UNIVERSITÀ DI TRENTO

Dipartimento di Ingegneria
e Scienza dell'Informazione

# SleepCode

Progetto per il Corso di Ingegneria del Software
Anno Accademico 2023-2024

---

# Report Finale

---

*Descrizione:* resoconto dell'attività di lavoro del progetto: acquisizione e applicazione dei principi dell'ingegneria del software; distribuzione del lavoro, ruoli, attività; criticità; autovalutazione.

*Numero documento:* D5 *Versione documento:* 1.1

*Membri del gruppo:*
Raffaele Castagna
Alberto Rovesti
Zeno Saletti

*Numero gruppo:* G17

*Ultima revisione:* 6 febbraio 2024

# Indice

**1 Approcci all'ingegneria del software** **3**
1.1 I Seminari . . . 3
1.1.1 BlueTensor . . . 3
1.1.2 Il metodo Kanban . . . 3
1.1.3 IBM . . . 4
1.1.4 Meta . . . 4
1.1.5 U-Hopper . . . 4
1.1.6 Red Hat . . . 4
1.1.7 Microsoft . . . 5
1.1.8 Sistemi Legacy . . . 5
1.1.9 Gestione del ciclo di vita del software e modernizzazione 6
1.1.10 APSS e IT . . . 7
1.2 Considerazioni e approcci adottati . . . 7

**2 Organizzazione del lavoro** **8**
2.1 Ruoli e attività . . . 9
2.2 Distribuzione del carico di lavoro . . . 10

**3 Criticità** **10**

**4 Autovalutazione** **11**

**Consigli utili per la consultazione del testo:** Se il lettore per file `.pdf` attualmente in uso lo consente, è possibile navigare con più semplicità e velocità all'interno di questo documento cliccando sugli elementi dell'indice.

## Scopo del documento

Questo documento rappresenta il report finale del progetto. L'attenzione è rivolta all'impegno dedicato alla realizzazione di tutti i deliverables e del prototipo finale, all'organizzazione del lavoro e la sua distribuzione tra i membri del gruppo, nonché alla rilevanza delle nozioni relative all'ingegneria del software acquisite, e applicate in questo progetto, durante il corso e i seminari. Si specificano inoltre in questo documento non solo punti di forza, ma anche la consapevolezza delle criticità incontrate nel percorso e degli eventuali provvedimenti o soluzioni adottati per mitigare i problemi e riuscire nella realizzazione dell'intero progetto.

# 1 Approcci all'ingegneria del software

Nella presente sezione vengono riassunti brevemente i punti più significativi dei seminari tenutisi durante il corso, ponendo anche attenzione a metodi, principi e strumenti propri dell'ingegneria del software che sono stati descritti dai relatori.

Segue una breve riflessione su come alcune delle idee colte dai seminari abbiano influenzato gli approcci adottati durante la realizzazione del progetto.

## 1.1 I Seminari

### 1.1.1 BlueTensor

La presentazione "Metodi di Ingegneria del Software applicati allo sviluppo di un progetto di AI" di BlueTensor, tenuta dall'Ing. Jonni Malacarne, CEO di Bluetensor Srl, si concentra su soluzioni personalizzate nell'ambito dell'intelligenza artificiale (AI), enfatizzando la rapidità di sviluppo. Discute oltre 50 progetti AI in vari settori e un approccio multidisciplinare. Include collaborazioni scientifiche con università e massimizzazione del ROI. Presenta il "BlueTensor Framework", una piattaforma completa basata su AI per diverse applicazioni come supporto decisionale, ottimizzazione della produzione, miglioramento dell'esperienza utente e gestione del know-how aziendale. La presentazione copre anche prodotti AI-based ready to use, applicazioni in Natural Language Processing, Computer Vision e Predictive Analysis. Infine, descrive i metodi ingegneristici per l'analisi e la comunicazione con il cliente, come UML e Entity-Relationship Diagrams, e fornisce esempi di progetti e tecnologie utilizzate.

