Proyecto UNLa

Alan Pérez, Micaela Peralta, Nicolás Ojeda, UNLa, 29 de septiembre 3901, Buenos Aires, Argentina.

En este proyecto vamos a tratar el armado de un juego en el lenguaje de programación Phyton, desde la planificación, el diseño y análisis, programación, documentación y testeo de software.

En este breve resumen, trataremos acerca de la historia del juego:

A millones de años luz de la tierra, se encuentra nuestro personaje Alien, quien tuvo dificultades con su nave espacial en un vuelo de emergencia y aterrizó en un planeta desconocido. Su misión en el primer nivel es armarse de valor y esquivar diversos objetos que caen desde las alturas, para poder recoger las piezas perdidas de su nave y poder emprender el viaje de vuelta a su hogar. Cuando lo consigue, ya nuevamente en su nave, debe evitar los meteoritos espaciales que complican su vuelta y volar hacia su planeta.

-Programación, Phyton, juego, desarrollo, producción, software –

Introducción

El proyecto se encuentra dividido y pensado en 4 partes:

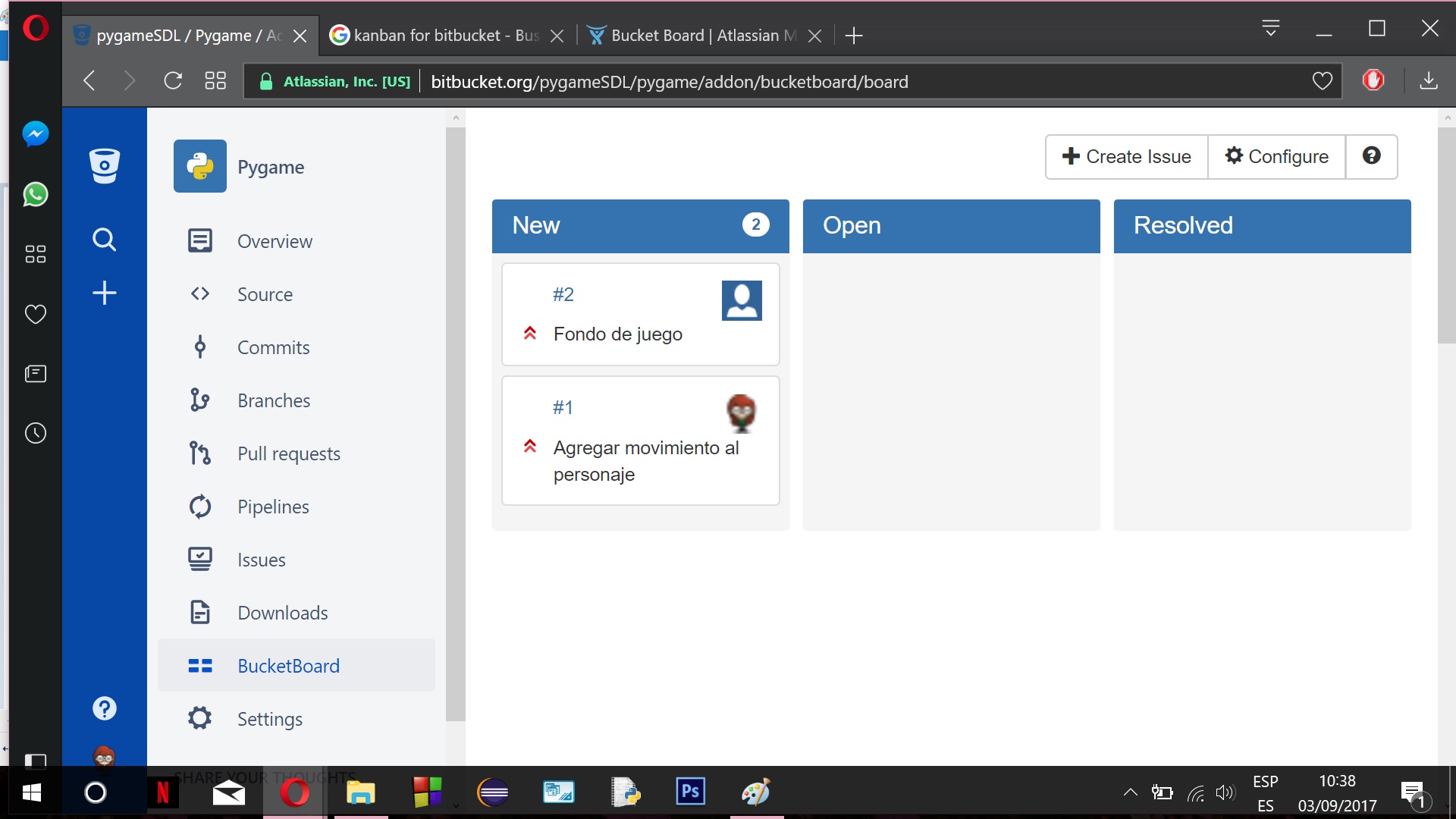
* Documentación del producto
* Documentación técnica
* Código
* RRHH

