

PROYECTO FINAL

INTEGRACIÓN CONTINUA EN EL DESARROLLO ÁGIL



ÁLVARO RUIZ CALZADA

# INSTALACIÓN

Ha habido un cambio en los requisitos mínimos de los *self-hosted* runners de Github Actions ([aquí](https://docs.github.com/es/actions/hosting-your-own-runners/managing-self-hosted-runners/about-self-hosted-runners)) para usar Ubuntu 20.04 o superior. Dado que el contenedor aportado en la guía de clase tenía una versión anterior, se ha actualizado el SO a una versión más reciente.

De forma análoga, se ha actualizado la versión java de compilación a las 17 y las versiones de los plugin y dependencias definidas en el pom.xml del proyecto.

# GESTIÓN DE LAS RAMAS

Se han creado dos ramas, ambas precedidas por los códigos de los requisitos que cumplimentaban.

Primero se han realizado los requisitos REQ1 y PU y Stage. Tras ello se ha hecho una PR, se ha mergeado y se ha sacado desde master una segunda rama con los requisitos restantes.

# STAGE

A diferencia del entorno productivo, se ha decidido que el despliegue al entorno preproductivo sea por despliegue continuo. También se limita a la rama de master. De tal manera que se convierta en un entorno donde probar y validar los cambios antes de subir a producción.



Dado que el fichero de publish-profile es distinto al de producción, se ha creado otro secret para este entorno.

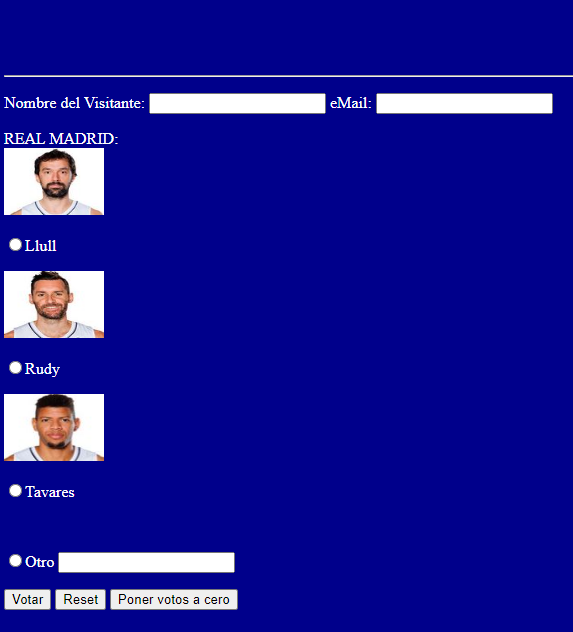
# QA

Como pedían los requisitos de la práctica, se ha limitado a 20 los mayor issues. Se ha trabajado en reducir estas incidencias, centrándonos principalmente en añadir un sistema de loggeo (slf4j), refactorizar el código HTLM proporcionado como base (añadiendo etiquetas como ¡DOCTYPE y dejando de usar etiquetas deprecadas) y aplicando buenas prácticas (evitar el select \* o quitando los casteos innecesarios).

Por otro lado, se ha reducido el rango de cobertura de código a 0 para evitar problemas en el despliegue dado que la cobertura adicional del código no se requería.

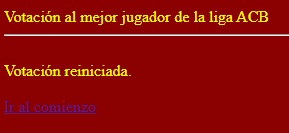
# REQ-1

Se ha creado un botón de reinicio de votos que permite forzar que todos los jugadores de la base de datos tengan su contador de votos a cero.



*Imagen 1 – Botón “Poner votos a cero”.*

Para avisar al usuario que el borrado se ha realizado correctamente se ha creado una vista adicional (VotacionReiniciada.jsp) que confirmaba el borrado. Esta vista además contaba con enlace par volver a la página principal.



*Imagen 2 – Vista de confirmación de reinicio de votación.*

# PU

Se ha definido con Mockito la simulación a la base de datos. Para ello se ha validado que las llamadas que se realizaban eran las correctas.

# REQ-1

Se ha creado un botón de “Ver votos” que muestra una tabla con los atributos (id, nombre y votos) de los jugadores.



*Imagen 3 – Botón “Ver votos”.*



*Imagen 3 – Vista de votos de cada jugador.*

Se ha modelizado una clase Jugador para facilitar la lectura en base de datos.

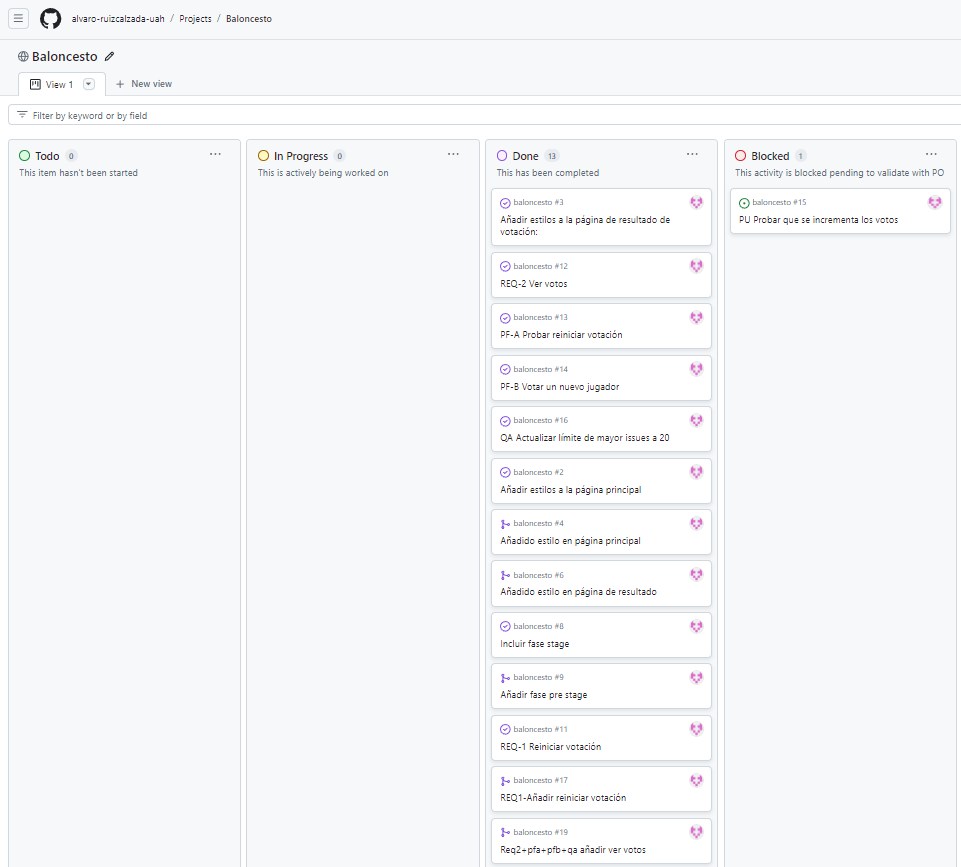
# PF-A & PF-B

Se han desarrollado de forma análoga a la propuesta vista en la base del proyecto.

# PROYECTO

Debido a que no se ha conseguido implementar correctamente el PU, se ha creado una nueva columna “Blocked” a la que corresponde esta tarea (issue).

La foto final es:



*Imagen 4 – Tablero del proyecto.*