### 1.1.2 Il metodo Kanban

La presentazione "Kanban Simulation" dell'Università di Trento, a cura del Dr. York Rössler, introduce al metodo Kanban e ne illustra l'uso tramite una simulazione. Vengono trattati argomenti come l'analisi statistica e il debriefing della simulazione. Si esplora il contrasto tra il Kanban industriale e quello per il lavoro di conoscenza, evidenziando le origini e l'evoluzione del metodo. La simulazione fornisce esperienze pratiche sulle dinamiche di Kanban, tra cui l'uso di lavagne Kanban, la gestione del flusso di lavoro e l'impatto dei limiti di lavoro in corso (WIP). La presentazione sottolinea anche l'importanza di adattarsi e ispezionare il lavoro in ambienti incerti e rapidamente mutevoli.

### 1.1.3 IBM

La presentazione "Cloud Computing with IBM Cloud" di Ferdinando Gorga, specialista tecnico del cloud pubblico in IBM Technology Sales Italy, si concentra sulle offerte e le sfide affrontate dal Cloud IBM. Discute l'importanza strategica del Cloud IBM nella modernizzazione dei sistemi software, la trasformazione digitale e la creazione di nuovo business. La presentazione esplora vari aspetti del Cloud IBM, inclusi i suoi data center globali, potenza computazionale, e tecnologie software, con un'enfasi sulla sicurezza, crittografia di livello militare e l'adozione di tecnologie open source. Include esempi pratici di architettura e applicazioni del Cloud IBM.

### 1.1.4 Meta

Il seminario tenuto da Heorhi Raik, ingegnere software presso Meta, si è incentrato sulle tecniche della progettazione software tenute a Meta e le responsabilità della figura dell'ingegnere informatico. Raik ha discusso di temi quali la qualità dei dati, le interfacce utente, la privacy, i permessi, e le tecniche di web scraping/crawling. Ha evidenziato l'alta mobilità degli ingegneri software in Meta e come questi non si limitino alla codifica, ma contribuiscono anche alla progettazione architetturale e alla gestione dei progetti. Raik ha sottolineato l'approccio flessibile di Meta nella metodologia di sviluppo software, con team che adottano metodologie proprie. Tra gli strumenti e linguaggi più usati vi sono Hack, JavaScript, Python, SQL, e Mercurial per il controllo versione, oltre a VS Code e Buck.

### 1.1.5 U-Hopper

La presentazione "Lezioni apprese dalle trincee" di Daniele Miorandi, CEO di U-Hopper, tratta dell'esperienza aziendale di U-Hopper nel campo dei big data e dell'intelligenza artificiale. La presentazione copre diversi aspetti, inclusa la storia dell'azienda, il suo stack tecnologico, processi e strumenti di ingegneria del software, e progetti esemplari. Viene posta particolare enfasi sul ruolo dell'analisi dei big data, l'utilizzo di piattaforme basate su cloud, e le pratiche di sviluppo software come il test coverage e i processi di revisione. Inoltre, vengono discussi esempi pratici di applicazioni in vari settori come logistica, vendita al dettaglio, banking, sanità e produzione.

### 1.1.6 Red Hat

La presentazione "Il mio viaggio nel mondo open source" di Mario Fusco racconta la sua carriera nel campo dello sviluppo software e il suo coinvolgimento nel mondo open source. Partendo dal suo primo lavoro come sviluppatore Java, Fusco descrive il suo percorso professionale, inclusi progetti significativi e ruoli assunti, come la guida dello sviluppo di Drools

in Red Hat e la sua nomina a Java Champion. Si concentra sui vantaggi del lavoro in open source, come la condivisione di conoscenze, revisione tra pari, meritocrazia, visibilità e collaborazione comunitaria. La presentazione fornisce anche una panoramica delle licenze open source e discute come le aziende possono trarre profitto dal software open source. Infine, Fusco esamina diversi livelli di partecipazione ai progetti open source e condivide consigli su come mantenere sani questi progetti.

