**Convenzioni per App Studenti Unimol**

Di seguito, sono raggruppate le convenzioni stabilite per le varie fasi del progetto, divise per software da utilizzare, struttura del template e terminologia del RAD e casi d’uso.

**Software da utilizzare:**

-Latex per il RAD.-Visual Paradigm Online per modellare i diagrammi.

-AdobeXd/Pencil per i mockups e di conseguenza gli Activity Diagrams.

**Struttura del RAD completo:**

-Raggruppare i requisiti in gruppi identificati da una sottosezione. Ad esempio:

<Titolo sottosezione>

Messaggi

-Visualizza messaggi: <desc>

-Invia messaggi: <desc>

**Terminologia e convezioni del documento RAD:**

-Ove si faccia riferimento all’app come descrizione nel documento, essa deve essere denominata e scritta in modo tale: “app *Studenti Unimol”*.

-Per quanto riguarda gli attori, l’utente finale è chiamato “*Studente*”, il sistema da interrogare per ricevere risposte è “*Esse3*”. L’altro servizio, necessario per la gestione delle aule, sarà indicato come “*Aule Unimol*”. Eventuali attori saranno indicati con lettera maiuscola e corsivo in tutto il documento, tranne che nei diagrammi dove è omessa la dicitura in corsivo per i nomi degli attori.

-Per tutti i nomi relativamente importanti, è previsto il corsivo e la lettera maiuscola. Allo stesso modo per le parole appartenenti ad altre lingue. Seguono *Esse3, Università degli Studi del Molise, storage, ecc.*

*-*Le sigle formalmente accettate nella documentazione sono quelle strettamente legate al contesto universitario: CFU, app, ecc.

-L’insieme dei corsi di un Corso di Laurea è indicato come “Piano di studio”. Ci si riferisce ai corsi frequentati con il termine “esami” solo se il contesto applicato è quello degli appelli e della sessione d’esame, altrimenti essi saranno etichettati come “corsi”.

-Utilizzare nomi consistenti e compatti tra requisiti, nome casi d’uso (che dev’essere costituito da azione + sostantivo: per esempio, Gestione appelli), descrizione, tabelle e diagrammi.

-Si usa “login” con l’articolo determinativo maschile “il”.

-I nomi comuni vanno tutti con le lettere minuscole, a meno che non si presentino all’inizio della frase.

-Si modellano come eccezioni solo quelle effettivamente inserite nelle tabelle dei casi d’uso.

-Gli activity diagrams verranno modellati usando gli screen mockups.

-Il sincronizzatore viene indicato con la parola *Sync****.***

**Caratteristiche casi d’uso**

* Per specificare gli attori, si usa la dicitura “Inizializzato da *X*. Partecipa *X*”.
* La precondizione è sempre “Lo studente ha effettuato il login”. Se non esistono altre differenti precondizioni, questa sarà posta all’inizio del relativo capitolo e omessa dalle tabelle per evitare la ridondanza.
* Evitare dettagli implementativi nelle azioni delle tabelle (ad esempio evitare le parole “clicca”, “fa un tap”, ecc.); sostituire con “Accede alla sezione”, “Seleziona”, “Apre” scegliendo la modalità più adeguata tra queste tre elencate.
* In tutte le tabelle in cui è presente *Esse3,* specificare che il sistema salva in locale una copia dei dati ricevuti, nella descrizione e nella tabella. Di conseguenza, gestire le eccezioni di connettività assente e copia presente in locale e connettività assente e copia non presente.
* Il sistema mostra un output, mentre l’utente visualizza qualcosa. Attenzione ai verbi da usare.
* Per ogni caso d’uso specificare l’eccezione “*Esse3* non risponde”, se tale attore è presente nel caso d’uso.
* Per ogni caso d’uso, quando *Esse3* compie l’azione di restituire i dati, allora la dicitura sarà: “Elabora la richiesta e invia i dati relativi a x (esempio: agli appelli)”. Stessa cosa per il successo dell’operazione, ad esempio “Conferma il successo dell’operazione”.
* Punto a fine frase, anche nelle tabelle.
* La condizione d’uscita è il risultato finale del caso d’uso. “Lo studente visualizza tutti i corsi del piano di studio”.
* Nel diagramma, indicare il sistema con App Studenti Unimol.

**Convenzioni Git**

**OBBLIGATORIO:**

- FARE ALMENO UN PULL AL GIORNO DALLA CARTELLA GENERALE DEL PROGETTO, SIA NEL PROPRIO BRANCH CHE NEL BRANCH DEVELOP PER RESTARE ALLINEATI.

- FARE SEMPRE UNA PULL PRIMA DI FARE UNA PUSH.

- Ogni gruppo ha un suo branch e lavora in quello.

- Pushare sempre nel proprio branch, poi fare una merge request di unione del proprio branch con il branch develop, solo quando il branch è in uno stato consistente e privo di errori. La merge request sarà approvata da un mantainer di develop (c’è n’è uno in ogni gruppo).

**BRANCHES TUTORIAL**

git branch -a -> mostra tutti i branch evidenziando quello in cui ci trova.

git checkout -b nomebranch -> crea un branch locale corrispondente a quello remoto (i branch remoti si chiamano origin/nomebranch, quelli locali solo nomebranch. All’atto della push, vengono mergiati, ma va fatto una tantum)

git chechout nomebranch -> switcha sul branch desiderato, quindi sul proprio branch o su develop.

**PER PUSHARE NEL PROPRIO BRANCH:**

git checkout miobranch

git pull

git add nomefile (aggiungere SOLO i file modificati, altrimenti associa il commit e relativo messaggio a tutti i file)

git commit -m “messaggio di commit” (sostituire un messaggio di commit che sia esplicativo delle modifiche effettuate)

git push

Solo quando il branch è in uno stato consistente e privo di errori, andare su GitLab. Cliccare sulla sezione “merge request” a sinistra, poi new merge request, scegliere come branch source il proprio branch e come branch target “develop”. Creare la merge request, aspettare che un manteiner la approvi.

**SVOLGERE LE OPERAZIONI**

- Aprire un terminale pulito, andare nella cartella del progetto allineato con il repository remoto con il comando cd + path, eseguire le operazioni nella cartella.

- Se si utilizza un IDE in cui si ha il progetto selezionato, aprire un terminale nel progetto (es. in WebStorm e PhpStorm View -> tool windows -> terminal) e dare i comandi.

- Se si utilizza un IDE come quelli della JetBrains in alto VCS, e poi PRIMA UPDATE PROJECT (pull) e POI commit. Update project aggiorna tutto il progetto allineandolo con il repository remoto, commit serve per fare commit and push. Commit: mostra i file modificati e - cliccandoci sopra - evidenzia un diff tra versione remota e versione locale. In basso a destra ha una freccia accanto a “commit” su cui c’è scritto “commit & push”. Dopo aver fatto tutto ciò si può andare su GitLab a fare la merge request se necessario.