Proyecto 03
Retrospectiva

Universidad Nacional Autónoma de México
Facultad de Ciencias
Modelado y Programación

3 de Octubre de 2019

1 Introducción

Ninguna persona nace sabiendo, el conocimiento se adquiere con el esfuerzo, el estudio y la practica. Desde que se entra a la carrera, cada uno ha crecido como programador a partir de practica y conocimiento y sobretodo esfuerzo.

Cuando se inicio el curso muchos de ustedes solamente dominaban 2 lenguajes, pero ahora se espera que al menos tengan ya una noción general de nuevos lenguajes y una nueva practica, que es migrar entre ellos.

Existe un mundo de lenguajes, unos más famosos que otros, pero la computación es una profesión que esta en constante cambio, por lo que no solo es necesario aprender un lenguaje, si no aprender realmente lo que significa programar, y así no importa que tanto avance las nuevas tecnologías, ustedes puedas evolucionar con ellas, y a pesar de ellas.

2 Desarrollo

Tu reto de esta semana mirar al pasado y ver que tanto haz crecido como programador, y evaluar que tanto mejoraste, o que necesitas en su defecto seguir mejorando.

Deberás hacer un equipo de 3 personas, y en equipo escoger un proyecto del pasado que a los 3 les llame la atención. Una vez autorizado el proyecto, tu misión es mejorarlo, para esto sigue las siguientes instrucciones.

- Agreguen a sus respectivos equipos a sus repositorios de *github*, no olviden borrar a su equipo anterior.
- Creen una rama cada quien, recuerden que previamente debieron seguir todo el análisis del problema y la división del mismo.
- Usen un mínimo de 5 ramas, y no olviden hacer rebase, en caso que uno de sus compañeros subiera un cambio a *master*.
- Recuerden que primero van las pruebas, si no como saben que arreglaron algo.
- Subirán el proyecto pasado como el nuevo proyecto para comparar las diferencias.

También deberán cumplir los siguientes incisos.

- Deberas hacer al menos 3 mejoras a tu proyecto (interfaz gráfica, menú, etc.) mejoras necesarias para un funcionamiento correcto no se contarán como mejoras, osea, arreglar lo que impedia que compilará. Una mejora contará solo si mejora su funcionalidad, eficacia, etc. (Recuerden las cualidades del software. Y deberán estar correctamente justificadas en tu reporte.
- Deberá contar con 5 pruebas nuevas, incluye a tus mejoras y deberán estar justificadas en tu reporte. Usa las pruebas como ayuda para programar, recrea bug o problemas que te aparezcan e intenta solucionarlos.
- Haz un archivo de bash que corra tu proyecto al menos 1,000 veces, y que tome el tiempo promedio en que tardo en ejecutarse.
- Pon en una situación extraordinaria a tu código nuevo, una situación que antes tu proyecto antiguo no podía lidiar y anótalo en tu reporte.
- Utiliza un nuevo lenguaje de programación.

Recuerda seguir el análisis que siempre se solicita en cada uno de los proyecto y explicarlo en un pdf, bien redactado que tenga la siguiente estructura.

- Introducción
- Definición del problema (analicen como se definió antes el problema y si se requirió cambiar algo en la redacción)
- Análisis del problema (analicen ambas soluciones y los cambios propuestos)
- Selección de la mejor alternativa (¿Por qué mejoraría ahora tu proyecto?) Explica un poco el lenguaje que se esta utilizando y el por qué.
- Pseudocódigo (Bien hecho y de ambas implementaciones)
- Pruebas y Mejoras
- Análisis estadístico
- Situación extraordinaria
- Plan a futuro

Se deberán distribuir el trabajo de manera equitativa y anotar que cosa hizo cada quien. Se puede usar la IDE que quieran, pero se espera un proyecto ordenado, y sobre todo amigable (fácil de usar).

Se manejara Github en equipo, por lo que ten en cuanta las siguietes recomendaciones:

- Agrega primero tus pruebas en *master*, al igual que cualquier documento, carpetas o/y archivos que sean necesarios para correr tu programa.
- Crea con ***git checkout -b "nombre_de_tu_rama*** una rama por filtro, no creen ramas de más, deben pensar en nombres de las ramas que describan el funcionamiento del archivo. Al igual que los *commits* deben ser a conciencia, describiendo el cambio que hicieron.

- Creen al menos 4 *commits* por rama, y solo haga *merge request* cuando su código pase las pruebas. Recuerda que si quieres cambiarte de rama y tienes cambios en tu rama actual, simplemente haz ***git stash*** para guardar tus cambios y cambiarte entre ramas sin tener que hacer *commits* innecesarios, y ***git stash pop*** para sacarlo del *stash*.
- Jamás hagan –***force*** si están trabajando en una rama con un colaborador.
- Finalmente siempre haz ***git status*** para ver los cambios que hiciste y también para saber que es lo que vale o no la pena agregar y agregarlo con ***git add "path de lo que quieres agregar"***. Muchas veces se crean archivos que contienen información delicada como contraseñas o simplemente pruebas o archivos inútiles y muchas veces no queremos borrarlos, para eso te ayuda especificar que quieres agregar.

Recuerda que debe estar bien documentado todo tu código.