### 1.1.7 Microsoft

Il seminario è stato tenuto da diego Colombo che ha condiviso la sua esperienza di lavoro a Microsoft, concentrandosi sul ruolo del testing. Le slide rappresentano una disamina del processo di test nel software engineering. I test sono classificati in diverse categorie in base all'obiettivo e al livello di integrazione, come test unitari per singole funzioni e test funzionali per intere funzionalità. Vengono discussi i motivi per testare, come prevenire regressioni e individuare bug, e l'importanza dei test nel documentare e verificare il comportamento del software. Si elencano gli strumenti di supporto ai test, inclusi framework di unit testing e strumenti di automazione UI. Inoltre, viene descritto dove e quando avviene il testing dai computer di sviluppo alle fasi di produzione, e durante diverse fasi del ciclo di vita del software, come l'integrazione continua e il monitoraggio post-deployment. Infine, si fornisce una visualizzazione grafica di come i test si integrano nel flusso di lavoro di sviluppo e deployment.

### 1.1.8 Sistemi Legacy

La presentazione si concentra sui sistemi legacy nel contesto del software engineering. Il relatore, Andrea Molinari, tratta vari argomenti come l'inquadramento del settore, il fact-checking, le questioni legate ai sistemi legacy, in particolare l'obsolescenza, e la loro progettazione e gestione.

Molinari definisce obsoleti i sistemi legacy, ma ancora in uso perché soddisfano le esigenze originali. Si approfondiscono le caratteristiche distintive come hardware di tipo mainframe, effetti di lock-in con i fornitori, e l'occupazione di grandi spazi. Viene esaminata l'importanza dei protagonisti hardware come IBM e HP e si discutono specifiche tecniche e ambienti operativi, evidenziando la disponibilità e la resilienza di tali sistemi. Molinari prosegue esplorando il mondo dei sistemi legacy, con un focus particolare su COBOL e il suo ruolo nei sistemi bancari.

La discussione si estende alle moderne pratiche di migrazione di applicazioni mainframe verso il cloud e l'importanza del mainframe nella gestione delle transazioni quotidiane a livello globale. Si toccano anche le sfide dell'obsolescenza e la resistenza al cambiamento, mettendo in luce come l'innovazione può essere ostacolata da sistemi obsoleti. Inoltre, si parla

della transizione dai linguaggi legacy verso architetture e paradigmi di programmazione moderni, evidenziando l'importanza della transizionalità, della potenza di calcolo, della business continuity, e della protezione dei dati.

Successivamente si passa all'aggiornamento dei sistemi legacy, con particolare attenzione a COBOL e alle tecnologie mainframe. Si discute delle strategie per la loro modernizzazione, inclusa la migrazione verso nuove piattaforme. Viene evidenziato come COBOL sia ancora ampiamente utilizzato nonostante l'età, e vengono presentate soluzioni per integrare tali sistemi nel panorama IT attuale, mantenendo la continuità operativa e riducendo i costi complessivi.

Si passa poi a una discussione sull'evoluzione dei sistemi informatici legacy, enfatizzando l'importanza del linguaggio COBOL e le sfide della sua modernizzazione. Vengono trattati temi come il debito tecnologico, la gestione dei progetti legacy, e le metodologie di gestione del cambiamento, incluso l'Agile. Viene anche esplorato il ruolo degli Enterprise Architects nei progetti di trasformazione e l'uso di strumenti di modellazione come UML, SysML e ArchiMate per la progettazione di sistemi informativi complessi. Le diapositive suggeriscono l'adozione di approcci misti per la conversione e l'aggiornamento del codice legacy, con un'enfasi sulla preservazione del patrimonio informativo attraverso l'emulazione e la modernizzazione.

