UniBg Notes

Project Plan

Progetto di Ingegneria del software

Bilal Drissi 1087161
Daniele Iania 1085982
Mattia Nicastro 1086903

a.a 2024/2025

Indice

1. Introduzione

UniBg Notes è un progetto nato da un'idea molto semplice: fornire una piattaforma nella quale studenti delle varie facoltà dell'università degli studi di Bergamo possano condividere **appunti** delle lezioni, **schemi**, **temi d'esame** passati con le relative **risoluzioni** e qualsiasi altro **materiale** che possa risultare utile per un determinato corso di una qualsiasi facoltà, in modo tale che ogni studente abbia la possibilità di accedere a materiali ben più affidabili e specifici dell'università di Bergamo piuttosto che materiali di altri atenei che magari risultano anche diversi in argomenti o metodologie.
Per fornire questo servizio si è pensato di creare un'applicazione web facilmente utilizzabile da un qualsiasi *browser* che gestisce le varie utenze e tutti i materiali caricati mediante l'utilizzo di un ***database***.
I membri che svilupperanno questo progetto saranno **Daniele Iania, Mattia Nicastro** e **Bilal Drissi**.

2. Modello di processo

Il processo di sviluppo del progetto sarà diviso in **6 diverse fasi** che saranno così composte:

- **Analisi e comprensione del problema**: capire a fondo i punti focali del problema dal quale si parte, individuandone le caratteristiche principali

- **Ingegneria dei requisiti (detta anche *Requirements Engineering*)**: fase che permette di specificare le **funzionalità** che si implementeranno all'interno del *SW* che si sta creando (nel capitolo **8. Metodi e tecniche** sarà specificato come avverrà questa fase).

- **Progettazione e *design***: in questa fase rientra lo studio dell'architettura del *SW*, quindi la specifica di tutti i componenti (*HW* e *SW*) che ci dovranno essere per implementare l'applicazione.

- **Implementazione**: vera e propria stesura del codice (nel capitolo **8. Metodi e tecniche** sarà specificato come avverrà questa fase).

- ***Testing***: fase che permette di verificare e validare la corretta funzionalità di tutti i requisiti che sono stati implementati (nel capitolo **8. Metodi e tecniche** sarà specificato come avverrà questa fase).

- **Manutenzione**: fase finale del progetto che sarà continuamente portata avanti con eventuali correzioni di *bugs*, implementazioni di nuove funzionalità

richieste, miglioramento del codice o qualsiasi altra motivazione che porti a mettere mano al codice già finito.

Queste 6 fasi verranno svolte nell'ordine nel quale sono state esplicate e nei capitoli successivi verrà spiegato come si articolano al loro interno.

3. Organizzazione del processo e del team

L'organizzazione interna sarà interamente gestita mediante la metodologia **SCRUM** i cui punti principali sono specificati come segue:

- **Daily SCRUM**: quotidianamente verrà svolto un *meeting*, sia in presenza che da remoto, per discutere sull'avanzamento dello *sprint*; gli orari saranno decisi di volta in volta a seconda degli impegni di ciascuno dei membri.
- **Sprint**: dati i tempi brevi che si ha a disposizione, per ogni *sprint* è prevista una durata variabile da **1 settimana** fino ad un massimo di **3 settimane**.

Utilizzando la metodologia *SCRUM*, ogni singolo *sprint* vedrà al suo interno l'applicazione della metodologia di sviluppo scelta (specificata nel capitolo **8. Metodi e tecniche**) in maniera iterativa.
Oltre a questo, ci sarà una fase pre - *sprint* identificata come ***Sprint Backlog*** dove verranno specificati i requisiti che verranno implementati in quel dato *sprint*, mentre, alla fine di quest'ultimo, verrà svolto lo ***Sprint Retrospective*** che permetterà di analizzare lati positivi e negativi dello *sprint* appena passato individuando punti di miglioramento o mantenimento.

