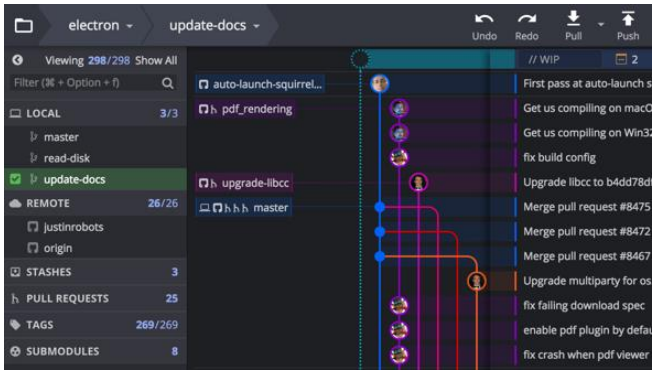
CENTRO DE ENSEÑANZA CONCERTADA
"Gregorio Fernández"

ED_T4_PI: Manual de uso de un Git Client



Los repositorios remotos y la herramienta GIT son utilidades fundamentales para cualquier desarrollador software, combinadas con la gestión de las diferentes ramas de trabajo definidas dentro de la metodología *GitFlow*. Git viene con herramientas de GUI integradas para confirmar (git-gui) y navegar (gitk), pero hay varias herramientas de Git Client que aportan una interfaz diferente. Realiza un manual de uso de una de dichas herramientas que contenga al menos los siguientes puntos:

- Instalar la herramienta elegida, en el caso de que sea necesario, y realizar la configuración inicial. ¿Por qué has elegido esta herramienta Git Client? **(1 pto)**
- Crear un repositorio en GitHub que almacene el proyecto Java llamado AppGitClient, que contenga dos ramas: main y develop. El archivo MANIFEST.MD que contiene el proyecto debe ser ignorado en las copias instantáneas. Recuerda que debes crear el archivo README.MD con la descripción del proyecto. Clonar el repositorio en local. **(0,75ptos)**
- Crear un archivo llamado GitCliente.java en la rama main y subirlo a remoto. **(0,75ptos)**

```java
package appgitclient;

/**
 * gitclient : Describe el tipo de Gitclient elegido
 * @author rhtuf
 */
public class GitClient {


    private String cliente;
    private String version;
    private String sistemOper;
    private String licencia;

    public GitClient() {
    }

    public String getCliente() {
        return cliente;
    }

    public void setCliente(String cliente) {
        this.cliente = cliente;
    }


    public String getVersion() {
        return version;
    }

    public void setVersion(String version) {
        this.version = version;
    }

    public String getSistemOper() {
        return sistemOper;
    }

    public void setSistemOper(String sistemOper) {
        this.sistemOper = sistemOper;
    }

    public String getLicencia() {
        return licencia;
    }

    public void setLicencia(String licencia) {
        this.licencia = licencia;
    }
}
```

- Modificar el archivo anterior en remoto, insertando las siguientes líneas de código dentro de la clase Gitclient.java. Incluir el nuevo archivo en la rama develop. **(0,75ptos)**

```java
@Override
public String toString() {
    return "GitClient{" + "cliente=" + cliente + ", version=" + version + ", sistemOper=" + sistemOper + ", licencia=" + licencia + '}';
}
```

- Ver las diferencias del archivo anterior en local. **(0,75ptos)**
- Actualizar el archivo anterior en local. **(0,75ptos)**
- Eliminar el archivo AppGitclient.java en local en la rama main. **(0,75ptos)**
- Ver el historial de confirmaciones. **(0,75ptos)**

- Actualizar todas las confirmaciones en remoto en la rama main. **(0,75ptos)**
- Crear una nueva rama en local llamada features/newrequest[1]. **(0,75ptos)**
- Crea un nuevo archivo llamado ClientGUI.java en la rama anterior **(0,75ptos)**
- Fusionar los cambios de la rama anterior en la rama develop en local. **(0,75ptos)**
- Subir los cambios realizados a la rama develop en remoto. **(0,75ptos)**

Cada uno de los apartados implementados debe contener una captura de pantalla del manejo de la herramienta en la que se aprecie la correcta ejecución de cada orden en cuestión. El resultado de los cambios también debe quedar reflejado en el repositorio remoto compartido con la profesora en GitHub.