L'ultimo argomento è stato sull'architettura del software e la gestione dei progetti IT legacy. Si discutono temi come l'agilità rispetto alla pianificazione predittiva, i casi studio di grandi infrastrutture dati come i Data Lake in ambito sanitario, e le piattaforme di collaborazione. Viene posto l'accento sulla necessità di adattarsi al cambiamento e di valorizzare le interazioni individuali oltre ai processi. Viene anche messo in discussione se il legacy sia veramente "morto", suggerendo che, nonostante l'età, queste tecnologie e metodi possono ancora essere vitali e funzionali quando integrati con nuovi approcci e tecnologie.

### 1.1.9 Gestione del ciclo di vita del software e modernizzazione

La presentazione "Application Lifecycle Management and Modernization" di Gerardo Marsiglia, Enterprise Architect, si concentra sul ciclo di vita dell'applicazione e sulla sua modernizzazione. Analizza la gestione del ciclo di vita delle applicazioni, evidenziando aspetti come il project management, il software configuration management e il change management. Discute anche l'evoluzione dei processi di sviluppo software, passando dai metodi tradizionali a pratiche come DevOps e Agile. Viene inoltre esplorato l'uso di architetture cloud, l'importanza dell'automazione e la modernizzazione delle applicazioni per adattarsi a nuove tecnologie e piattaforme.

### 1.1.10 APSS e IT

La presentazione riguarda l'organizzazione e i servizi dell'Azienda Provinciale per i Servizi Sanitari (APSS) e il Dipartimento Tecnologie, inclusi i servizi di Ingegneria Clinica, Operazioni e Infrastrutture IT, Politiche per la Sanità Digitale e Soluzioni di Sanità Digitale. Viene fornita una panoramica delle iniziative tecnologiche, la gestione delle attrezzature sanitarie, l'innovazione e la digitalizzazione dei servizi, oltre ai progetti trasversali come il Fascicolo Sanitario Elettronico, il Patient Summary e la Telemedicina. Si discute l'importanza della trasformazione digitale per migliorare l'efficienza operativa, la sicurezza dei dati, e l'adozione di nuove tecnologie. Inoltre, vengono esaminati i piani per la migrazione dei servizi su piattaforme cloud e il rinnovo delle infrastrutture IT per garantire la continuità e la sicurezza dei servizi.

## 1.2 Considerazioni e approcci adottati

Il nostro approccio è stato in particolare ispirato dai seminari di BlueTensor e Kanban. Il primo ha posto l'accento sul ruolo dell'ingegneria del software durante un processo di sviluppo, rendendoci più consapevoli della necessità di rendere razionale tale processo: cura dei documenti di progetto, chiarezza, comunicazione e opportunità date da strumenti a supporto della collaborazione e sviluppo. Il secondo ha invece sottolineato l'importanza della comunicazione e della robustezza del team, specialmente nell'affrontare i problemi. Proprio da Kanban abbiamo notato la necessità di circoscrivere i problemi e suddividerli tra i vari membri senza sovraccaricarci in tempi eccessivamente brevi, organizzandoci però tramite strumenti di messaggistica (come descritto nella prossima sezione) anziché la classica lavagna.

Un altro approccio degno di nota è stato il tentativo di simulare la presenza del cliente durante lo sviluppo del progetto. Ciò avveniva in particolare su iniziativa del team leader sollecitando a porre domande e a ottenere feedback da parte dei vari membri del gruppo sul proprio operato, oppure raccogliendo dubbi e quesiti da porre ai professori (Raffaele Castagna, in quanto sviluppatore, ha posto le domande riguardanti l'implementazione).

Nonostante le difficoltà, questa esperienza ha avuto un risvolto positivo, rafforzando la coesione del team. Abbiamo imparato l'importanza di una pianificazione flessibile e di essere pronti di fronte a sfide inaspettate. La maggior parte dei seminari hanno migliorato la nostra comprensione dell'importanza di un'architettura del sistema ben pensata, che sia allo stesso tempo sicura, efficiente e scalabile e che ponga la dovuta attenzione anche al cliente.

