UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Dirección y Gestión de Proyectos

Manual de Coordinación

Doble Grado Ingeniería Informática y Matemáticas

Doble Grado Ingeniería Informática y ADE

Autores:
Bolívar Peláez, Clara
Borrajo Yusty, Valeria
Gómez López, Javier
Ruiz Gómez, Soledad

Octubre 2024

Índice

1. METODOLOGÍA DESARROLLO Y CICLO DE VIDA **2**

2. RECURSOS SOFTWARE DESARROLLO **2**

3. ORGANIZACIÓN DEL EQUIPO DE TRABAJO (ESTRUCTURA, NORMAS) **2**

3.1. Disponibilidad de los integrantes 3

4. HERRAMIENTAS PARA COMUNICACIONES EN EL EQUIPO DE TRABAJO **3**

5. RELACIONES CON EL CLIENTE (ENTREVISTAS, REUNIONES, REVISIONES, . . .) **3**

6. ESTÁNDARES DE DOCUMENTACIÓN **3**

7. ESTÁNDARES DE CÓDIGO **3**

8. PLAN DE GESTIÓN DE CAMBIOS **4**

9. CONTROL DE VERSIONES (MÉTODO Y HERRAMIENTAS) **4**

10. GESTIÓN DE CALIDAD (DURANTE EL DESARROLLO Y AL FINAL, INCLUIR HERRAMIENTAS) **4**

11. PLAN DE MEDICIÓN DEL DESEMPEÑO, RECOMPENSASA **4**

1. METODOLOGÍA DESARROLLO Y CICLO DE VIDA

Vamos a utilizar una metodología ágil basada en iteraciones. Hay tres iteraciones: las marcadas por las profesoras de la asignatura. Al final de cada iteración se revisa el trabajo realizado en una reunión con el cliente (que son las profesoras y el Colegio *San Rafael*) y se hacen los ajustes necesarios. Se procurará tener un prototipo que mostrar al cliente al final de cada iteración.

Además, se harán reuniones diarias que consistirán en una actualización del trabajo diario de cada participante que deberá hacerse antes de las 8:00PM por el grupo de `WhatsApp`.

Cada vez que cumplamos con los objetivos impuestos, organizaremos una reunión bajo demanda. En esta verificaremos los propósitos ya realizados y fijaremos nuevos. La reunión será en persona y, en su defecto, vía `Google Meet`.

2. RECURSOS SOFTWARE DESARROLLO

- **Frontend:** React, CSS, Javascript, HTML, Bootstrap
- **Backend:** Django
- **Base de Datos:** PostgreSQL
- **Control de Versiones:** GitHub
- **Gestión de Tareas:** Jira y Google Sheets
- **Comunicación Interna:** WhatsApp y Google Meet

3. ORGANIZACIÓN DEL EQUIPO DE TRABAJO (ESTRUCTURA, NORMAS)

Nos repartiremos roles rotatorios en cada iteración para que todos seamos coordinadores. Los roles serán:

- **Coordinador:** se encargará de asignar los objetivos semanales de cada uno y supervisar su cumplimiento, además de defender la solución propuesta en cada iteración.
- **Catalogador:** se encargará de generar cualquier tipo de documentación (exceptuando la documentación inherente al código, de lo cual se encarga cada uno) y de gestionar las ramas de trabajo en `GitHub`.
- **Gestor de calidad:** se encargará de revisar que la solución sea ejecutable y funcione correctamente antes del final de cada iteración, asegurarse de que se cumplen los objetivos en las reuniones bajo demanda, se documentan adecuadamente, y se reparte de manera justa el trabajo.
- **Gestor de accesibilidad:** se encargará de revisar el trabajo de todos los compañeros al final de cad iteración, en pos de aplicar validadores de accesibilidad a la aplicación web para comprobar su grado de accesibilidad. Además, en medio de las iteraciones irá hablando con los compañeros para generar informes que formarán parte de la documentación y que describirán el progreso del proyecto en cuanto a usabilidad y accesibilidad.

Como ya hemos comentado, al final de cada semana, se hace una reunión para ver el trabajo realizado y ponerlo en común. En principio tendrán lugar los viernes a las 4:00pm. Además, hay un *daily meeting* que consistirá en informar del trbabajo diario por `Whatsapp`.

3.1. Disponibilidad de los integrantes

- **Soledad:** Jueves, sábado y lunes completos.
- **Javier:** Entre semana tardes
- **Valeria:** Fines de semana enteros, lunes y martes por la tarde.
- **Clara:** Todos los días por la mañana.

4. HERRAMIENTAS PARA COMUNICACIONES EN EL EQUIPO DE TRABAJO

Tenemos un grupo de `WhatsApp` para la comunicación diaria. Si hiciesen falta reuniones online serían a través de `Google Meet`.

A parte, nos vamos a apoyar en `Google Calendar` para la organización y para la gestión de tareas con `Jira`.

5. RELACIONES CON EL CLIENTE (ENTREVISTAS, REUNIONES, REVISIONES, . . .)

La primera reunión con el cliente será el 27 de Septiembre de 2024.

El coordinador se encargará de concertar una reunión con el cliente, como mínimo, en cada iteración. Con las profesoras (un componente del cliente en su totalidad) se podrán concertar reuniones con más frecuencia, en caso de que el equipo lo considere necesario a raíz de nuevas dudas o cambios. Todas las reuniones han de concertarse vía mail.

