Universidad de Sevilla
Evolución y Gestión de la Configuración
Curso 2023/2024
Grupo: 3

Políticas del equipo

INNOSOFT PROGRAMAZE

Tutor: David Romero Organvídez

Miembros del grupo: Casal Ferrero, Rubén
Domínguez Ruiz, Andrés
Fernández Castillo, Javier
Montero Martínez, Francisco Jesús
Otero Barbasán, Manue

Repositorio: https://github.com/motero2k/programaze

ÍNDICE

1. Introducción

En este documento se detallan **todas las políticas** seguidas por el equipo de desarrollo “mace” a la hora de **desarrollar la aplicación web “Programaze”** para la gestión de las propuestas del comité de programa en las jornadas de Innosoft.

2. Alcance

Las políticas aplican desde el **inicio del proyecto hasta su cierre** y conciernen a **todos los miembros** del proyecto (desarrolladores, coordinadores), **el incumplimiento de estas será sancionable** con lo indicado en su apartado correspondiente.

3. Políticas aplicables

3.1. Políticas de control del proyecto

P-CTRL-Sancionadora:

- Se llevará a cabo una **evaluación regular de la implicación** de los miembros del equipo en el proyecto. Esta evaluación se entregará al final del proyecto y se realizará mediante una escala de **0 a 1**, donde 0 representa una baja implicación y 1 una alta implicación.
- Los miembros del equipo tendrán la oportunidad de **discutir sus calificaciones y proporcionar justificaciones** antes de que se apliquen sanciones.
- **Las sanciones pueden incluir:** reasignación de tareas, asignación baja de la puntuación final, o, en casos extremos la expulsión del grupo si así se decide por el resto del equipo.

P-CTRL-Roles:

- Se elige a un **Scrum Master** encargado de coordinar al equipo y hacer de mediador en la resolución de conflictos.

3.2. Políticas de interacción dentro del equipo

P-TEAM-Comunicación:

- **Para las discusiones cotidianas** y actualizaciones rápidas, la comunicación del equipo se llevará a cabo a través de los grupos de **whatsapp o discord** pertinentes.
- **Las reuniones de control de equipo se programan semanalmente** para discutir el progreso del proyecto, los desafíos y la planificación.
- Se fomentará la comunicación abierta y la colaboración, y los miembros del equipo deberán responder a las solicitudes de comunicación en un **plazo razonable (máx 1 semana).**
- En caso de faltas de respeto o falta de implicación se aplicará la política P-CTRL-Sancionadora.

P-TEAM-Reuniones:

Se distinguen las **reuniones semanales** de control de las **técnicas**:

- En las reuniones semanales de control se procura ser lo más **conciso** posible, dando un **estado actual** del proyecto y la necesidad de reuniones técnicas. Se toman decisiones de alto nivel o rápidas de resolver. Para esto es imperante el uso de un **orden del día** preparado previamente.
- Las reuniones técnicas tienen una duración mayor y sirven para trabajar en grupo o discutir ciertas decisiones.

Se fomentará la participación de todos los miembros del equipo de trabajo y al final de la reunión se debe guardar el **diario de sesión.**

P-TEAM-Resolución de conflictos:

- Es **deber** de todos los miembros resolver los conflictos de **forma pacífica**, pudiéndose aplicar medidas sancionadoras en caso contrario.
- Para resolver un conflicto se hablará entre los **interesados** presentando argumentación que respalde su punto de vista, en caso de no llegar a una resolución el **Scrum Master** hará de mediador, quien podrá apoyarse de quien considere para resolver el conflicto.

3.3. Políticas de documentación

P-DOC-Formato:

- Para documentar se utiliza el **formato establecido por el equipo**. Este es el que está en las **plantillas de documentos** dentro de la carpeta compartida de Google Drive.
- La documentación debe estar escrita en **castellano** y debe intentar ser **clara y precisa** en sus explicaciones.

P-DOC-Contenido:

- **Como mínimo** se realizarán los **documentos requeridos** por la asignatura *Evolución y Gestión de la Configuración.*
- Además de esta, se pueden realizar **otros documentos** comunes en el desarrollo de proyectos de Ingeniería del Software como: Documento de Requisitos, Decisiones de diseño, Testing, etc.

3.4. Políticas de desarrollo

P-DEV-Control de versiones:

- Se debe utilizar un sistema de control de versiones, en este caso **Git**, para gestionar el repositorio del proyecto. Se usará **GitHub** para almacenarlo en la nube.
- Todos los cambios en el código deben registrarse en el repositorio con comentarios significativos siguiendo la **política de commits y la política de ramas.**

P-DEV-Commits:

- Los commits se mantendrán **atómicos**, lo que significa que cada commit debe abordar **un solo cambio** o una única funcionalidad.
- Se seguirá el estándar de **"Conventional Commits"** en el proceso de commits, lo que significa que cada commit debe tener un mensaje que seguirá el **formato "tipo: *mensaje*":**
    - tipo = "***feat***" para una nueva característica.

    - tipo ="***fix***" para una corrección.
    - tipo ="***docs***" para documentación, etc.).
    - Los **mensajes de commit** deben ser **claros y descriptivos** para que cualquier miembro del equipo pueda entender el propósito del commit sin necesidad de revisar el código.