## 2 Organizzazione del lavoro

Il nostro progetto ha richiesto un'attenta pianificazione e divisione del lavoro per sfruttare al meglio le competenze di ciascun membro del team.

L'idea iniziale è stata proposta da Raffaele Castagna, e successivamente raffinata attraverso discussioni con tutti i membri del gruppo. La parte di sviluppo del prototipo e i diagrammi strettamente collegati ad esso sono stati assegnati a Raffaele, mentre la scrittura, revisione e creazione dei diagrammi appartenenti ai vari deliverables è stata suddivisa tra Zeno Saletti e Alberto Rovesti.

Il ruolo di team leader è stato assegnato a Zeno Saletti, che si è occupato (oltre alle mansioni precedentemente descritte) di organizzare incontri, la maggior parte brevi ma settimanali durante il calendario delle lezioni, alcuni prolungati e con cadenza mensile. Questi incontri sono stati utili a suddividere e chiarire i ruoli, discutere idee e ottenere feedback sul lavoro svolto individualmente. Alberto Rovesti ha contribuito nell'assistenza delle mansioni assunte da Zeno Saletti (escluse quelle di team leader), spartendo tra i due membri i compiti di stesura e revisione di diagrammi e testi, in particolar modo nel documento D2.

La collaborazione tra i membri del team è stata facilitata dall'uso di strumenti come Git e Github per la condivisione di risorse, editor di diagrammi come Lucid Chart, editor di testo online come Google Documenti e Whatsapp e Telegram per la comunicazione quotidiana. Questo ha permesso al team di rimanere costantemente aggiornato sullo stato del progetto e di affrontare tempestivamente eventuali problemi. Le riunioni mensili, sia in presenza che virtuali, hanno offerto momenti preziosi per la condivisione di aggiornamenti, la discussione di strategie e l'adattamento dei piani in base alle esigenze del progetto.

## 2.1 Ruoli e attività

| Componente del team | Ruoli | Attività principali |
|---|---|---|
| Raffaele Castagna | Sviluppatore, Concept designer, Architetto | Ha concepito e sviluppato l'idea originale del progetto, sia nelle sue funzionalità che nella sua struttura architetturale, implementata infine nel prototipo finale. Ha contribuito fornendo supporto a tutti i deliverables, ma ha assunto un ruolo chiave nel D4, guidandone lo sviluppo e buona parte della stesura del documento; ha concepito il mock-up di front-end nel D1 (sviluppando successivamente l'interfaccia), l'architettura definita nel diagramma dei componenti in D2 e nel class diagram in D3. |
| Alberto Rovesti | Progettista, Revisore, Correttore | Ha svolto un lavoro di supporto progettuale nei deliverable D1 e D2, definendo parte degli Use Cases e dei diagrammi nel D2 ed effettuando controlli al class diagram e codice OCL in D3; ha dedicanto particolare attenzione al D5, guidando soprattutto la stesura del paragrafo sui seminari. Ha revisionato i deliverables finali. |
| Zeno Saletti | Project leader, Redattore, Analista, Progettista | Coordinazione del lavoro e dei ruoli dei membri; organizzazione periodica e gestione dei meeting; analisi, definizione formale e organizzazione dei requisiti (D1, D2) e del contesto (D2); definizione di buona parte degli Use Cases e diagrammi in D2; supporto architetturale per i componenti (D2) e class diagram (D3); stesura definitiva dei documenti in LaTeX; cura della veste grafica e dei diagrammi finali per i documenti D1, D2, D3, D4, D5. |

### 2.2 Distribuzione del carico di lavoro

La tabella seguente mostra la distribuzione indicativa del tempo dedicato (in ore) tra i membri del gruppo, distinguendo il lavoro dedicato ai diversi deliverables.

