Specifiche per il Progetto di Laboratorio di Sistemi Operativi

Tutor: Andrea Colledan – andrea.colledan@unibo.it
Anno Accademico 2021-22

Versione 1.1 – 6 Settembre 2022

In questo documento sono presentate le specifiche del progetto del corso di Sistemi Operativi per la Laurea Triennale in Informatica per il Management dell'Università di Bologna, anno accademico 2021-2022. **Tali specifiche possono essere soggette a cambiamenti e a revisioni. Ogni modifica viene comunicata tempestivamente attraverso la bacheca di annunci ufficiale del corso su Virtuale e sul gruppo ufficiale del corso.** In ogni momento e compatibilmente con la disponibilità dei docenti, è possibile prenotare un ricevimento in presenza o virtuale sulla piattaforma Teams. A tal fine, contattare il tutor all'indirizzo `andrea.colledan@unibo.it` specificando la motivazione della richiesta di colloquio. Il progetto verte attorno alla realizzazione di un semplice file server condiviso. Le tre fasi del progetto descritte in questo documento sono:

1. Preliminari: formazione dei gruppi e inizializzazione del progetto
2. Il progetto e la sua realizzazione
3. La consegna e la discussione del progetto

1 Preliminari

1.1 Formazione dei gruppi

I gruppi sono costituiti da tre (3) o quattro (4) persone. Coloro che intendono partecipare all'esame comunicano entro e non oltre il **31 Maggio**

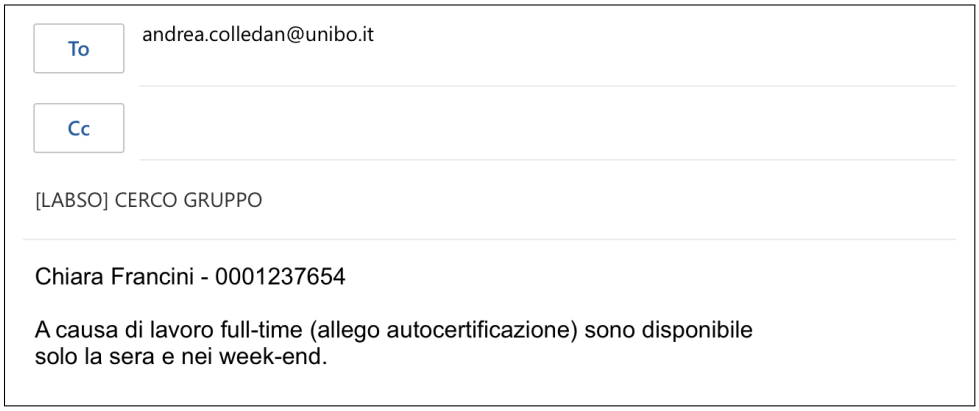
2022 la composizione del gruppo di lavoro al tutor, via posta elettronica. Il messaggio deve essere inviato dalla propria mail istituzionale (dominio `studio.unibo.it`). Deve avere come oggetto **[LABSO] FORMAZIONE GRUPPO** e contenere:

- Il nome del gruppo
- Per ciascun componente una riga contenente:
  - Nome, cognome, matricola
- Un indirizzo email di riferimento a cui mandare le notifiche. **N.B. è incarico del referente trasmettere eventuali comunicazioni agli altri membri del gruppo.**

Esempio di email di formazione gruppo

Chi non riuscisse a trovare un gruppo può inviare allo stesso indirizzo email un messaggio con oggetto **[LABSO] CERCO GRUPPO**, specificando:

- Nome, cognome, matricola
- Eventuali preferenze legate a tempi di lavoro. Si cercherà di costituire gruppi di persone con tempi di lavoro compatibili, nel limite delle possibilità.



Esempio di email di ricerca gruppo

Le persone senza un gruppo vengono assegnate il prima possibile **senza possibilità di ulteriori modifiche**. Per tale motivo è caldamente consigliato rivolgersi al tutor per la ricerca di un gruppo come *ultima* soluzione.

1.2 Creazione della repository

Il codice sorgente del progetto è contenuto in uno spazio cloud del servizio online GitLab. Questo spazio è creato e gestito seguendo la procedura descritta di seguito. **N.B. Le istruzioni che seguono devono essere completate entro e non oltre la scadenza di presentazione dei gruppi.**

1. Ogni membro del gruppo crea un account su GitLab (a meno che non ne abbia già uno).

2. Il referente del gruppo crea un nuovo progetto cliccando su **"+"** → **"New project/repository"** nella barra superiore della schermata principale di GitLab e selezionando **"Create blank project"**. Inserisce `LABSO_<NOME_GRUPPO>` come nome del progetto, imposta la repository come privata e clicca su **"Create project"**.