Al final de cada iteración se hará una entrega de prototipos y feedback al cliente. El coordinador se encargará de presentar la solución.

6. ESTÁNDARES DE DOCUMENTACIÓN

Usaremos LaTeX para generar todo tipo de documentación técnica. La estructura estándar de la documentación constará de Propuesta Técnica, Requisitos, Casos de Uso, Actas y cualquier documentación generada a lo largo del desarrollo.

En cada reunión (tanto de equipo como con el cliente), el catalogador se encargará de redactar un acta que refleje las decisiones tomadas y los cambios o tareas a realizar.

En cuanto a la documentación del código, comentaremos brevemente antes de cada función la siguiente información: parámetros de entrada, parámetros de salida y breve comentario de la funcionalidad. Además, separaremos cada fichero en secciones: variables (globales), y clases o funciones.

También se llevará un registro de horas dedicadas al proyecto, donde cada persona indicará cuánto tiempo ha invertido, cuándo lo ha hecho y con qué finalidad (a qué lo ha dedicado). Esta información será útil sobre todo de cara al funcionamiento del equipo, pues nos permitirá regularnos si hay algún individuo que esté trabajando de más o de menos, repercutiendo así de manera positiva en nuestra eficiencia como grupo de trabajo. Se registrará en `Google Sheets`.

7. ESTÁNDARES DE CÓDIGO

El catalogador se encargará de la creación y fusión de ramas de trabajo en `GitHub`.

Usaremos *ESLint* (se encargará de detectar errores y problemas de calidad en el código) y *Prettier* (se encargará del formateo y el aspecto visual del código) para mantener un estilo de código consistente y sin errores.

Para la nomenclatura de las funciones o variables compuestas por más de una palabra, usaremos `CamelCase`.

Se redactará el código en inglés.

La documentación de las funciones ya ha sido especificada en el apartado anterior. Importante recalcar que, siempre que sea necesario, dividiremos el código de un mismo fichero en las secciones ya mencionadas.

8. PLAN DE GESTIÓN DE CAMBIOS

Dependiendo del tipo de cambio, se tomarán unas medidas u otras.

En caso de que un miembro no cumpla con su trabajo asignado alguna semana (por cualquier motivo) y, por consecuente, se genere un retraso, se discutirá con el equipo al completo si hay que realizar algún ajuste adicional. En caso de generarse un retraso inevitable, se hablará con la profesora para valorar posibles ajustes del proyecto.

Por otro lado, todos los cambios que se realicen respecto en los requisitos del proyecto serán discutidos en la reunión bajo demanda. Además, posteriormente deben ser aprobados por el cliente. Sin estas dos validaciones, no se pueden implementar esos cambios.

9. CONTROL DE VERSIONES (MÉTODO Y HERRAMIENTAS)

Vamos a usar `GitHub` con un esquema sencillo de rama *Main* y una de *Development*. Además, cada uno de los intgegrantes del equipo tendrá una rama para realizar su trabajo. Cuando se necesiten más ramas a medida que avancemos en el desarrollo de la app, se irán creando por el catalogador, previo acuerdo del equipo entero.

Al final de cada iteración, el prototipo funcional se subirá a la rama *Main*.

En cualquier otro momento del desarrollo del proyecto, esta rama no se tocará. La rama *Developmet* sólo se actualizará si el coordinador da permiso para ello. En cada reunión bajo demanda se hará revisión del trabajo individual de cada integrante así como de los objetivos cumplidos para evaluar si es apto de subirlo a *Development*. A lo largo de la semana, si algún integrante quisiera hacer alguna modificación en esta rama, ha de informar al equipo entero y el coordinador dará el visto bueno o no.

Cada uno puede actualizar su propia rama libremente y, en caso de necesitar la creación de nuevas ramas, deberá hablar tanto con el catalogador (que creará la rama) como con el coordinador (que dará el visto bueno).

10. GESTIÓN DE CALIDAD (DURANTE EL DESARROLLO Y AL FINAL, INCLUIR HERRAMIENTAS)

Cada integrante del grupo debe comprobar que todo funcione bien (no hay problemas de ejecución ni lógicos) antes de implementar nuevas funcionalidades. Además, debe asegurar el funcionamiento de lo que haya implentado al terminar.

Antes del final de cada iteración, el gestor de calidad es responsable de verificar que el prototipo esté funcionando y listo para poder enseñárselo al cliente.

El gestor de calidad también revisará a lo largo de la semana (o al final de esta) que los integrantes hayan seguido las pautas de organización del proyecto.

Como herramientas que faciliten el papel del gestor de calidad, encontramos *ESLint*, *Prettier*, *GitHub*, *Google Calendar* y *Jira*.

11. PLAN DE MEDICIÓN DEL DESEMPEÑO, RECOMPENSA-SA

Si un compañero ha necesitado ser cubierto en su trabajo una semana, para la siguiente que tenga disponibilidad debe retornar el favor a quien le haya cubierto.

Si alguien hace su trabajo mucho mejor que el resto de compañeros, le felicitaremos públicamente ante la profesora y pediremos un incremento en su calificación final. En el caso en el que alguien se descuelgue del grupo y no colabore, haremos lo mismo, pero del revés.

Después de la defensa final, nos iremos de cervezas a celebrar un buen trabajo hecho.