| Componente del team | D1 | D2 | D3 | D4 | D5 | Totale |
|---|---|---|---|---|---|---|
| Raffaele Castagna | 23 | 24 | 26 | 102 | 3 | 178 |
| Alberto Rovesti | 22 | 28 | 15 | 9 | 16 | 90 |
| Zeno Saletti | 29 | 32 | 28 | 11 | 8 | 108 |
| Totale | 74 | 84 | 69 | 122 | 27 | 376 |

Si può notare una notevole differenza nel tempo speso per la composizione del documento D4, che ovviamente comprende anche il lavoro necessario all'implementazione, testing, cura e documentazione del codice. Infatti, in qualità di sviluppatore, Raffaele Castagna ha accumulato il maggior numero di ore durante la realizzazione di questa parte del progetto, dedicando la maggior parte degli sforzi alla creazione del prototipo finale in tutti i suoi aspetti, documentandosi quando necessario sui framework e gli strumenti impiegati.

L'ammontare di ore dedicato invece da Alberto Rovesti per il documento D5 è giustificato dal tempo impiegato per redarre il paragrafo relativo ai seminari.

La quantità di ore leggermente superiore registrata da Zeno Saletti, in alcuni documenti, è indicativa del tempo dedicato, oltre agli altri compiti, anche alla stesura definitiva dei documenti in LaTeX e al ruolo di team leader nel supervisionare il lavoro dei membri.

Come descritto nella prossima sezione, la mancanza di ruoli chiari durante l'inizio del progetto ha comportato un accumulo di ore relativamente elevato da parte di tutti i membri sugli stessi documenti, in particolare D1 e D2. Inoltre, ciò è in parte dovuto alle attività di supporto, aggiornamento e revisione delle diverse parti del progetto.

## 3 Criticità

Una prima criticità, riscontrata durante le primissime fasi di sviluppo del progetto (in corrispondenza del concepimento dei documenti D1 e D2), è stata la difficoltà nel suddividere ruoli e compiti. Migliorando la comunicazione, grazie alle tecnologie a disposizione, agli incontri sempre più regolari e alle scadenze imposte dal team leader, è stato possibile suddividere razionalmente ed efficientemente il lavoro, riconoscendo e sfruttando anche le capacità e conoscenze pregresse di ogni membro.

Durante lo sviluppo della dell'applicativo di SleepCode, sono state rilevate diverse sfide e criticità, alcune delle quali hanno richiesto una riconsiderazione significativa dei nostri piani iniziali. Una delle principali difficoltà è stata legata all'integrazione di Firebase, il servizio di database scelto per

questo progetto. Oltre agli ostacoli specificati nel documento D4 (sezione riguardante il soddisfacimento dei requisiti), la criticità principale è emersa dalla integrazione di Firebase nel progetto durante lo sviluppo delle API. La maggior parte dei metodi precedentemente implementati e forniti da Firebase, infatti, erano incompatibili con chiamate API in quanto hook. Ciò ha comportato un rallentamento durante lo sviluppo e una quantità di API inferiore alle aspettative iniziali.

Oltre a questo abbiamo riscontrato difficoltà durante il testing con la libreria Jest, tuttavia queste difficoltà sono state poi risolte attraverso pacchetti specializzati per l'integrazione di Jest con Firebase.

# 4 Autovalutazione

Nel corso dello sviluppo del nostro progetto, ciascun membro del team ha avuto l'opportunità di riflettere sul proprio contributo e sulle aree di miglioramento personale. Riteniamo che abbiamo lavorato tutti con costanza e coerenza, e che il prodotto finale sia robusto e di qualità soddisfacente. Sulla base di queste riflessioni riteniamo che la seguente autovalutazione rispetti l'impegno, e la qualità del lavoro, che ogni membro del team pensa di aver dedicato secondo i propri compiti:

| Componente del team | Voto |
|---|---|
| Raffaele Castagna | 29 |
| Alberto Rovesti | 28 |
| Zeno Saletti | 28 |

## Versioni

- *Versione 1.0*: prima stesura.
- *Versione 1.1*: revisione in vista della consegna.