DOCUMENTO DE PROYECTO

Grupo:
decide-single-juanymedio-fork-3
Miembros:
Julio Navarro Rodríguez
Pablo Martínez Valladares
Manuel Palacios Pineda

1. Introducción

El propósito de este documento es proporcionar una revisión exhaustiva y detallada del proyecto recientemente completado, centrándonos tanto en el desempeño de los miembros del equipo como en los aspectos técnicos y funcionales del sistema desarrollado.

Este análisis tiene como objetivo no solo reconocer los logros y desafíos enfrentados por el equipo durante el proceso del proyecto, sino también proporcionar una comprensión profunda de las características, funcionalidades y la arquitectura del sistema; así como su configuración y el proceso de integración de cambios del proyecto.

2. Indicadores del proyecto

Miembro	Horas	Commits	LoC	Tests	Issues	Incremento
Martinez Valladares, Pablo	60			6		- Votaciones binarias - Workflow de integración continua - Documentación del proyecto
Navarro Rodríguez, Julio	60		257	8		- Grupos censitarios reutilizables - Importación de censo mediante CSV - Despliegue en Docker - Documentación del proyecto
Palacios Pineda, Manuel	20	0	0	0	1	- Documentación del proyecto
TOTAL	140			14		

3. Integración con otros equipos

En principio, el grupo se llamaba **decide-part-juanymedio-3**, ya que éramos parte de un equipo *part* con otros dos grupos:

- **decide-part-juanymedio-1**
- **decide-part-juanymedio-2**

Cada grupo escogió los módulos que querían modificar y si surgían dependencias con módulos de otros grupos, era necesaria la coordinación entre miembros de ambos grupos. Por ejemplo, para desarrollar la característica "Votación con varias preguntas" (módulo voting, seleccionado por el grupo 3), era necesario realizar cambios en la vista de la cabina de votación (módulo booth, grupo 1).

Después del M2, dadas las dificultades en la coordinación entre grupos, se optó por una separación del equipo, y cada grupo pasó a ser un grupo *single*.

4. Resumen ejecutivo

Nuestro grupo cuenta con pocos incrementos, debido a la ausencia de la mayoría de integrantes y su abandono en la semana anterior al M3. Los incrementos realizados por parte del grupo son los siguientes:

- Votaciones binarias (Sí/No): La app debe ser capaz de ofrecer un nuevo tipo de votación: las votaciones binarias. En estas votaciones, el usuario votante deberá responder a una pregunta con sólo dos opciones: Sí o no.

- Grupos censitarios reutilizables: La app debe ser capaz de poder agrupar usuarios para poder reutilizar dichos grupos para censos de futuras votaciones.

- Importación de censo mediante CSV: La app debe ser capaz de recibir un fichero CSV y crear a partir de él el censo de una o varias votaciones.

5. Descripción del sistema

Decide está compuesto por varios subsistemas que pueden funcionar de manera aislada, estos subsistemas se conectan entre sí para crear el sistema Decide. El conjunto de subsistemas y el propio sistema principal tienen definida una API la cuál puede ser consumida.

A continuación, se enumerarán los distintos subsistemas de la aplicación, su funcionamiento y cómo afectan nuestros incrementos a cada uno de ellos.

5.1. Autenticación

El subsistema de Autenticación es un componente crucial en cualquier sistema de votación electrónico. Su función principal es verificar la identidad de cada votante para garantizar que cada persona está autorizada a votar y que lo hace una única vez, evitando así la duplicidad de votos y el fraude electoral.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

5.2. Censo

Se encarga de administrar y mantener el censo de cada votación, es decir, el conjunto de individuos elegibles para participar en una votación específica. Este subsistema asegura que solo las personas autorizadas puedan votar en una elección determinada, manteniendo la integridad y la legitimidad del proceso electoral.

Este módulo ha visto incrementada su funcionalidad después de implementar los incrementos “Grupos censitarios reutilizables” e “Importación de censo mediante CSV”. Estos incrementos no modifican la funcionalidad básica del módulo, simplemente permiten realizar nuevas operaciones que facilitan la gestión del mismo; agilizando principalmente el proceso de asignar posibles votantes a una o varias votaciones de forma inmediata.

5.3. Votaciones

Su propósito es facilitar la creación, configuración y administración de las votaciones. Cada votación debe tener asignada un nombre, una descripción, una fecha de inicio, una autorización, una clave pública y una pregunta. Esta pregunta, a su vez, debe tener asignadas dos o más opciones las cuales deben ser elegibles por los votantes.

Después de implementar el incremento “Votaciones binarias”, ahora las preguntas tienen asignadas también un atributo “question_type”, el cual puede ser “MCQ” (multiple choice question, lo que llamaríamos una votación estándar) o “YN” (yes/no, lo que llamaríamos una votación binaria). Al crear una pregunta y seleccionar el valor “YN”, automáticamente se crearán las opciones “Sí” y “No”, y no se podrán agregar más opciones a la pregunta.