3. Il referente aggiunge ogni membro del gruppo alla repository. Per fare ciò, dal menu della repository seleziona **"Project information"** → **"Members"** e in seguito clicca su **"Invite members"**. Nella schermata di invito membri, il referente cerca ciascun membro col nome utente con cui quest'ultimo è iscritto a GitLab, seleziona come ruolo `Developer` e clicca su **"Invite"**.

4. Allo stesso modo, il referente aggiunge il tutor come reporter della repository. Per fare ciò, invita l'utente `andcol` con ruolo `Reporter`.

2 Il progetto

La seguente sezione fornisce le specifiche e le informazioni necessarie alla realizzazione del progetto.

2.1 Antefatto

L'anno è il 2026. È il tuo primo giorno di lavoro presso la NovaTec s.r.l. di Nervesa della Battaglia (TV). Hai appena finito di sistemare le tue cose sulla scrivania quando un collega ti informa che gli è stato chiesto di accompagnarti dal titolare dell'azienda, nel suo ufficio. Vi incamminate assieme lungo uno stretto corridoio col pavimento in linoleum, superate la macchinetta del caffè e infine ti lascia davanti a una precaria porta a vetri su cui è appeso un cartello "aprire piano". Attraversata la porta, ti ritrovi al cospetto di un uomo sulla sessantina, coi capelli grigi pettinati a lato, le guance rosse per la rasatura e una polo azzurra che sembra più vecchia di te. Dall'alto della scrivania in truciolato a cui è seduto, il titolare della NovaTec s.r.l. di Nervesa della Battaglia (TV) distoglie lo sguardo dal mastodontico monitor CRT che fino a quel momento deteneva la sua attenzione indivisa, inforca degli spessi occhiali senza montatura e ti scruta per un istante. Senza perdere tempo in presentazioni, si rivolge a te: "Tu che hai studiato informatica e te ne intendi di computer," inizia, "ho bisogno che mi scrivi un programmino che permetta a quelli nell'ufficio di là di condividere tra loro dei file e lavorarci assieme. Riesci?" Dopo un attimo iniziale di sorpresa, gli rispondi: "Credo di sì, ma se devono solo condividere e modificare dei file, non possono semplicemente usare Google Docs?" Il tuo interlocutore corruccia la fronte e ribatte: "Cos'è, una di quelle chiavette che si collegano a internet?" Non sai cosa rispondere. Dopo una lunghissima pausa in cui metti in discussione le tue scelte di vita, ti ricordi che devi parlare: "No, non fa niente, le scrivo il programmino, non si preoccupi." Lui non ti saluta, si toglie gli occhiali e si rimette a fissare il monitor. Mentre esci dalla stanza, facendo attenzione a chiudere piano la porta, una singola lacrima ti riga la guancia. È il tuo primo giorno di lavoro presso la NovaTec s.r.l. di Nervesa della Battaglia (TV).

2.2 Funzionalità

Il progetto richiede di implementare un semplice file server che permetta a più utenti di connettersi contemporaneamente per condividere, visualizzare e modificare file testuali. Il file server resta in esecuzione e custodisce i veri e propri file su disco. Gli utenti del servizio accedono ai suddetti file dalla propria macchina attraverso un'applicazione client che comunica col server e che permette loro di:

- Creare nuovi file
- Leggere i file esistenti
- Modificare e rinominare i file esistenti
- Eliminare i file esistenti

Quando un utente richiede di leggere un file, il server apre una *sessione di lettura* per quel file. Quando un utente richiede di modificare un file, viene aperta una *sessione di scrittura* per quel file.

L'apertura delle sessioni di lettura e scrittura rispetta delle regole fondamentali: un numero arbitrario di utenti può leggere lo stesso file contemporaneamente, ma finché c'è almeno un utente che sta leggendo il file nessuno può modificare lo stesso file. Nell'altro verso, quando un utente sta modificando un file, nessuno può leggere o modificare lo stesso file. Ovviamente, nessun file può essere rinominato o eliminato mentre qualcuno lo sta leggendo o modificando.