4. Procedure

All'unanimità le procedure standard che il *team* ha deciso e quindi seguirà durante tutto il ciclo di vita del *SW* sono:

- La programmazione avverrà in lingua **inglese**, quindi sia il nome delle variabili, metodi, commenti o qualsiasi altra cosa inerente al codice, verrà fatta interamente in lingua inglese.
- Le **variabili,** le **costanti** e i **metodi** verranno invece scritti seguendo la tecnica *camelCase*.
- I nomi delle **classi** inizieranno sempre con la lettera maiuscola ed anche essi seguiranno la tecnica *camelCase*.
- I **file** contenenti codice seguiranno allo stesso modo lo standard *camelCase*.
- Qualsiasi altro ***file*** presente nel *file system* del nostro progetto che avrà una natura diversa da quella di un *file* contenente codice *Java* (ad esempio *file* del *database, file* di testo, …) saranno salvati interamente in minuscolo ed utilizzando il classico trattino come separatore (-).

5. Attività di gestione

Come specificato nel capitolo **3. Organizzazione del processo e del *team*** ogni componente del *team* seguirà i fondamenti della tecnica *SCRUM* e per questo, tutte le attività di gestione sono comprese in questa metodologia, infatti, grazie al ***Daily SCRUM*** i membri saranno in grado di confrontarsi direttamente sul corretto procedimento dello sviluppo delle varie funzionalità di quel determinato *sprint*.
Si vuole mettere in chiaro come l'attività di *Sprint Review* non potrà essere applicata con il relativo cliente, poiché non si ha un cliente diretto ma sarà il *team* come tale che vorrà fornire un servizio universitario, quindi la **revisione** dello *sprint* che viene svolta alla fine di questo sarà effettuata tra i componenti del *team*.

6. Rischi

Durante lo sviluppo di questo progetto si potrà incontrare diversi rischi che potrebbero minare la corretta realizzazione del progetto stesso, qui di seguito verranno esplicitati quali potrebbero essere e delle possibili soluzioni:

- **Scarsità tempistiche**: viste le dimensioni del progetto e specialmente la presenza di altri esami ed altri impegni individuali, potrebbe succedere che il tempo a disposizione non basti per completare tutti i requisiti richiesti.
  Una possibile **soluzione** a questo è dare la sicurezza di quei **requisiti minimi** che sicuramente il *SW* alla fine del tempo a disposizione avrà e rispetterà; è pianificato che i requisiti minimi che sicuramente il *SW* rispetterà sono:
  - Gestione delle utenze
  - Condivisione materiale universitario
  - Filtraggio materiale universitario per corsi specifici

7. Personale

Il *team* è formato da tutte figure frequentanti la facoltà di ingegneria informatica dell'università degli studi di Bergamo (UniBg) e nello specifico sono:

- Mattia Nicastro
- Bilal Drissi
- Daniele Iania

8. Metodi e tecniche

In questa sezione verranno specificate le metodologie e le tecniche specifiche che seguiranno il *team* nelle principali fasi di vita del *SW*.

Ingegneria dei requisiti

Il *team* ha deciso che, per svolgere una corretta *RE* questa verrà fatta seguendo queste fasi:

- **Analisi**: viene studiata una specifica parte del problema e come questa può essere risolta mediante l'implementazione di una o più funzionalità, generando quindi uno o più requisiti.
- **Stesura**: si specifica il o i requisiti che sono stati pensati per risolvere quella specifica parte di problema.
- **Validazione**: si valida il o i requisiti, controllando di conseguenza che questo effettivamente possa risolvere il problema dal quale si è partiti.

Queste tre fasi verranno **ripetute** in maniera iterativa per ogni singolo "sotto - problema" che verrà identificato all'interno dell'applicazione che i membri dovranno creare.