P-DEV-Ramas:

- Se emplea una aproximación a **Git Flow**.
- Debe existir una **rama principal "main"** que refleje la versión estable y desplegable del software. Además, una rama **"develop"** que contenga los cambios de desarrollo
- Cada característica, corrección de error o mejora **importante** debe tener su propia rama de desarrollo que **nace de develop y se une a develop al finalizar.**
- **Para hacer merge de una rama se debe haber pasado una revisión por al menos un miembro del equipo.** Este principio está sujeto a excepciones por máxima urgencia o por la minimalidad del cambio (Ejemplo: una errata en un texto).

3.5. Políticas aseguramiento de la calidad

P-QA-Integración Continua:

- Se realizarán pruebas de **integración continua** como se especifica en la asignatura *Evolución y Gestión de la Configuración*. Cada vez que se hace un cambio en una de las ramas ***main*** o ***develop*** deben ejecutarse.

P-QA-Buenas Prácticas:

- Se debe desarrollar usando las ***buenas prácticas en la ingeniería del Software****. Para ello:*
- **Refactorizar habitualmente**, todo código que esté creando deuda técnica debe tenerse en cuenta y decidir ignorar el problema (en casos pequeños) o fijar una fecha para arreglarlo.
- Dejar **comentarios en las funciones** que se creen, ser **consistente en la estructura de directorios** del sistema y pedir una segunda opinión del trabajo realizado (**Política de revisión de código)**.
- En general, sea código o no, **pedir una segunda opinión** para que revise el trabajo.

P-QA-Revisión de código:

- Para que una ***petición de cambio*** se apruebe se debe realizar una Pull Request indicando claramente qué cambios se han realizado.
- **La PR no puede dar conflictos**, si los diera es el miembro que realizó la petición hacer un nuevo commit en la PR resolviendo los conflictos.
- **Los test de integración continua deben pasar sin errores.**
- **El revisor debe proporcionar comentarios constructivos** y, si es necesario, solicitar correcciones antes de aprobar la fusión.

3.6. Políticas de gestión de tareas(incidencias)

P-ISSUE-Control de las tareas:

- Se utiliza **GitHub Projects** para la gestión de tareas, y a la hora de documentarlas, estas seguirán una plantilla.
- Durante las reuniones de progreso el **Scrum Master asigna a cada miembro del equipo una tarea por afinidad** y se establecen plazos claros para su finalización.
- En caso de indiferencia se realiza asignación por parte del Scrum Master según habilidad o por asignación aleatoria.
- Cada tarea desarrollada se dará por completada cuando se realice una revisión de la misma por miembros que no hayan participado directamente en la tarea.
- Se realizan **reuniones regulares de control** para evaluar el progreso y ajustar la gestión de tareas según sea necesario.

P-ISSUE-Documentar incidencia:

Toda incidencia deberá indicar lo siguiente:

- **El identificador de la incidencia**
- **La descripción de la tarea a realizar:** Debe indicar de forma resumida lo que hay que hacer
- **El tipo de incidencia:** Indica a qué apartado del proyecto afecta la incidencia (**calidad, código, integración,pruebas,despliegue,documentación, construcción**).
- **La prioridad de la incidencia:** Indica la prioridad, y debe estimarse según la importancia e impacto del mismo sobre el avance del proyecto:**Minimal,Low,Medium,High,Very High y Critical.**
- **Rol/es de la incidencia:** Indica quién o quiénes son las personas que se van a encargar de solventar la incidencia. Normalmente esas personas serán las mismas que originaron la incidencia(ya sea algo que hayan desarrollado,probado o integrado).
- **El estado de la incidencia:** Indica el estado actual de la incidencia:**Todo,In Progress,In Review, Done.**

P-ISSUE-Clasificar tipo de la incidencia:

Se categorizan de la siguiente forma:

- **Calidad:** Cuando la incidencia afecta a un aspecto de calidad del proyecto. En general, si una incidencia es de calidad, la prioridad no debe llegar a ser Alta
- **Código:** Cuando la incidencia afecta a una funcionalidad del proyecto.
- **Integración:** Cuando la incidencia afecta a la integración continua de la aplicación.
- **Pruebas:** Cuando la incidencia afecta a una o varias pruebas sobre la aplicación.
- **Despliegue:** Cuando la incidencia afecta al comportamiento del despliegue de la aplicación.
- **Documentación:** Cuando la incidencia ha aparecido en la documentación del proyecto.
- **Construcción:** Cuando la incidencia afecta a los script de construcción de la aplicación.

P-ISSUE-Asignar prioridad a la incidencia:

La prioridad de la incidencia dependerá de dos cosas: **el impacto y la urgencia.**
La urgencia dependerá del tiempo en el que el equipo quiere solucionar la incidencia.
El impacto dependerá de los efectos sobre el proyecto que son resultantes de la incidencia.
Aquella persona que se encargue de documentar la incidencia tendrá que pensar en el impacto y la urgencia de solucionar esa incidencia. Los tipos de prioridad de una incidencia son:
Minimal,Low,Medium,High,Very High y Critical

P-ISSUE-Actualizar estado de la incidencia:

- Al añadir una nueva incidencia, el estado predeterminado será **Todo.**
- Una tarea pasará a **In Progress** cuando se esté trabajando sobre la misma.
- Una vez completada la tarea, pasará a **In Review**, en donde esperará la tarea a ser revisada, y una vez revisada y aceptada, pasará al **Done**
- Cuando se pase alguna tarea a **In Review,** los encargados deberán hacer un **Pull Request** de la tarea realizada y deberán comunicar al grupo sobre esta petición de revisión.
- Toda actualización de la incidencia deberá también actualizarse sobre el registro de incidencias.

4. Bibliografía

Intencionalmente en blanco

5. Anexos

Intencionalmente en blanco