Winning Software Solution
winningsoftwaresolution@gmail.com

ShopChain
SyncLab

Norme di progetto

Informazioni	
Redattori	Elia Scandaletti Raffaele Oliviero Giovanni Cocco Alberto Nicoletti
Versione	2.0.0
Uso	interno

Descrizione

Questo documento contiene le procedure, gli strumenti e i criteri di qualità che verranno usati nel progetto.

Versione	Data	Persona	Attivtà	Descrizione
2.0.0	20/03/2022	Alberto Nicoletti	Redazione	Rifacimento documento completo
0.0.9	16/02/2022	Giovanni Cocco	Redazione	Stesura di norme software
0.0.8	08/02/2022	Raffaele Oliviero	Redazione	Stesura di: analisi dei requisiti, analisi delle tecnologie e piano di progetto
0.0.7	04/02/2022	Giovanni Cocco	Redazione	Stesure delle revisioni di avanzamento interne
0.0.6	16/01/2022	Elia Scandaletti	Redazione	Adeguamento di verifica e validazione secondo decisioni collettive
0.0.5	09/01/2022	Raffaele Oliviero	Redazione	Stesura struttura del piano di qualifica e della verifica e validazione
0.0.4	09/01/2022	Raffaele Oliviero	Redazione	Ristrutturazione documento
0.0.3	22/12/2021	Elia Scandaletti	Redazione	Stesura regole stesura verbali
0.0.2	07/12/2021	Elia Scandaletti	Redazione	Inserimento norme da verbale del 01/12/2021
0.0.1	14/11/2021	Elia Scandaletti	Redazione	Stesura configurazione Github
0.0.0	13/11/2021	Elia Scandaletti	Redazione	Stesura documentazione

Contents

1 Introduzione

1.1 Scopo del documento

Lo scopo del presente documento è definire procedure, strumenti e criteri di qualità al fine di stabilire un Way of Working. Ogni membro del gruppo sarà tenuto a rispettare le indicazioni qui presentate.

2 Processi Primari

2.1 Fornitura

Il processo di fornitura determina le attività e le risorse che servo per gestire e assicurare il progetto. In qusta sezione vengono stilate le norme per la gestione dei rapporti con il proponente ed il committente.

2.1.1 Repository pubblica

Durante tutto il progetto verrà usata un unica repository pubblica. La parte di repository che interessa al proponente ed al committente è la cartella *public* definita come parte pubblica della repository. Questa cartella contiene un file read-me in cui viene spiegato ed elencato il contenuto della parte pubblica con un avviso che chiede di restarne all'interno, ignorando il resto della repository, detta parte privata.

La parte pubblica contiene i documenti necessari alle revisioni di periodo con il committente.
Ogni file nella parte pubblica deve indicare la versione nel nome del file stesso nel seguente modo: <NomeFile>_v<versione>.pdf.
Ogni modifica alla parte pubblica deve essere approvata dal responsabile, cui spetta accettare le pull request della parte pubblica.

2.1.2 Contatti con proponente e committente

Tutti i contatti con il proponente ed il committente avvengono tramite la mail del gruppo *winnningsoftwaresolution@gmail.com*.

L'invio di una mail nasce dalla richiesta di uno o più componenti del gruppo e necessita di una discussione con il responsabile e della sua approvazione.

Le mail per la presentazione ad una revisione di periodo necessita della conferma di tutti i componenti, che valutano l'effettiva preparazione del gruppo per la revisione. Questa mail deve contenere la lettera di presentazione con i link ai vari documenti nella parte pubblica della repository.

I colloqui devono essere richiesti tramite mail. La reale necessita di un colloquio va discussa durante la riunione settimanale.

I colloqui con il proponente si svolgono su Meet o su Discord. I colloqui con il committente si svolgono su Zoom.

2.2 Sviluppo

2.2.1 Gestione

Gestione tempistiche

Lo sviluppo del prodotto va da mercoledì (corrispondente all'inizio del ciclo di Sprint) al sabato estremi inclusi, i restanti 3 giorni vengono spesi per effettuare la verifica sul lavoro svolto.

2.2.2 Analisi dei requisiti

Scopo

Fornire un'analisi dettagliata del capitolato proposto al fine di trovare i requisiti che coprano interamente le esigenze della proponente.

Fornire ai verificatori tutti i requisiti con per poter effettuare la validazione del sistema e delle sue unità.