En cada una de ellas se encuentra ordenado, para una mejor organización, las partes del juego. Pasando desde la guía de usuario, imágenes, sonidos, música y el código completo del juego.

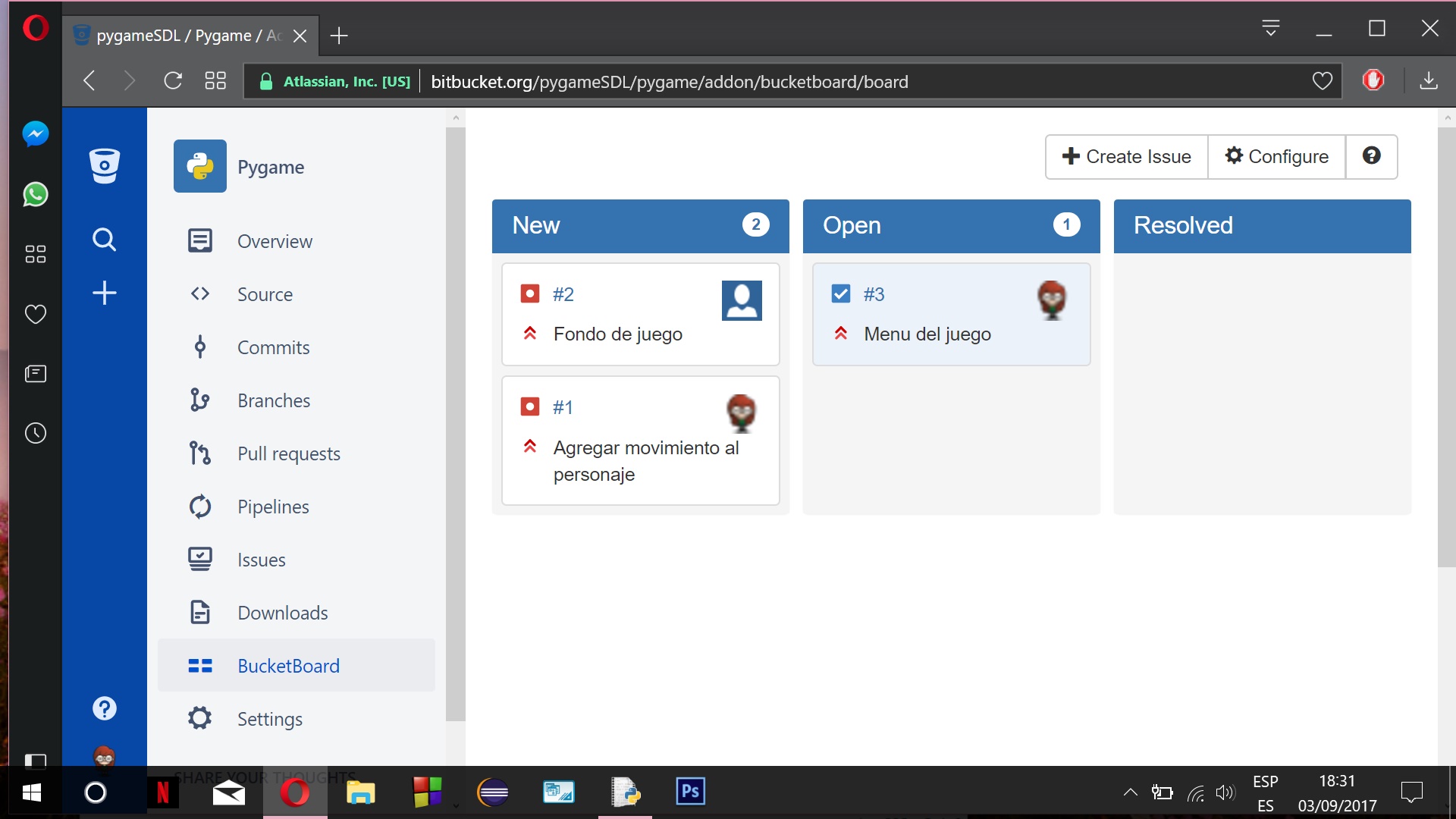
En cuanto a la programación, utilizamos un control de versiones para poder ahorrar tiempo, trabajar en equipo simultáneamente y, en caso de errores, poder volver a una versión estable del código. Para todo esto usamos el software controlador de versiones Git y el cliente SourceTree.

