TEC | Tecnológico de Costa Rica

Anexo Proyecto III: GOT

Algoritmos y Estructuras de Datos II

Integrantes:

José Alejandro Chavarría Madriz
Harold Espinoza Matarrita
Natalia González Bermúdez
Sebatian Moya Monge

Profesor:

Antonio González Torres

Semestre I
2020

Tabla de contenidos

Introducción

GOT es un programa manejador de versiones para archivos de texto que permite la colaboración entre varios usuarios. GOT se maneja mediante una aplicación de consola programada en C++ que permite gestionar los archivos mediante diferentes comandos escritos en consola. La aplicación fue creada con el fin de permitir la colaboración entre usuarios en la elaboración de archivos, brindar a los usuarios una base de datos segura donde guardar sus archivos y la posibilidad de recuperar versiones anteriores de su trabajo. Con el fin de explicar al mayor detalle las capacidades y el funcionamiento de la aplicación, he aquí un breve resumen

got init < name > : Este comando se encarga de crear un repositorio donde serán guardados todos los archivos deseados y sus versiones. También se encarga de mantener organizados los archivos en el área local.

got help : Despliega la información necesaria para poder utilizar GOT. Es una ayuda que da una explicación básica de los comandos.

got add [-A] [name] : Permite agregar a la pila de archivos que se desean enviar a la base de datos, los archivos especificados. Si se especifica el comando [-A] este añade todos los archivos con cambios nuevos. Al ser agregados quedan pendientes de commit.

got commit < mensaje > : Envía todos los archivos pendientes al server, esto acompañado de un mensaje. Al hacer esto se retorna un número commit. Es importante mencionar que el servidor verifica que el usuario esté al día con los demás commits, sino rechaza el del usuario.

got status < file >: Muestra los cambios que se realizaron en el último commit, si se especifica el archivo, esté muestra el historial de cambios que se encuentra en el server.

got rollback < file > < commit >: Permite retornar un archivo en específico a un commit en específico que se encuentre en el server.

got reset < file>: Deshace los cambios locales de un archivo y los devuelve al último commit del server.

got sync < file >: Recupera los cambios que se encuentran en el server de un archivo en específico y los añade al local, si hay cambios en el local, el usuario decide con cual quedarse.

La base de datos utilizada es MySQL y se comunica con la aplicación mediante un API programado en Javascript utilizando Node.js

Equipo de trabajo

Lineamientos

- Respetar las decisiones tomadas como grupo.
- Cumplir con los horarios establecidos (reuniones, entregas, etc).
- Informar al grupo antes de realizar algún cambio que se considere importante en el proyecto.
- Responder los mensajes relacionados al trabajo con menos de 24h de diferencia.
- Velar por la justa repartición del trabajo.
- Al organizar una reunión se debe velar por la conveniencia de horario para los miembros del grupo. De no llegar a un acuerdo, gana la mayoría democrática. En caso de empate, se lanza una moneda.
- Separar lo personal y lo profesional.
- Respetar las convenciones para el manejo de GIT.

Roles y responsabilidades

Coordinador: Como coordinador recae la responsabilidad de plantear la arquitectura general del proyecto, lo que conlleva tener un conocimiento general de las partes del mismo. También recibe la responsabilidad de convocar las reuniones de grupo y repartir de forma general el trabajo entre los miembros del grupo. Y asegurarse que cada miembro trabaja dentro de los lineamientos establecidos.

Integrante: Como integrante recae la responsabilidad de fiscalizar las decisiones del coordinador, y compartir cualquier disconformidad al respecto. Además es importante que profundice el conocimiento de la sección del proyecto que se le ha asignado, con el fin de poder reportar a los demás miembros del grupo de manera efectiva los avances realizados.

