para revisar que las funcionalidades implementadas son acordes con las diseñadas y que todo el equipo tiene en todo momento conocimiento del estado del proyecto y de la respuesta de las interfaces.

Para cada *pull request* no sólo se hacen revisiones de fallos y optimización, sino que cada usuario se descarga la rama en local, la compila y comprueba que lo que se ha implementado coincide y cumple con los requisitos establecidos en el documento de especificación, tanto a nivel de funcionalidad como a nivel de diseño de interfaz y experiencia de usuario.



### 4. Conclusiones

Tenemos un total de **116 ramas en el repositorio de la API** y **153 en el de front-end**, lo que equivale a unas **270 pull requests** revisadas, probadas y *mergeadas*.

Sin duda, el flujo de trabajo que hemos llevado nos ha ayudado mucho a reducir el número de bugs, tener un código más limpio y optimizado, y sobre todo a saber qué está haciendo cada uno en cada momento y cómo lo está haciendo. De esta forma, todos sabemos cómo está implementada cada funcionalidad y para qué sirve cada uno de los archivos del proyecto, por lo que resulta muy fácil rotar o intercambiar tareas y aconsejarnos o ayudarnos entre nosotros, lo que ha mejorado mucho la calidad y profesionalidad del proyecto y ha agilizado su desarrollo.