Para tener un control y poder planificar correctamente el juego, diariamente realizamos una serie de minutas en las cuales comentábamos en grupo ideas y aportes generales. A medida que empezamos la programación y sólo por control, realizamos reuniones de avances, en las cuales anotamos las cosas que modificamos del código, que día y la hora correspondiente.

Como ayuda para nosotros, utilizamos una de las metodologías ágiles de organización de tareas: Kanban y a continuación dejaremos algunas capturas de pantalla en el medio de la producción del juego.

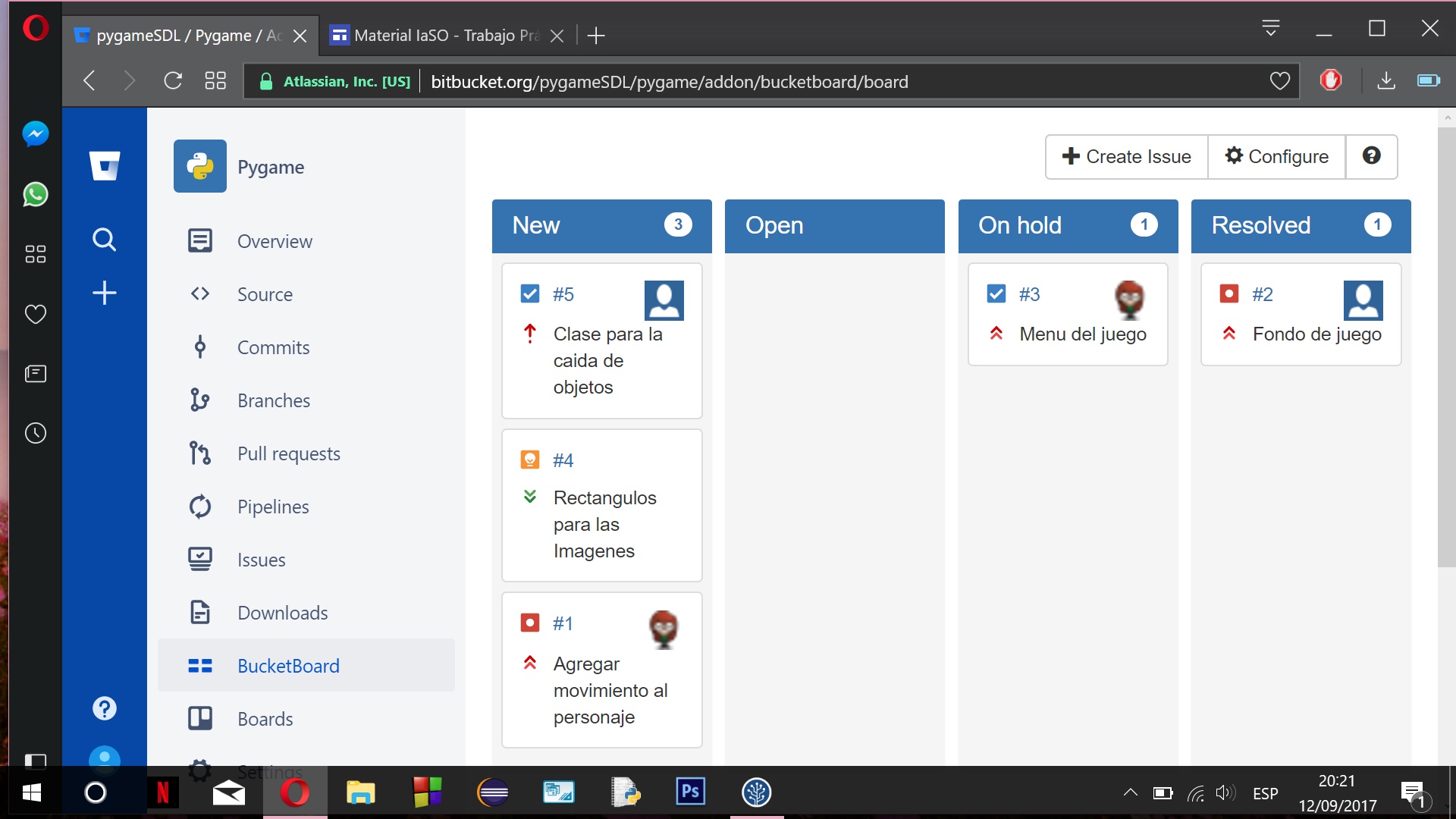


1)