Descrizione

Fonti da cui rilevare le informazioni:

- Capitolato
- Verbali dei meeting con la proponente
- Verbali dei meeting interni

Struttura

- Descrizione del documento
- Casi d'uso
  - Utilizzare il programma *StarUML* per il disegno dei diagrammi.
  - Esportare i diagrammi in formato *SVG*.
  - Numerare ogni caso d'uso come *UCX*.
    Ogni generalizzazione di X viene numerata come *UCX.x* .
    Dove X e x sono numeri interi positivi >= 1, progressivi.
    E' buona norma seguire una numerazione progressiva in base all'ordine in cui sono disposti i diagrammi, sebbene nel tempo possono essere eliminati/aggiunti casi d'uso.
    Per evitare di dover rinumerare tutti i diagrammi:
    Se si aggiungono casi d'uso, partire dalla numerazione del caso d'uso con indice X maggiore.
    Per l'aggiunta di generalizzazioni, le regole sono analoghe per l'indice x.
  - Ogni diagramma deve essere identificato dalla stringa *Figura N*.
    Dove N è un numero interi positivo >= 1, progressivo.
- Requisiti
  - Ogni requisito è identificato da un codice univoco composto da: R(per requisito)+F/Q/V(per la tipologia: funzionale, qualità, vincolo)+O/D/F(per la rilevanza: obbligatorio, desiderabile, facoltativo)+x(un numero univoco a due cifre).

Modifiche al documento

Prima di apportare qualsiasi modifica significativa funzionale al documento (e.g. aggiunta/rimozione casi d'uso, requisiti...) bisogna contattare il proponente e attendere conferma di accettazione delle modifiche che si vogliono apportare al documento.

2.2.3 Progettazione

Scopo

La progettazione ha compito di definire tutte le caratteristiche che il prodotto dovrà avere per soddisfare al meglio tutti i requisiti degli stakeholders.

Descrizione

Tramite questa attività si andrà a fissare l'architettura del prodotto.

Specifica Architetturale

- Gli elementi devono essere il più semplici possibile, contenendo solo il necessario e nulla di superfluo;
- Gli elementi non devono essere accessibili dall'esterno se non strettamente necessario;
- Gli elementi distinti devono essere il meno accoppiati possibile;
- Gli elementi parte delle stesse funzionalità/moduli devono essere il più coesi possibile.

Design Pattern

Implementare codice tramite l'uso di design pattern solo ove il costo di un'architettura simile sia inferiore ai benefici.

Diagrammi UML

- **Diagramma delle classi**: consentono di descrivere tipi di entità, con le loro caratteristiche e le relazioni tra esse. Si astraggono da un linguaggio di programmazione specifico, permettendo di non dipendere da esso nella rappresentazione.
  Riferimento diagramma delle classi
- **Diagramma di sequenza**: modellano le iterazioni tra gli oggetti in un unico caso d'uso. Illustrano come le diverse parti di un sistema interagiscono per lo svolgimento di un compito e l'ordine in cui queste interazioni avvengono durante l'esecuzione di un specifico caso d'uso.
  Riferimento diagramma di sequenza

2.2.4 Organizzazione dei file

Tutta la documentazione sarà contenuta all'interno della cartella `docs/`. La cartella è a sua volta divisa in due sottocartelle `docs/interni/` e `docs/esterni/`, che conterranno rispettivamente la documentazione interna al gruppo e quella da condividere con gli altri stakeholders.

I verbali interni saranno conservati nella sottocartella `interni/verbali/`. I verbali esterni saranno conservati nella sottocartella `esterni/verbali/`.

Per ogni documento sarà presente una cartella che sarà chiamata <`nome`>`/`, dove <`nome`> è il nome del documento.

Il file principale di ogni documento sarà all'interno della rispettiva cartella e sarà chiamato <`nome`>`.tex`, dove <`nome`> è il nome del file.

Le specifiche di nomenclature sono descritte in 3.1.4.

Eventuali altri file utilizzati nella stesura del documento saranno posizionati all'interno della stessa cartella.

2.2.5 Workflow Github

Ogni modifica alla repository dovrà seguire i seguenti passaggi:

- Modifica del codice su repository locale;
- Push su un branch remoto;
- Pull request tramite piattaforma web GitHub per merge sul branch `main`;
- Review della pull request;
- Merge sul branch `main`.

Un branch remoto è un branch su cui operano uno o più membri del gruppo e ha come scopo l'elaborazione di un prodotto o di un processo di progetto. Il branch si chiamerà come il prodotto o il processo relativo.

Le modifiche su un branch remoto avverranno tramite merge di un branch locale o con commit diretto. I membri del gruppo che lavorano sul branch concordano una modalità.

