**Software Configuration Management**

GitHub è lo strumento prescelto per la gestione di configurazione dell’intero progetto, sia per il codice sia per la documentazione; a tale scopo, è stato creato un repository, suddiviso in varie parti (branches, “rami”), con la finalità di garantire un’ottima gestione delle singole componenti e in modo tale che il lavoro di un collaboratore non influenzi troppo quello degli altri membri (l’uso di più branch permette di non confondere le modifiche apportate dai tre membri su uno stesso componente).

**ORANIZZAZIONE DELL’AMBIENTE DI LAVORO**

All’interno del repository si trovano diversi branches, ognuno dedicato a un particolare componente del progetto (Diagrammi UML, GUI, Codice, Controller…). Per sviluppare un ottimo progetto software, bisogna tenere conto di tre importanti componenti:

* *Front-end*, trasforma il linguaggio programmatore in codice intermedio universale EM (lato Client);
* *Back-end,* traduce il codice EM in codice assembler per la tipologia di macchine a cui è destinato (deve adattarsi al sistema dell’ospedale), tipico del lato Server;
* *Documentazione,* dove si inserisce tutti i file descrittivi prodotti;

***GESTIONE DELLE MODIFICHE***

La baseline del progetto si trova nel branch “*main*”: per effettuare modifiche su una determinata parte del progetto, viene creato un nuovo branch, al fine di lavorare sullo specifico componente (codice, interfaccia grafica, diagrammi UML, documentazione…); ogni membro del gruppo può lavorare su tutti i branch creati e, per evitare incongruenze/collisioni tra i vari lavori, si possono creare più sotto-branch per risolvere problemi specifici.

Eventuali modifiche sui rami secondari possono essere trasferite su quello principale, attraverso l’uso di *pull-request*, le quali devono essere approvate dallo scrum master assegnato per tale componente del progetto (prima di effettuare la modifica, i membri del team devono confrontarsi per capire se tale modifica è necessaria o meno, ma spetta allo scrum master avere l’ultima parola sull’approvare o meno la nuova componente). Nelle fasi finali di elaborazione del progetto, si effettuano attività di *merge* (“unione”), per trasferire tutte le modifiche più importanti del progetto, situate nei vari branches, verso il main.

***ASSEGNAZIONE DEI COMPITI***

I compiti assegnati ai tre collaboratori vengono definiti in fase di configurazione, così come il ruolo di *scrum master*, che viene assegnato ad un singolo membro per ogni parte del Progetto: egli è incaricato di verificare che le modifiche proposte non abbassino il livello di qualità del sistema, ma deve anche mantenere rigore e disciplina nel lavoro di squadra; lo scrum master agisce da “leader”/”supervisore” per la propria parte definita, ma svolge anche il ruolo di “motivatore” per mantenere il team concentrato sugli obiettivi da completare.

Prima di iniziare con il lavoro vero e proprio, il *product owner* ha il compito di compilare una lista di desideri con priorità per i vari punti del progetto (*product backlog*); successivamente, all’inizio di ogni *sprint* (ognuno della durata di una settimana), lo *scrum master* deve scegliere un insieme di funzionalità (*sprint backlog*) da implementare e definire tale insieme come obiettivo corrente dello sprint. Il team, motivato dallo scrum master, deve essere in grado di completare i punti dello *sprint backlog* entro la conclusione dello sprint stesso: durante la settimana, i collaboratori si incontrano per valutare i progressi svolti nei giorni precedenti e assicurarsi che il lavoro proceda senza troppi intoppi (*weekly scrum*). L’obiettivo è andare a concludere, passo per passo, i vari punti del *product backlog*.

Per risolvere errori/malfunzionamenti/difetti, o per valutare nuove modifiche proposte, si aprono apposite *issue*, assegnate al membro designato, il quale ha il dovere di risolvere il problema e poi chiudere *l’issue*, indicandola come completata (in presenza di modifiche importanti si usano le *pull request*).