5.4. Cabina

Su objetivo principal es proporcionar una interfaz de usuario clara y accesible para permitir a los votantes emitir sus votos de manera sencilla y segura. Este subsistema se centra en la experiencia del usuario, asegurando que el proceso de votación sea intuitivo, directo y libre de complicaciones. Además, incorpora la comprobación de acceso mediante el censo al pedir el inicio de sesión del usuario.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

5.5. Almacenamiento

Este componente se encarga de almacenar de forma cifrada los votos en la base de datos, asegurando la privacidad de la intención de voto de cada elector, al mismo tiempo que mantiene un registro de quién ha votado y quién no.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

5.6. Recuento

Utiliza una red de MixNet, que es un conjunto de nodos (ordenadores) que colaboran para cifrar los votos utilizando una clave compartida. Este enfoque garantiza que los votos sólo puedan ser descifrados cuando todas las autoridades involucradas en el proceso de votación lleguen a un consenso, proporcionando un alto nivel de seguridad y confidencialidad.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

5.7. Post-Procesado

El subsistema de Procesamiento de Resultados de Votación es responsable de interpretar y traducir los datos crudos obtenidos del recuento de votos en resultados comprensibles y significativos. Este componente toma la lista de números generada tras el recuento y la analiza para determinar los resultados finales de la votación, adaptándose a las diferentes modalidades de votación empleadas.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

5.8. Visualización

Tras procesar y determinar los resultados de la votación, este subsistema se encarga de transformar los datos numéricos en representaciones gráficas y tablas, facilitando su comprensión y análisis tanto para administradores electorales como para el público general.

Este módulo no se ha visto afectado ni mejorado por ningún incremento realizado a lo largo del desarrollo.

6. Visión global del proceso de desarrollo

Supongamos que queremos implementar un cambio dentro de nuestro proyecto de Decide. Dicho cambio consiste en hacer gráficas en tiempo real para la visualización de los resultados de una votación. A continuación, se muestra un ejemplo de todo el proceso para implementar un cambio en el proyecto.

6.1. Rama de la tarea

El primer paso para implementar un cambio es crear una rama dentro del proyecto, para poder trabajar en la nueva característica de forma independiente. La gestión de ramas se explica con más detalle en el documento *Plan de gestión del código fuente*, dentro de esta misma carpeta. Esta rama se crea a partir de la rama *develop* y puede recibir, por ejemplo, el nombre **Feat-Resultados_tiempo_real**.

6.2. Crear la issue

El siguiente paso consiste en crear la issue dentro de nuestro tablero Kanban, en la columna *Todo*. Suponiendo que, tenemos un gran margen de tiempo para implementar la feature y no tiene dependencias con otras tareas, podemos llamar a la issue **Feat: Resultados en tiempo real [L]**. Dentro de la documentación del proyecto, se encuentra el *Plan de gestión de incidencias*, donde se explica con más detalle el proceso de crear y gestionar una issue.

En la configuración de la issue, debemos indicar una descripción clara de en lo que consiste la tarea. Además, debemos asignar a algún miembro del grupo para que se encargue de desarrollar la feature. A continuación, se deben indicar la etiqueta o etiquetas de la issue. En este caso, marcamos la issue con la etiqueta *code*. Por último, de manera opcional, podemos vincular esta issue con la rama que creamos previamente. Esto no es obligatorio, pero sí es recomendable.

6.3. Desarrollar la feature

Una vez el miembro del equipo asignado comience a trabajar en el desarrollo de la issue, se deberá mover ésta a la columna *In progress* del tablero. Entonces, el miembro asignado se traerá la rama nueva a su entorno local para empezar a desarrollar los cambios pertinentes.

A lo largo del desarrollo de una feature no es nada atípico realizar varios commits antes de finalizar por completo el desarrollo. Estos commits deben seguir la estructura de Conventional Commits.

Supongamos que, el miembro asignado ya ha completado la parte funcional de la issue, y después de probar varias veces de manera informal el comportamiento del código está convencido de que su trabajo es válido. Es entonces cuando decide realizar un commit con su progreso, pero elige dejar los tests para otro momento. El desarrollador haría un commit de este estilo:

feat(resultados_tiempo_real): completada la funcionalidad, excluyendo tests

Cuando llegue el momento en que el desarrollador de la feature tenga que realizar los tests, deberá diseñarlos en base a alguna de estas técnicas:

- Particiones equivalentes (equivalent partitioning): Se divide el espacio de prueba en clases equivalentes. Las particiones deben ser disjuntas. Inconveniente: rápida explosión.

- Valores límite (boundary analysis): Según esta técnica los errores en los programas suelen estar en los valores límites de las entradas de un programa. Se prueban los valores límite de las particiones equivalentes en caso de haberlas.

