

SEVEN CLICKERS

You Wish We Click

7clickersgroup@gmail.com

Norme di Progetto

Versione	0.1.1
Stato	-
Uso	Interno
Approvazione	-
Redazione	Mirko Stella Giacomo Mason Gabriele Mantoan Marco Brigo
Verifica	Giacomo Mason Tommaso Allegretti
Distribuzione	<i>Seven Clickers</i> Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione

Questo documento contiene le norme di progetto seguite dal gruppo Seven Clickers per il progetto ShowRoom3D

Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
0.1.1	03-12-22	Marco Brigo	Analista	Stesura punto 2.3,2.4,aggiornate norme su branching, pull request, milestone e project board, tolto punto su Jira _g
0.1.0	22-11-22	Giacomo Mason Tommaso Allegretti	Verificatori	Verifica documento
0.0.3	16-11-22	Gabriele Mantoan	Verificatore	Modificate norme riguardo verifica _g e vita delle issue _g ; aggiunte norme riguardo Jira _g e indici di glossario
0.0.2	13-11-22	Gabriele Mantoan	Verificatore	Aggiunte norme riguardo Issue tracking system _g , ruoli e nomi dei branch liberi
0.0.1	06-11-22	Mirko Stella Giacomo Mason	Analista Verificatore	Creazione documento

Indice

1	Introduzione	3
1.1	Scopo del documento	3
1.2	Scopo del prodotto	3
2	Documentazione	3
2.1	Convenzioni generali	3
2.1.1	Versionamento	3
2.1.2	Struttura Generale	4
2.1.3	Verbali	4
2.1.4	Date	4
2.1.5	Nomi di persona	5
2.2	Verifica	5
2.3	Approvazione	5
2.4	Strumenti per la stesura	5
3	Strumenti collaborativi	6
3.1	GitHub _g	6
3.1.1	Repository _g	6
3.1.2	Branching	6
3.1.3	Commits	6
3.1.4	Pull Requests	7
3.1.5	Milestone _g	8
3.1.6	Projects Board _g	8
3.1.7	Issue Tracking System _g	8
3.2	Glossario	8
4	Organizzazione del gruppo	9
4.1	Ruoli	9
4.1.1	Responsabile	9
4.1.2	Analista	9
4.1.3	Amministratore	9
4.1.4	Progettista	9
4.1.5	Programmatore	10
4.1.6	Verificatore	10

1 Introduzione

1.1 Scopo del documento

Lo scopo del documento è quello di stabilire le regole che ogni componente del gruppo SevenClickers deve rispettare per mantenere un ambiente di lavoro che mira a massimizzare l'economicità dei processi durante il ciclo di vita del prodotto ShowRoom3D.

Le norme verranno inserite in modo incrementale per regolamentare le attività di progetto imminenti rimandando quelle meno urgenti a quando se ne presenterà la necessità.

Inoltre tali norme potranno subire modifiche nel tempo in modo da garantire un miglioramento continuo della qualità del lavoro svolto. Il responsabile di progetto ha il compito di comunicare l'aggiunta di una nuova norma o la modifica di una già esistente a tutti i componenti del gruppo e di assicurarsi che siano comprese a pieno.

1.2 Scopo del prodotto

Il prodotto in questione nasce dalla necessità dell'azienda SanMarco Informatica di fornire una soluzione agli sprechi derivati dall'adozione di uno ShowRoom tradizionale proponendo uno ShowRoom3D che sia ugualmente o ancora più immersivo.

2 Documentazione

Questa sezione descrive le convenzioni, gli strumenti e le modalità con cui il gruppo si impegna a stilare la documentazione interna ed esterna relativa al progetto.

2.1 Convenzioni generali

Le convenzioni di seguito riportate vengono applicate a tutti i documenti. Esse rendono i documenti stilati omogenei tra loro contribuendo a rendere il progetto professionale.