È auspicabile che i branch locali non compaiano nella repository remota.

Prima di effettuare un push alla repository remota o un merge è necessario fare uno squash dei commit superflui.

Non appena chi lavora su un branch remoto ritenga che ci sia materiale pronto per una revisione, apre una pull request su GitHub.

La review dovrà essere fatta dal verificatore. Il verificatore potrà accettare la pull request.

Una pull request può essere accettata solo se tutti i prodotti coinvolti superano i requisiti minimi definiti nel *Piano di Qualifica*.

Quando il verificatore accetta un pull request, dovrà eliminare il branch relativo a quest'ultima.

Non è consentito fare il caricamento diretto di file tramite interfaccia web o desktop.

Non è consentito forzare i push sulla repository remota.

2.2.6 Codifica

Scopo

La codifica ha il compito di trasformare l'architettura del prodotto in codice eseguibile dai calcolatori.

I programmatori trasformano quindi il lavoro fatto dai progettisti in codice, basandosi puramente sulle scelte architetturali.

Descrizione

La scrittura del codice deve rispetare le norme elencate di seguito

Convenzioni linguistiche

- Il codice deve essere scritto in lingua inglese (nomi di variabili, classi, commenti ecc..)

Convenzioni nomenclatura file

- Nel caso il file contenga una classe, il nome del file deve essere: *NomeClasse.EstensioneFile.*
- Nel caso il file non descriva una classe dare un nome significativo al file che ne descriva il compito (e.i. *index.html* per la pagina principale di un sito).

Convenzioni documentazione

- Se il compito di un metodo è particolarmente complesso e difficile da dedurre, aggiungere un commento per esplicitare il suo funzionamento.
- Indicare all'interno di un commento la stringa *TODO:* seguita dal lavoro/correzione da fare su quel blocco di codice in futuro.

Convenzioni stile di codifica

- Utilizzare la notazione *PascalCase* per nominare varibili, funzioni, classi.
- Se il linguaggio prevede l'utilizzo delle classi, suddividere ogni classe in un file separato.
- Se un istruzione condizionale(e.g. if-else) o un istruzione di ciclo(e.g. for, while) prevede un blocco, la parentesi di apertura del blocco deve essere in linea con l'istruzione associata.
- Indentare il codice tramite l'utilizzo di 4 spazi.

Server/TypeScript

- In caso di istruzione asincrona, deve essere inclusa all'interno di una *Promise*.

HTML

- Utilizzare tutte le buone prassi per rendere il sito accessibile, seguendo le linee guida WCAG 2.1

CSS

- Mantenere un solo file chiamato *style.css* per la versione desktop
- Mantenere un solo file chiamato *mini.css* per la versione mobile
- Specificare le unità di misura in *em* o in *%*

3 Processi di supporto

3.1 Documentazione

3.1.1 Ciclo di vita

Ogni documento passa per le seguenti fasi di vita:

- **Pianificazione:** il documento viene discusso e delineato per sommi capi sulla base delle necessità a cui deve rispondere;
- **Redazione:** un membro del gruppo, nel ruolo di redattore, scrive fisicamente il documento;
- **Revisione:** un membro del gruppo, nel ruolo di revisore, controlla che non siano presenti errori grammaticali e che sia aderente alle norme di progetto;
- **Approvazione:** un membro del gruppo, nel ruolo di responsabile, verifica che il contenuto del documento sia corretto e coerente con lo scopo dello stesso.

Non è possibile per un membro del gruppo coprire più ruoli nella gestione dello stesso documento.

Una volta che un documento è revisionato o approvato può comunque tornare in fase di stesura sulla base delle esigenze del momento.

3.1.2 Strumenti Utilizzati

Per la stesure dei documenti verrà utilizzato il linguaggio LaTeX.

Ogni documento dovrà includere il file `docs/template/style.tex` che contiene indicazioni di stile.

Ogni documento userà per la prima pagina il template `docs/template/front_page.tex`.

Eventuali risorse che dovessero servire per il template, saranno nella stessa cartella.

3.1.3 Struttura

Ogni documento avrà:

- Una pagina di intestazione;
- Una pagina di elenco delle versioni, dove ammesso;
- Una pagina di indice, dove ammesso;
- Il contenuto.