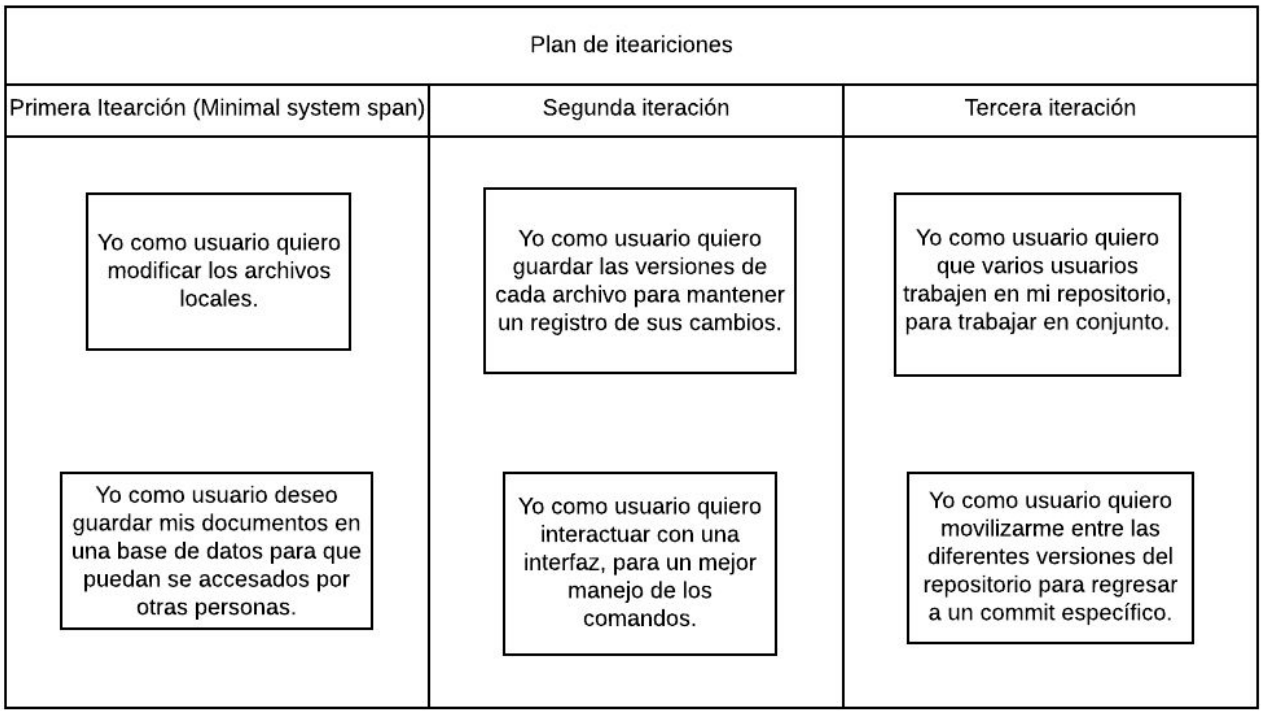
José: Se le fue asignado el rol de coordinador. Específicamente se le asigna el desarrollo del API que permite la comunicación de la aplicación con el servidor.

Harold: Se le fue asignado el rol de integrante. Específicamente se le asignan funciones de desarrollo en la base de datos y el GUI.

Natalia: Se le fue asignado el rol de integrante. Específicamente se le asignan funciones de desarrollo en el GUI, sobretodo en encriptación y compresión de documentos

Sebastián: Se le fue asignado el rol de integrante. Específicamente se le asignan funciones de desarrollo en el la base de datos, y algunas funciones adicionales en el GUI.

Plan de iteraciones



Carga de los miembros por iteración

Se describe la tarea por realizar y su peso porcentual para la finalización de esa iteración:

Primera iteración

Jose: Trabajar en el API para comunicación de la aplicación con la base de datos. (33%)

Harold: Trabajar en el desarrollo de la estructura de la base de datos. (16.5%)

Natalia: Trabajar en las instrucciones de la aplicación (34%)

Sebas: Trabajar en el desarrollo de la estructura de la base de datos. (16.5%)

Segunda iteración

Jose: Trabajar en la integración del API con la aplicación. Definir los métodos que debe utilizar la base de datos. (25%)

Harold: Trabajar en el desarrollo de las funciones de la base de datos. (25%)

Natalia: Encriptación y compresión de datos y construir requests HTTP para el API (25%)

Sebas: Trabajar en el desarrollo de las funciones de la base de datos. (25%)

Tercer iteración

Jose: Pruebas con varios usuarios (10%)

Harold: Trabajar en funciones de merge (35%)

Natalia: Trabajar en implementación de “gotigneore” y “add” (20%)

Sebas: Trabajar en funciones de comparación y reemplazo de archivos (35%)

Bitácora

1/08/20

- Natalia:
- Sebastian: planteamiento de ideas iniciales para el desarrollo del proyecto

2/08/20

Se realiza una reunión grupal para discutir las generalidades del proyecto. Se habla de los principales retos, ideas al respecto y la estructura del proyecto.