2.1.1 Versionamento

Il numero di versione permette di capire lo stato in cui si trova un documento. Un documento può trovarsi nei seguenti stati:

- **Approvato:** Il documento è verificato ed approvato dal Responsabile di progetto
- **Verificato:** Il documento risulta verificato ma non ancora visionato dal Responsabile di progetto
- **In Sviluppo:** Sono presenti delle modifiche che non sono state verificate

Il numero di versione ha il formato **X.Y.Z** dove:

- **X** indica una versione approvata dal Responsabile di progetto, la numerazione parte da 0 e la prima versione approvata è la 1.0.0
- **Y** indica una versione verificata dal Verificatore, la numerazione inizia da 0 e si azzerà ad ogni incremento di X. La prima versione verificata è la 0.1.0
- **Z** indica una versione in fase di modifica da parte dei redattori che ne incrementano il numero ad ogni modifica, la numerazione parte da 1 e si azzerà ad ogni incremento di X o Y. La prima versione modificata è la 0.0.1

2.1.2 Struttura Generale

Ogni documento deve presentare le seguenti sezioni nell'ordine in cui vengono presentate:

- **Intestazione:** Contiene:
 - Logo compreso di motto
 - Indirizzo email di gruppo
 - Titolo
 - Tabella contenente le informazioni generali
 - * Versione
 - * Stato
 - * Uso
 - * Approvazione: indica il responsabile di progetto che ha approvato il documento
 - * Redazione: elenco dei collaboratori che hanno partecipato alla stesura del documento
 - * Verifica: elenco dei verificatori che hanno verificato il documento
 - * Distribuzione: elenco delle persone o organizzazioni a cui è destinato il documento
 - Breve descrizione del documento
- **Registro delle modifiche:** Tabella che identifica ogni versione del documento indicandone:
 - Versione
 - Data
 - Autore
 - Ruolo
 - Descrizione
- **Indice:** Elenco ordinato dei titoli dei capitoli, ovvero delle varie parti di cui si compone il documento.
- **Contenuto:** Varia a seconda del tipo di documento.

2.1.3 Verballi

Rispettano tutta la struttura generale. In aggiunta presentano:

- **Informazioni Generali** Contengono:
 - Luogo
 - Data
 - Ora
 - Partecipanti
- **Tabella tracciamento temi affrontati:** tabella che riassume i punti salienti della riunione indicandone
 - Codice: ha il formato **VX Y.Z** dove X indica la tipologia di verbale, Y indica il numero di verbale (incrementale rispetto agli altri verballi) e Z indica il numero dell'argomento trattato (incrementale rispetto agli altri argomenti del verbale)
 - Descrizione: breve descrizione di uno specifico argomento trattato

2.1.4 Date

Le date devono rispettare il seguente formato: **dd-mm-yyyy** All'interno delle tabelle il formato deve essere il seguente: **dd-mm-yy**

2.1.5 Nomi di persona

All'interno dei documenti i nomi di persona rispetteranno l'ordine nome seguito dal cognome della persona menzionata.

2.2 Verifica

La verifica viene svolta da due verificatori prima del merge con il branch `documentation`. Consiste nell'esaminare i file prodotti da chi ne ha fatto la stesura e segnalarne la non validità o la presenza di errori nei concetti esposti.

Un verificatore dovrà verificare il documento a partire dalle modifiche fatte dopo l'ultima versione verificata. Le modifiche da verificare quindi possono essere dedotte dal registro dei cambiamenti presente in ogni documento. Una volta controllato il documento, il primo verificatore segnalerà eventuali errori e successivamente dovrà spuntare come approvata la Pull Request nella sezione dedicata su GitHub.

A questo punto se un secondo verificatore noterà la necessità di qualche altro cambiamento da apportare, chi dovrà apportare le modifiche farà una pull in locale per allineare il proprio branch con quello in remoto e continuare con il proprio lavoro.

Dopo il push delle modifiche se i file risultano corretti anche dal secondo verificatore esso aggiungerà i nomi dei verificatori all'intestazione modificando il file `titlepage.input.tex` e creerà la riga nel registro delle modifiche, nel file `changelog.input.tex`, inserendo la nuova versione secondo le regole di versionamento e scrivendo nella colonna Autore sia il suo nome, che quello del primo verificatore, e in descrizione "Verifica". Dopo aver eseguito un commit sul ramo da integrare approva la Pull Request e conferma il merge secondo le norme di progetto descritte nella sezione dedicata.

2.3 Approvazione

L'approvazione viene svolta dal Responsabile di Progetto.

Si effettua questa pratica non appena il file prodotto viene verificato e passa tutte le eventuali reviews create dai verificatori e non necessita di ulteriori modifiche o revisioni future. Spetterà quindi l'ultima parola al Responsabile di Progetto che dovrà approvare la versione del file. In caso si riscontrassero ulteriori problematiche, verrà riportata la pull request nella fase di verifica citata precedentemente nel punto 2.2.