Oltre a questo, verrà data una specifica priorità ai requisiti seguendo la tecnica ***Kano***, in modo tale che in fase di sviluppo si avrà una pianificazione migliore di quali requisiti dovranno essere fatti prima e quali dopo.

I requisiti verranno specificati senza l'utilizzo di linguaggi tecnici, ma utilizzando il **linguaggio naturale**.

Implementazione & *Testing*

Le varie specifiche di queste due fasi del progetto sono state accorpate perché la metodologia di implementazione che si è decisa di adottare comprende anche la fase di *testing*, infatti, verrà sviluppata questa applicazione web in maniera **incrementale**, seguendo questi quattro passaggi in maniera iterativa per ogni versione del progetto:

- Specifica degli obiettivi: si identificano quali sono gli obiettivi e quindi i **requisiti** che si vogliono **implementare** nella specifica versione che si sta creando.
- Identificazione e risoluzione rischi: in ogni versione che si crea possono presentarsi delle situazioni anomale ed in questa fase si cerca di **prevederle**

ed al tempo stesso di specificare una strategia da mettere in atto in caso si verifichino
- Sviluppo e *testing*: fase vera e propria della scrittura del codice inerente a questa versione specifica del prodotto dove alla fine di essa si verifica che i requisiti selezionati inizialmente siano stati effettivamente implementati.
- Pianificazione ciclo successivo: si inizia già a pensare a quali potrebbero essere le ipotetiche funzionalità e quindi i possibili requisiti che verranno implementati nella versione successiva.

IDE di sviluppo

L'IDE che verrà utilizzato per lo sviluppo del codice sarà *Eclipse*, utilizzando come linguaggio di programmazione *Java*; tutte le librerie aggiuntive o strumenti complementari che verranno utilizzati per lo sviluppo del codice, verranno specificati all'interno del capitolo **10. Pacchetti di lavoro**.

9. Garanzia di qualità

La qualità del software verrà garantita utilizzando come criterio i seguenti fattori:
- **Correttezza**: soddisfa, oltre alle specifiche del sistema, le esigenze degli utenti.
- **Affidabilità**: deve svolge le funzionalità proposte in maniera corretta e precisa, ossia che il sistema deve essere robusto e stabile durante l'utilizzo da parte degli utenti.
- **Integrità**: deve permettere l'accesso ai soli studenti/membri dell'ateneo per garantire uniformità, affinità e affidabilità dei vari documenti pubblicati, evitando così l'accesso da parte di utenti non autorizzati.
- **Usabilità**: deve garantire l'esecuzione sui vari sistemi operativi presenti sul mercato e la compatibilità su di essi.
- **Manutenibilità**: deve garantire che eventuali bug, errori e migliorie future, possano essere corretti e aggiunti senza troppe difficoltà.

10. Pacchetti di lavoro

Come specificato in precedenza nel capitolo **8. Metodi e tecniche** il linguaggio utilizzato per la scrittura del codice sarà solo e solamente *Java* e gli strumenti che utilizzeremo affiancati a quest'ultimo saranno un insieme di librerie ed ambienti di lavoro; qui di seguiti i principali:
- **Vaadin**: piattaforma che permette la realizzazione di applicazioni web mediante il solo utilizzo del linguaggio *Java*, sia per quanto riguarda la programmazione lato *front - end* sia per quanto riguarda la programmazione *back - end*.

- **Papyrus**: libreria *Java* che permette di creare diagrammi UML in maniera facilitata, tra cui *Use Case Diagram* e *Class Diagram*
- **SQLite**: database leggero e embedded che permette di gestire facilmente dati in un file locale, ideale per software web per archiviare e recuperare documenti e informazioni degli utenti senza necessità di un server esterno.
- **Window Builder**: *plugin* molto utile per la creazione della parte grafica mediante l'utilizzo della libreria *swing* con *Java*; esso propone un'interfaccia grafica intuitiva e di facile utilizzo.