Le sezioni avranno le seguenti caratteristiche:

Pagina di intestazione

La pagina di intestazione contiene:

- Logo e nome del gruppo;
- Nome del progetto e azienda cliente;
- Contatto;
- Titolo del documento;
- Informazioni sul documento:
  - Elenco dei redattori;
  - Elenco dei revisori;

  - Elenco dei responsabili;
  - Versione, laddove prevista;
  - Destinazione d'uso.
- Breve descrizione.

La destinazione d'uso può essere "interno" o "esterno".

Elenco delle versioni

L'elenco delle versioni è una tabella con le seguenti colonne:

- Versione;
- Data;
- Persona;
- Attività;
- Descrizione.

Con persona si intende il membro del gruppo che svolge l'attività.
Le righe della tabella sono in ordine cronologico inverso.

Indice

L'indice è creato usando il comando `\tableofcontents`.

Norme di progetto

Scopo Lo scopo delle norme di progetto è definire procedure, strumenti e criteri di qualità al fine di stabilire un Way of Working. Ogni membro del gruppo sarà tenuto a rispettare le indicazioni lì presentate.

Titolo Il titolo del documento è "Norme di progetto".

Nome del file Il file sarà chiamato `norme_di_progetto.tex`.

Verbali

Scopo Lo scopo dei verbali è tenere traccia del dialogo interno ed esterno al gruppo e delle eventuali decisioni prese.

Titolo Ogni verbale sarà titolato "Verbale del <data>[con <esterni>]", dove <data> indica la data dell'incontro nel formato europeo. In caso all'incontro siano presenti persone esterne al gruppo, verrà usata anche la dicitura "con <esterni>", dove <esterni> indica i partecipanti alla riunione separati da virgole. Gli esterni possono essere identificati collettivamente tramite il nome dell'azienda o ente che rappresentano, tramite nome e cognome o tramite titolo e cognome.

Esempi:

- "Verbale del 09/11/2021";
- "Verbale del 28/10/2021 con SyncLab";
- "Verbale del 25/12/2021 con prof. Cardin".

Nome dei file I file saranno chiamati <data>_<tipo>.tex, dove <data> è la data dell'incontro in formato <yyyy>_<mm>_<dd> e <tipo> è I se non sono presenti persone esterne al gruppo, E altrimenti.

Indice Non è presente una pagina con l'indice.

Struttura Ogni verbale deve riportare:

- Data, ora e durata;
- Luogo;
- Partecipanti;
- Ordine del giorno;
- Riassunto dei contenuti;
- Eventuali impegni assunti.

Stesura All'inizio di ogni riunione verrà nominato un partecipante che redigerà il verbale e uno che lo approverà.

Chi redige il verbale ha 24 ore per completare il lavoro. Chi approva il verbale ha 24 ore per approvarlo o segnalare eventuali correzioni da apportare.

Nel secondo caso, il redattore avrà ulteriori 24 ore per adeguarsi alle indicazioni.

Piano di Qualifica

Scopo Lo scopo del *Piano di Qualifica* è definire le metriche e i requisiti minimi di qualità per l'approvazione dei prodotti del progetto.

Titolo Il titolo del documento è "Piano di Qualifica".

Nome del file Il file sarà chiamato `piano_di_qualifica.tex`.

Piano di Progetto

Scopo Lo scopo del *Piano di Progetto* è definire gli obiettivi da raggiungere, stabilendo le tempistiche e chi se ne occupa e tiene traccia del loro progresso, valutando i costi sostenuti rispetto a quelli preventivati.

Titolo Il titolo del documento è "Piano di Progetto".

Nome del file Il file sarà chiamato `piano_di_progetto.tex`.

Analisi dei requisiti

Scopo Lo scopo dell'*Analisi dei requisiti* è raccogliere i risultati dell'attività di analisi dei requisiti. Esso contiene quindi la descrizione dei casi d'uso del prodotto software da sviluppare, ed i requisiti suddivisi per tipologia.

Titolo Il titolo del documento è "Analisi dei requisiti".

Nome del file Il file sarà chiamato `analisi_dei_requisiti.tex`.

Analisi delle tecnologie

Scopo Lo scopo dell'*Analisi delle tecnologie* è valutare le tecnologie disponibili alla realizzazione del progetto. Esso contiene quindi le metriche di valutazione e l'analisi delle tecnologie in base a tali metriche.

Titolo Il titolo del documento è "Analisi delle tecnologie".

Nome del file Il file sarà chiamato `analisi_delle_tecnologie.tex`.

3.1.4 Convenzioni utilizzate

Riferimenti a file e cartelle

Per indicare il nome di un file o di una cartella si utilizza il `testo monospaziato`.
Il nome di una cartella termina sempre con `/`.