Ad un esito positivo di questa fase di approvazione, verrà aggiunto il nome di chi ha approvato il file nell'intestazione prodotta dal file `titlepage.input.tex`, creando una nuova riga nel registro delle modifiche, nel file `changelog.input.tex`, inserendo l'ultima versione secondo le regole di versionamento e scrivendo nelle rispettive colonne il suo nome e la descrizione "Approvazione".

Verrà poi chiusa la pull request da parte del Responsabile di Progetto e l'issue associata di approvazione del file, come spiegato dalla sezione dedicata, si chiuderà automaticamente.

2.4 Strumenti per la stesura

- LaTeX: è un linguaggio di marcatura per la preparazione di testi, basato sul programma di composizione tipografica TEX.

I nostri documenti sono scritti utilizzando solamente questo linguaggio. Nel branch `documentation` si possono trovare i file .pdf prodotti e la cartella "latex" che conterrà a sua volta tre cartelle interne:

- esterni e interni, contengono file .tex di documentazione esterna ed interna come ad esempio i verbali o altra documentazione esterna/interna
 - * Ogni documento prodotto contiene:
 - la cartella `config`, dove troviamo i file di template .tex riscritti in base al documento che si sta producendo
 - la cartella `res/sections`, dove troviamo i file .tex delle sezioni del documento
 - un file col nome del documento pdf con estensione .tex in cui raggruppa tutti gli altri file .tex prodotti nelle due cartelle sopracitate

- template, contiene file .tex di base utilizzati in ogni documento prodotto
 - * changelog.tex è il file di template che serve per scrivere il registro delle modifiche
 - * package.tex è il file che contiene tutti gli usepackage_g
 - * titlepage.tex è il file di template che contiene la configurazione della pagina iniziale di ogni documento
 - * tracking.tex è il file di template che contiene il tracciamento dei temi affrontati nel documento

3 Strumenti collaborativi

3.1 GitHub_g

Servizio di hosting per progetti software che implementa uno strumento di controllo versione distribuito Git_g. Oltre alla copia in remoto del repository_g di progetto ogni componente del gruppo ha una propria copia in locale.

Per ottenere una copia del repository_g ogni componente ha scaricato lo strumento Git_g ed eseguendo il comando 'git clone' da git_g bash viene creata una cartella collegata alla repository_g di progetto.

Non sono state imposte modalità specifiche sull'interazione con il repository_g remoto in modo da non sconvolgere le abitudini di lavoro di ciascun componente.

I componenti del gruppo abituati ad interagire con GitHub_g da interfaccia grafica possono continuare a farne uso.

3.1.1 Repository_g

Il repository_g si può trovare all'indirizzo <https://github.com/7clickers/ShowRoom3D> ed è pubblico. I collaboratori sono i componenti del gruppo SevenClickers che utilizzano il proprio account GitHub_g personale per collaborare al progetto.

3.1.2 Branching

Branches protetti:

- main: contiene le versioni di release_g del software
- documentation: contiene i template latex_g e rispettivi pdf della documentazione

documentation: I documenti presenti in documentation sono stati approvati dal Responsabile di progetto o almeno verificati dai verificatori.

Se si devono effettuare delle modifiche a branches protetti GitHub permette la creazione di un ramo di appoggio per la risoluzione dei conflitti. Successivamente, se non ci sono conflitti, verrà effettuato il merge e verrà eliminato il branch di appoggio.

Branches liberi: Vengono utilizzati per creare nuove funzionalità e gli sviluppatori possono effettuare i commit senza l'approvazione degli altri componenti del gruppo in quanto ciascun componente sviluppa su un solo branch alla volta salvo casi eccezionali. Un branch_g libero avrà il nome del documento che si sviluppa su quel branch_g, oppure della feature_g che va ad implementare.

Non appena i/il file nel branch sono stati verificati ed il merge è stato fatto, il branch libero verrà eliminato.

3.1.3 Commits

È preferibile che ogni commit abbia una singola responsabilità per cambiamento.

I commits non possono essere effettuati direttamente sui branch protetti ma per integrare delle aggiunte o modifiche sarà necessario aprire una Pull Request. All'approvazione di una Pull Request tutti i commit relativi al merge verranno raggruppati in un unico commit che rispetti la struttura sintattica descritta in seguito.

I commit dovranno essere accompagnati da una descrizione solo se ritenuta indispensabile alla comprensione del commit stesso.