4/08/20

- Jose: Creación de este documento
- Sebastian: Creación del repositorio

5/08/20

- Sebastian: Inicio de la documentación externa en la plataforma de versionamiento GitHub

6/08/20

- Harold: Investigación base de datos
- Sebastian: Investigación sobre manejo de base de datos

7/08/20

- Jose: Se genera el primer documento para el API. Se decide utilizar node.js para desarrollar el API en Javascript

8/08/20

- Jose: Se genera una segunda propuesta para la base de datos. Se crean en el API las direcciones respectivas a cada uno de los comandos que se van utilizar. Se investiga acerca de la utilización de librerías en C++.
- Sebastian:

9/08/20

- Jose: Se investiga acerca de librerías para realizar requests HTML desde C++. Se implementa “cpr” con Cmake para dicho fin. Se integran del API y un programa simple en C++ que utiliza cpr.
- Harold: Se investiga acerca de creacion de base de datos
- Natalia: Se inicia la aplicación de consola en C++ y se crea la clase Manager que se encargará de controlar la información que llegan de los mensajes de consola del usuario, se hacen diferentes pruebas par el manejo de archivos de texto y unas funciones para obtener el contenido de un archivo en un string y poder manejarlo y además para escribir en un archivo. También se integra el algoritmo MD5 para la encriptación del mensaje del commit.

10/08/20

- Jose: Se trabaja en documentación
- Harold: Se realiza todo el procedimiento de instalación para el manejo de base de datos en mysql utilizando mysql workbench
- Natalia: Se investiga acerca del algoritmo de Huffman para comprimir los archivos de texto y se integran las funciones para codificar y decodificar un archivo y obtener tanto la información del archivo comprimida y su respectivo diccionario.
- Sebastian: creación de la estructura de la base de datos

Se realiza una segunda reunión grupal para discutir los avances del proyecto. Se toman decisiones de los principales retos, ideas al respecto y la estructura del proyecto. Y se hace una repartición clara de las tareas por integrante.

11/08/20

- Jose: Se trabaja en documentación.
- Harold:Se investiga sobre los procedimientos y funciones de las bases de datos y así se inicia la creación de la base de los procedimientos que servirán para el manejo los elementos de las tablas
- Natalia: Se completan localmente las funciones de init, add, help y se inicia la función de commit para todos los archivos.
- Sebastian: integración de las tablas en la base de datos, se establecieron las relaciones entre las tablas. Se desarrollaron pruebas de ejecución y también documentación externa.

12/08/20

- Jose: Se trabaja en documentación. Se crean funciones en java para parsear Json en strings más simples para C++. Se agrega la función befriend al API. Se realizan pruebas de integración.
- Harold: Se hace la implementación de algunos de los procedimientos de la base de datos para el manejo de los datos en las tablas
- Natalia: Se completan las funciones de status, rollback, reset, befriend y sync. Para la función de sync se investiga una aplicación llamada vim para poder realizar los merge de los archivos con una pequeña interfaz en la consola utilizando el comando vimdiff.
- Sebastian: apoyo en la creación de los procedimientos de la base datos. También creación de casos de prueba.

13/08/20

- Jose: Pruebas de integración. Corrección de errores.
- Harold: Corrección de sintaxis en el llamado de los procedimientos, se genera la documentación de doxygen
- Natalia: Corrección de funciones de consola y build en Cmake.
- Sebastian: Creación del README de git. Corrección de procedimientos en la base de datos.

Reunión para integrar completamente todas las partes del proyecto.