Stringhe e nomi

Le stringhe vengono scritte tra doppi apici " ".
Per indicare un parametro in un nome o una stringa si usa del testo tra parentesi angolari < >. Il parametro non può contenere spazi.
Per indicare una parte di nome o stringa opzionale si usa del testo tra parentesi quadre [ ].

Riferimenti tra documenti

Per riferirsi a sezioni all'interno dello stesso documento si usa il comando `\ref`.
Per riferirsi a sezioni di un altro documento si usa il nome di quella sezione scritto in corsivo.

3.2 Versionamento

3.2.1 Repository GitHub

Per il versionamento del progetto si è scelto di utilizzare Git su piattaforma GitHub.
La repository `https://github.com/iota97/WinningSoftwareSolution` conterrà tutti i file del progetto.

3.2.2 Convenzioni

Per il versionamento si farà uso del Versionamento Semantico, secondo le seguenti linee guida, se non indicato diversamente successivamente.

Un qualsiasi scatto di versione può avvenire solo attraverso una previa verifica

3.2.3 Comandi utili

Alcuni comandi utili sono:

- **Sincronizzazione con repository remota:** `git pull`;
- **Creazione di un nuovo branch**: `git branch <nome_branch>`;
- **Passaggio a un branch specificata:** `git checkout <nome_branch>`;
- **Aggiunta delle modifiche alla stage area:** `git add .`;
- **Creazione del commit con le modifiche:** `git commit`;
- **Push sul remote di un nuovo branch:** `git push --set-upstream origin <nome_branch>`;
- **Push su un branch già esistente:** `git push`.

3.3 Verifica

3.3.1 Regole Generali

- Il verificatore si occupa di controllare le Pull Request aperte dagli altri membri del gruppo;
- Il verificatore non può aver contribuito ai prodotti che deve verificare;
- Il verificatore non può modificare i prodotti che verifica;
- Il verificatore, qualora trovasse errori, è tenuto a commentare le PR, usando la funzione apposita di GitHub.

3.3.2 Metodi di Verifica

Il verificatore può far uso dei test automatici presenti nel *Piano di qualifica*.

Per le sezioni di codice è consigliato un walkthrough del prodotto, similmente si dovrebbe chiedere a chi abbia scritto il codice eventuali delucidazioni su elementi poco chiari

3.4 Change Management

Nel caso si dovessero cambiare sezione del codice bisogna controllare che eventuali modifiche non vadano ad impattare altri elementi.

Per evitare ciò bisogna fare uso del *Diagramma delle classi* presente nelle *Specifiche architetturali*.

4 Processi organizzativi

4.1 Gestione dei processi

È stato scelto il modello SCRUM con sprint di durata settimanale. Si è deciso di effettuare una riunione interna ogni mercoledì dove discutere dei seguenti punti:

- Resoconto dello sprint precedente;
- Eventuali problemi riscontrati e soluzioni adottate;
- Resoconto di ore e costi per ogni componente;
- Pianificazione prossimi sprint.

I resoconti dovranno essere preparati dai singoli membri prima di ogni esposizione.
Ogni incontro dovrà produrre un aggiornamento delle milestones (e diagramma di Gantt).
La pianificazione dei prossimi sprint consisterà nel dividere il lavoro in item e decidere il numero di membri adatto per item.
Ogni ciclo di SCRUM prevederà un nuovo responsabile, scelto per cognome in ordine alfabetico.
Il responsabile dovrà assegnare le persone agli item considerando la rotazione dei ruoli.
Il responsabile dovrà poi creare le issues su GitHub relative agli item e assegnarvici i membri.
Le issues dovranno essere collegate alle milestones di appartenenza dal responsabile.
Il responsabile si occuperà di aggiornare preventivo e consultivo totale, compilando nel PdP la sezione del ciclo di Scrum corrispondente.
È stato inoltre deciso che in caso di problemi singolarmente insormontabili andrà notificato il responsabile su Telegram che si occuperà di assegnare un membro per la risoluzione.
Il responsabile dovrà creare la issue su GitHub relativa al problema assegnandovici la persona scelta.
Le soluzioni adottate per la risoluzione del problema andranno documentate dal responsabile stesso.

4.2 Miglioramento dei processi

Durante l'attività di consuntivo di periodo dell'incontro settimanale, viene discusso il lavoro fatto fino ad ora, dando particolare peso ai problemi riscontrati.

In base a ciò si valutano le eventuali migliorie e correzioni applicabili.