I messaggi di commit sui **BRANCH PROTETTI** dovranno seguire la seguente struttura sintattica:

`<label><#n_issue><testo>`

dove:

label: può assumere i seguenti valori

- feat: indica che è stata implementata una nuova funzionalità
- fix: indica che è stato risolto un bug
- update: indica che è stata apportata una modifica che non sia fix o feat
- test: qualsiasi cosa legata ai test
- docs: qualsiasi cosa legata alla documentazione

n_issue: indica il numero della issue a cui fa riferimento il commit (se non fa riferimento a nessuna issue viene omesso).

testo: indica con quale branch è stato effettuato il merge e deve rispettare la forma: merge from <nome branch da integrare> to <nome branch corrente>

descrizione: se aggiunta ad un commit deve rispondere alle domande WHAT?, WHY?, HOW? ovvero cosa è cambiato, perchè sono stati fatti i cambiamenti, in che modo sono stati fatti i cambiamenti.

I messaggi di commit sui **BRANCH LIBERI** dovranno seguire la struttura sintattica dei branch protetti ad eccezione del testo. Il testo dei commit sui branch liberi non è soggetto a restrizioni particolari a patto che indichi in maniera intuitiva i cambiamenti fatti in modo che possano essere compresi anche dagli altri collaboratori.

3.1.4 Pull Requests

Per effettuare un merge su un branch protetto si deve aprire da GitHub_g una Pull Request. La Pull Request permette di verificare il lavoro svolto prima di integrarlo con il branch desiderato.

Alla creazione di una Pull Request bisogna associare:

- i verificatori in carica hanno il compito di trovare eventuali errori o mancanze e fornire un feedback riguardante il contenuto direttamente su GitHub_g richiedendo una review con un review comment sulla parte specifica da revisionare o con un commento generico.
Non sarà possibile effettuare il merge finchè tutti i commenti di revisione non saranno stati risolti. Il merge potrà avvenire dopo aver risolto tutti i commenti di revisione ed approvato da almeno due verificatori.
- l'issue associata nell'opzione "Development" che verrà automaticamente risolta alla risoluzione della Pull Request
- la Projects Board di cui fa parte

Per i commit relativi alle Pull Requests seguire le regole descritte nella sottosezione Commits per i branch protetti.

3.1.5 Milestone_g

Ogni milestone corrisponde ad un aggiornamento di un macroargomento o la creazione di file molto importanti, come ad esempio: norme di progetto, piano di progetto, PoC. Ogni issue inerente a dei determinati macroargomenti verranno associate ad una milestone_g. Verranno inoltre predisposte delle milestone_g con una scadenza temporale a cui potranno essere associate le issue. La scadenza delle milestone_g è impostata in base al carico di lavoro da svolgere o alla rilevanza del file da produrre o aggiornare.

I verbali non verranno associati ad alcuna milestone in quanto non rilevanti ai fini di avanzamento di documenti con ben più importanza.

3.1.6 Projects Board_g

Le issue verranno tracciate nella **projectboard_g** della repo, che è visibile e modificabile da tutti i membri del gruppo. La projectboard_g è suddivisa in queste sezioni:

- **Todo:** Issue_g che non sono ancora state iniziate o che non sono ancora state assegnate
- **In Progress:** Issue_g che sono state assegnate e a cui almeno un membro a cui è stata assegnata ha iniziato a lavorarci
- **Pull Request:** Issue_g che è in fase di integrazione e necessita della verifica dei verificatori. Corrisponde all'inizio di una pull request
- **Done:** Issue_g che sono state chiuse e che sono state verificate (se necessitano di verifica_g)
- **Approved:** Issue_g col tag "approvable", e che hanno ottenuto l'approvazione_g del responsabile

Inoltre nella project board vengono registrate delle issue che non richiedono verifica, approvazione o neanche integrazione, con lo scopo di monitorare meglio il lavoro di ogni membro del team.

Queste issue verranno chiuse e archiviate manualmente una volta che avranno terminato la loro utilità, un esempio può essere la seguente issue:

diario di bordo 21-11-22; questa issue non necessita verifica, approvazione o integrazione perchè non è di interesse caricare il file nella repo, però è utile tracciare lo svolgimento della issue.

3.1.7 Issue Tracking System_g

Il gruppo utilizza l'issue tracking system_g di Github_g per tenere traccia delle issue_g. Le issue verranno determinate dal responsabile, ma la loro assegnazione verrà effettuata dai membri del gruppo, in base alle loro preferenze, i loro ruoli e alle loro disponibilità temporali.