2)

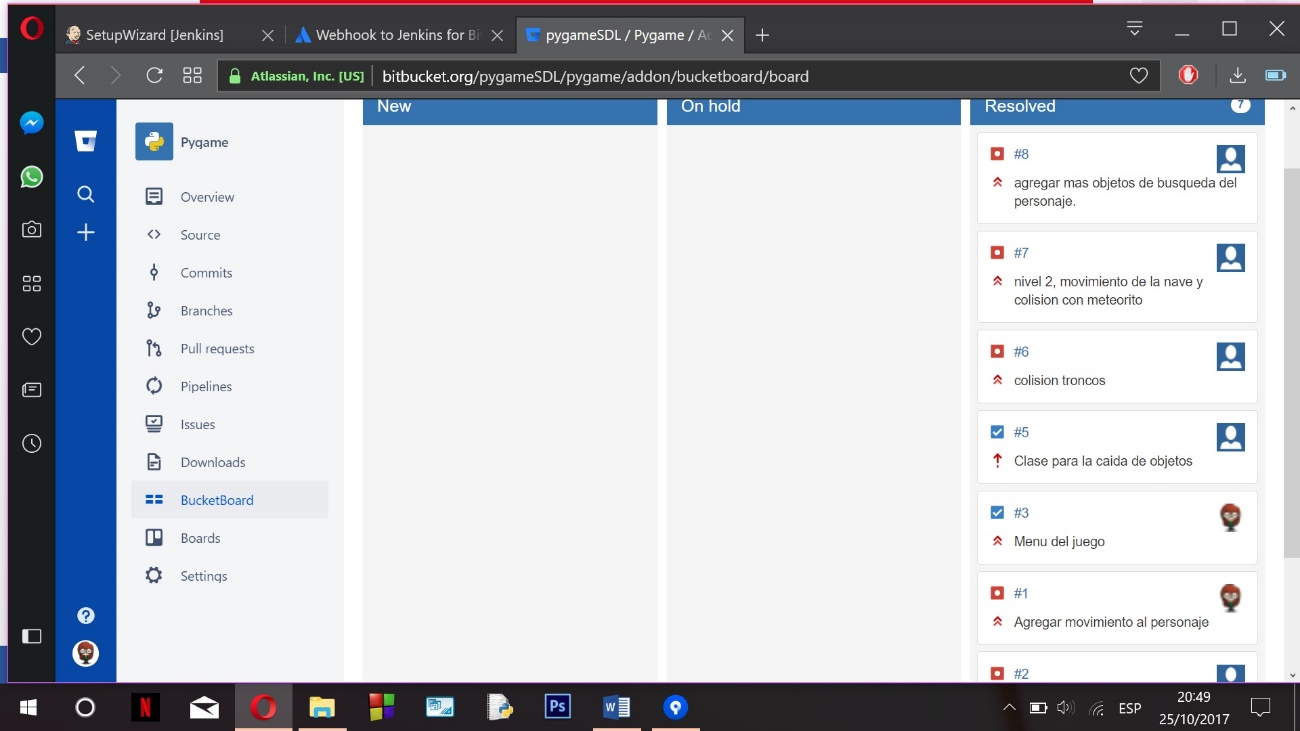
3)