Poiché, all'interno del progetto, ci sarà bisogno di interagire con un database relazionale verrà utilizzato **SQL** come linguaggio di creazione e manipolazione dati.

11. Risorse

Software e tool

- **Eclipse**: IDE per la realizzazione del progetto in JAVA (versione 2024-09)
    - **Maven**: estensione che aiuta a gestire le dipendenze, il processo di build e la sua struttura generale, rendendo più semplice lavorare su progetti grandi o complessi.
    - **Papyrus**: come spiegato nel capitolo **10. Pacchetti di lavoro**, permette di creare e gestire diagrammi UML.
    - **WindowBuilder**: come specificato nel capitolo **10. Pacchetti di lavoro**, permette la creazione e gestione dell'interfaccia grafica.
- **Vaadin**: come citato nel capitolo **10. Pacchetti di lavoro,** è una piattaforma per poter realizzare applicazioni web
- **SQLite**: come specificato nel capitolo **10. Pacchetti di lavoro**.

Hardware

Saranno utilizzati computer personali con specifiche e requisiti necessari per lavorare con i Software e tool sopracitati

Risorse umane

Il team del progetto sarà composto da tre membri. Ciascuno di essi ricoprirà il ruolo di **Product Owner**, **SCRUM Master** e **Development Team** siccome ogni membro svilupperà, supervisionerà il lavoro e aiuterà il mantenimento della focalizzazione sugli obiettivi durante la realizzazione del progetto.

12. Budget e programma

Non essendoci spese, non sarà previsto un budget monetario.
Il programma per lo sviluppo del progetto sarà suddiviso in sprint:

Sprint 1 - Comprensione e pianificazione iniziale

- Obiettivo: Comprendere il problema e definire le basi del progetto
- Attività:
    - identificazione dei requisiti iniziali e definizione degli obiettivi principali.
    - creazione del file iniziale dei requisiti e stesura della bozza del Project Plan.
- Durata: dal 06-11-2024 al 18-11-2024

Sprint 2 - Revisione e finalizzazione del Project Plan

- Obiettivo: Finalizzare il Project Plan e i requisiti.
- Attività:
    - Revisione e miglioramento del Project Plan.
    - Miglioramento del file dei requisiti.
    - Consegna della documentazione al docente.
- Durata:

Sprint 3 - Creazione dei modelli UML

- Obiettivo: Produrre i modelli UML per definire l'architettura del sistema.
- Attività:
    - Creazione dei diagrammi UML principali**.**
    - Revisione e validazione dei modelli UML con il team.
- Durata:

Sprint 4 - Inizio dello sviluppo del codice

- Obiettivo: Tradurre i modelli UML nelle prime componenti del sistema.
- Attività:
    - Implementazione delle funzionalità base definite nei requisiti iniziali.
    - Test unitari e integrazione delle funzionalità sviluppate.
- Durata:

Sprint 5 - Sviluppo incrementale e test

- Obiettivo: Aggiungere nuove funzionalità e migliorare quelle esistenti.
- Attività:
    - Sviluppo iterativo di nuove componenti del sistema.
    - Definizione e implementazione di casi di test per verificare le funzionalità.
- Durata:

Sprint 6 - Verifica e consegna finale

- Obiettivo: Verificare e validare il prodotto finito
- Attività:
    - Esecuzione di test completi sul sistema.
    - Consegna del progetto.

13. Possibili cambiamenti

Come specificato nel capitolo **3. Organizzazione del processo e del team**, verrà gestito lo sviluppo software con il metodo agile SCRUM. Attraverso la sua metodologia verrà monitorato il progresso del nostro software permettendo di riorganizzare le priorità in caso di problemi, riuscendo così a gestire con tranquillità i cambiamenti.

14. Consegna

La consegna del progetto con la propria documentazione verrà effettuata tramite GitHub, condividendo la relativa repository con il Prof. Gargantini e la Prof.ssa Bonfanti.