Nel caso un membro del gruppo dovesse rendersi conto che l'issue_g che sta svolgendo potrebbe essere suddiviso in ulteriori issue_g, dovrà rivolgersi al responsabile, che è l'unico che può aggiungere, modificare o eliminare le issue_g.

Ogni issue ha almeno un tag associato per specificarne lo scopo, inoltre il gruppo ha aggiunto il seguente tag a quelli già forniti da Github_g:

- **approvable:** indica una issue_g che dopo esser stata verificata_g, deve anche essere approvata_g dal responsabile per andare in fase di release_g
- **tag di priorità (alta/media/bassa):** descrivono quanto è urgente il completamento dell'issue_g

3.2 Glossario

All'interno del documento si possono trovare dei termini che possono risultare ambigui a seconda del contesto, o non conosciuti dagli utilizzatori.

Per ovviare ad errori di incomprensione che possono portare a problemi di vario genere e rallentamenti si è deciso di stilare un elenco di termini di interesse accompagnati da una descrizione dettagliata del loro significato.

I termini presenti all'interno del glossario vengono indicati con il pedice 'g' come nell'esempio seguente:

termine_g. Il glossario ordina i termini in ordine alfabetico in modo da permetterne una facile e veloce ricerca. Ogni componente del gruppo all'inserimento di un termine ritenuto ambiguo deve preoccuparsi di aggiornare il glossario in modo da mantenerlo sempre aggiornato.

4 Organizzazione del gruppo

4.1 Ruoli

I componenti del gruppo si suddivideranno nei seguenti ruoli per periodi di circa 2-3 settimane (dipendentemente dalle esigenze del periodo) e al termine del periodo i ruoli verranno risuddivisi. Visto che nelle varie fasi di sviluppo del progetto le attività da svolgere variano, non sempre sarà necessario coprire tutti i ruoli.

Inoltre sarà necessario tenere traccia delle ore che ogni componente dedica al progetto ed il ruolo associato a quelle ore, in modo da andare a rispettare la tabella degli impegni individuali.

I ruoli e le loro competenze sono i seguenti:

4.1.1 Responsabile

Deve avere la visione d'insieme del progetto e coordinare i membri, inoltre si occupa di rappresentare il gruppo con le interazioni esterne (proponente, committente ecc...). Le sue competenze specifiche sono:

- ad ogni iterazione_g c'è un solo responsabile
- presentare il diario di bordo in aula
- redarre l'ordine del giorno prima di ogni meeting interno del gruppo
- suddivide le attività del gruppo in singole issue (ma non le assegna ai membri del gruppo)
- in fase di release_g si occupa di approvare_g tutti i documenti che necessitano approvazione

4.1.2 Analista

Si occupa di trasformare i bisogni del proponente nelle aspettative che il gruppo deve soddisfare per sviluppare un prodotto professionale. Le sue competenze specifiche sono:

- interrogare il proponente riguardo allo scopo del prodotto e le funzionalità che deve avere
- studiare le risposte del proponente per identificare i requisiti_g e redarre l'analisi dei requisiti

4.1.3 Amministratore

Si occupa del funzionamento, mantenimento e sviluppo degli strumenti tecnologici usati dal gruppo. Le sue competenze specifiche sono:

- ad ogni iterazione_g basta un solo amministratore
- gestione delle segnalazioni e problemi dei membri del gruppo riguardanti problemi e malfunzionamenti con gli strumenti tecnologici
- valuta l'utilizzo di nuove tecnologie e ne fa uno studio preliminare per poter presentare al gruppo i pro e i contro del suo utilizzo

4.1.4 Progettista

Si occupa di scegliere la modalità migliore per soddisfare le aspettative del committente che gli analisti hanno ricavato dall'analisi dei requisiti_g. Le sue competenze specifiche sono:

- scegliere eventuali pattern architetturali da implementare
- sviluppare lo schema UML_g delle classi_g

4.1.5 Programmatore

Si occupa di implementare le scelte e i modelli fatti dal progettista. Le sue competenze specifiche sono:

- scrivere il codice atto a implementare lo schema delle classi
- scrivere eventuali test
- scrivere la documentazione per la comprensione del codice che scrive

4.1.6 Verificatore

Si occupa di controllare che ogni file che viene caricato in un branch protetto della repository_g sia conforme alle norme di progetto. Le sue competenze specifiche sono:

- controllare i file modificati o aggiunti durante una pull request tra un ramo non protetto e un ramo protetto siano conformi alle norme di progetto e cercano errori di altra natura (ortografici, sintattici, logici, build ecc...).