- Combinaciones par-wise, n-wise: Pairwise testing (también 2-wise testing). Es un método dentro de los llamados “combinatorios” que propone hacer pruebas sobre todas las posibles combinaciones de 2 parámetros de entrada. La hipótesis que maneja es que la mayoría de los errores son debidos a errores en un parámetro de entrada, la siguiente es la combinación de pares de parámetros de entrada, etc.

- Errores conocidos (error guessing): Errores comunes según nuestra experiencia. Ejemplo: cuándo se almacena un voto se puede almacenar con el mismo ID.

- Cobertura CRUD: Para cada elemento persistente, se prueban todas sus operaciones CRUD. Cada caso de prueba comienza por una C, seguida por todas las U y se termina por una D. Tras cada C, U o D, se ejecuta una R que nos sirve de oráculo.

La creación de tests, así como que de estos tests se obtengan unos resultados favorables; es algo obligatorio para poder dar por completado el desarrollo de una tarea.

Una vez el desarrollador vea que todos los tests han pasado correctamente, hará con nuevo commit parecido a este:

test(resultados_tiempo_real): tests completados

6.4. Pull requests

Luego de que el desarrollador asignado haya hecho un push a la nueva rama con los cambios realizados, es el momento de hacer una pull request para integrar estos cambios en la rama *develop*. Para ello, creamos una nueva pull request que incorpore los cambios de la nueva rama a *develop*. En el cuerpo de la pull request es muy importante incluir la siguiente expresión (sin paréntesis):

closes #(id-issue)

Esto se hace con la intención de vincular esta pull request con la issue que intentamos cerrar.

Cuando se crea la pull request, automáticamente se lanza el workflow de GitHub Actions que tenemos implementados para realizar la integración continua. Se va a ejecutar un build del proyecto en un servidor de CI, se van a ejecutar todos los tests y se va a ejecutar un análisis de código estático mediante Codacy. Si todos estos jobs se ejecutan correctamente, se puede aceptar la pull request y mergear la rama *develop*. En este punto, la issue ya se da por finalizada, por lo que podemos moverla en el tablero a la columna *Done* y darle a *Close issue*.

Supongamos que este es el único cambio que queremos enviar a producción. Para ello, debemos crear otra pull request (esta vez de *develop* a *master*), pero en este caso, debemos asignar al coordinador del grupo como revisor, ya que la rama *master* está protegida. De nuevo, se ejecutarán los workflows de CI y si todo se ejecuta correctamente, el coordinador puede aceptar la pull request.

7. Entorno de desarrollo

7.1. Espacio de trabajo y herramientas

- Sistema operativo: Linux
- IDE: Visual Studio Code
- Despliegue: Docker y Vagrant

7.2. Proceso de instalación

Para el desarrollo de este proyecto se han seguido las instrucciones y utilizado las versiones proporcionadas por el profesorado en las prácticas de la asignatura.

Actualizar paquetes de ubuntu:

sudo apt update

Instalar Python 3.10:

sudo apt install python3.10 python3.10-dev python3.10-venv python3-pip

Instalar Postgres:

sudo apt install postgresql libpq-dev

Crear y activar entorno virtual:

python3.10 -m venv egcenv
source egcenv/bin/activate

Instalar y configurar git:

sudo apt install git
git config --global user.name "[[nombre]]"
git config --global user.email [[email]]

Clonar el repositorio:

git clone https://github.com/decide-single-juanymedio-fork-3/decide

Instalar dependencias del proyecto:

pip install -r requirements.txt

Crear base de datos con postgres:

sudo su - postgres
psql -c "create user decide with password 'decide'"
psql -c "create database decide owner decide"

8. Ejercicio de propuesta de cambio

En el apartado *Visión global del proceso de desarrollo* de este mismo documento se explica paso por paso cómo evolucionaría una propuesta de cambio dentro del proyecto. Además, contamos con otros dos documentos: *Plan de gestión de incidencias* y *Plan de gestión del código fuente*, donde este procedimiento es explicado de forma más detallada con ayuda de capturas de pantalla.

9. Conclusiones y trabajo futuro

El equipo ha adquirido y desarrollado nuevas habilidades de cara a trabajar en proyectos de mayores dimensiones a los previamente vistos. La utilización de una tecnología no vista previamente ha supuesto una mejora en la capacidad de adaptación individual de cada miembro.

Como principal sugerencia, proponemos que se documente en mayor detalle el código de Decide. Pese a que en la wiki del repositorio original de Decide se explica el funcionamiento de cada subsistema; al principio del curso, dedicamos muchísimo tiempo en tratar de entender el funcionamiento del código de la app para poder comenzar a desarrollar incrementos. Este período de adaptación sería mucho más corto si el código estuviese documentado.