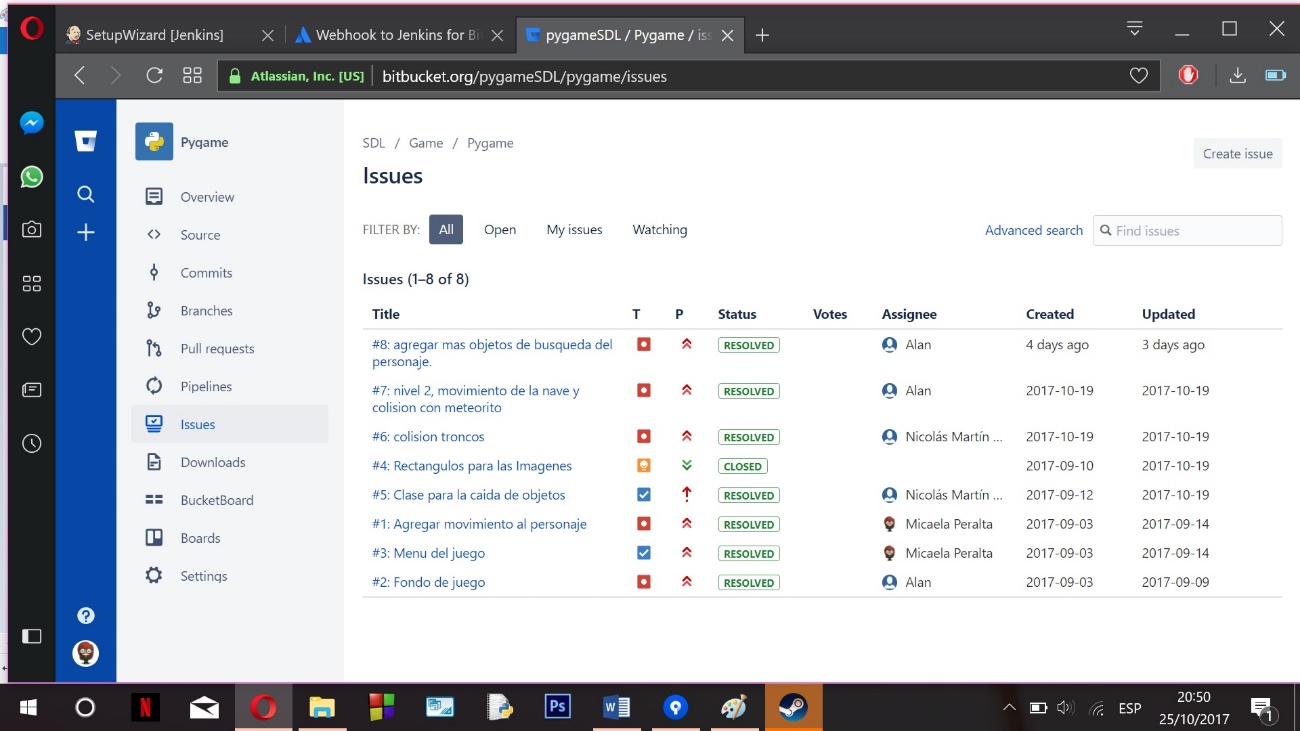


En estas imágenes podemos observar cómo se dividen y asignan las diversas tareas para la producción del juego:

En la imagen 1) (3/9) se ponen en el tablero las tareas o problemas a resolver.

La imagen 2) muestra que la tarea ya fue abierta y está en proceso de ser resuelta, como se muestra en la imagen número 3), que la tarea pasa de “Open” a “On hold” y luego, una vez finalizada, pasa a la columna “Resolved”.

4)

5)

En las imágenes 4) y 5) observamos como todas las tareas que fueron asignadas se llevaron a cabo hasta su resolución. La imagen 5) muestra desde otro punto las tareas, con un sistema propio del control de versiones “Bitbucket”.

Estado de la cuestión

Gracias al uso de las diferentes metodologías ágiles, es fácil (casi siempre) notar los diversos “bugs” o problemas que pueda presentar el programa en cuestión y para mayor rapidez de resolución de conflictos, asignamos la tarea a uno de los integrantes para que proceda. Si por algún motivo le surgiera un problema, otro integrante deberá solucionar el conflicto lo antes posible.

Para la detección de problemas, existen herramientas de integración continua que permiten detectar fallos lo más rápido posible mediante pruebas métricas y de calidad, compilando el código desde el repositorio central del control de versiones.

Investigación a futuro

Seguiremos desarrollando y mejorando el trabajo con las diferentes herramientas que están al alcance de cualquier programador. Las metodologías agiles e integración continua son en las que reforzaremos y trabajaremos para mejorar su uso.

Conclusiones

Luego de haber finalizado la producción del juego, podemos concluir que una parte muy importante del ambiente de desarrollo fueron las metodologías ágiles y el controlado de versiones. Sin el último hubiese llevado mucho más tiempo y trabajo darnos cuenta de errores y, a la vez, solucionarlos sin poder volver a una versión estable sería laborioso. Sabemos que trabajar en equipo no siempre resulta cómodo para todos, pero este tipo de sistemas hace que el proceso sea más eficiente y logra una mayor equidad a la hora de realizar tareas.

Otro punto es la correcta documentación del programa, la cual ayuda a la hora de analizar y agregar contenido y además da un mejor entendimiento al usuario.