Se un utente richiede un'operazione (e.g. leggere un file) in un momento in cui non può essere portata a termine (e.g. perché un altro utente sta modificando lo stesso file) *non* viene rifiutato dal server, bensì rimane in attesa finché non è possibile soddisfare la sua richiesta (e.g. finché la sessione di scrittura non viene conclusa).

2.3 Requisiti dell'implementazione

Linguaggio Il progetto è implementato in Java (ultima versione LTS: Java SE 17). La comunicazione di rete è implementata attraverso i socket, come visto a lezione. Il prodotto finale consiste di due parti: un'applicazione client e un'applicazione server che assieme implementano le funzionalità descritte in Sezione 2.2.

Client Il client viene avviato da linea di comando e accetta come argomenti l'indirizzo IP e la porta del file server a cui connettersi. Ad esempio:

```
java Client 192.168.0.1 25565
```

Se il server non è raggiungibile all'indirizzo e alla porta specificati, viene restituito un messaggio di errore. Se la connessione va a buon fine, il client resta in attesa di istruzioni dall'utente. I comandi accettati sono i seguenti:

- Il comando `list` restituisce una lista di tutti i file presenti sul server. Per ciascun file sono riportati
  - il nome del file
  - la data e l'ora dell'ultima modifica al file
  - quanti utenti stanno attualmente leggendo o modificando il file
- Il comando `create` prende come argomento il nome di un nuovo file che l'utente vuole creare sul server. Ad esempio

  ```
  create example.txt
  ```

  Se non esiste già un file con lo stesso nome sul server, questo viene creato, altrimenti il comando restituisce un messaggio di errore.
- Il comando `read` prende come argomento il nome di un file da leggere. Ad esempio

  ```
  read example.txt
  ```

  Se il file esiste, l'applicazione client visualizza il contenuto del file ed entra in *modalità di lettura*. In modalità di lettura, il client mette a disposizione il seguente comando aggiuntivo:
  - Il comando `:close` chiude il file e termina la sessione di lettura. Il client esce dalla modalità di lettura.

  Se il file non esiste, il comando `read` restituisce un messaggio di errore.
- Il comando `edit` prende come argomento il nome di un file da modificare. Ad esempio

  ```
  edit example.txt
  ```

  Se il file esiste, il client visualizza il contenuto del file ed entra in *modalità di scrittura*. Nella modalità di scrittura, il client mette a disposizione i seguenti comandi aggiuntivi:

  - Il comando `:backspace` elimina l'ultima riga del file. Se il file è vuoto, non fa nulla.
  - Il comando `:close` chiude il file e termina la sessione di scrittura. Il client esce dalla modalità di scrittura.
  - Ogni comando che inizia con un carattere diverso da ':' viene interpretato come una riga di testo regolare, che viene aggiunta in coda al file.

  Non è richiesto che l'utente salvi esplicitamente le proprie modifiche: ogni modifica apportata lato client viene immediatamente salvata lato server. Se il file richiesto non esiste, il comando `edit` restituisce un messaggio di errore.

- Il comando `rename` prende come argomenti il nome di un file esistente e il nuovo nome da assegnare a quel file. Ad esempio

  ```
  rename example.txt test.txt
  ```

  Se il file esiste e non esiste già un file col nuovo nome, il file esistente viene rinominato e risulta accessibile sotto il nuovo nome. Altrimenti, il comando restituisce un messaggio di errore.

- Il comando `delete` prende come argomento il nome di un file da eliminare. Ad esempio

  ```
  delete example.txt
  ```

  Se il file esiste, viene eliminato dal server e non risulta più accessibile.

- Il comando `quit` arresta il client.

Server Il server viene avviato da linea di comando e accetta come argomenti il percorso della cartella della macchina host in cui verranno custoditi i file e la porta su cui il server resta in ascolto di richieste di connessione da parte dei client. Ad esempio

```
java Server .  25565
```

avvia il file server e lo mette in ascolto sulla porta 25565. I file creati dagli utenti vengono custoditi nella stessa cartella dell'eseguibile. Anche il server accetta istruzioni da riga di comando e risponde ai seguenti comandi:

- Il comando `info` mostra a schermo le seguenti informazioni sul server:
  - Il numero di file gestiti dal server
  - Il numero di client attualmente connessi in lettura
  - Il numero di client attualmente connessi in scrittura
- Il comando `quit` disconnette eventuali client ancora connessi e chiude il server.

È richiesto che il file server conservi i file in modo **persistente**: chiudendo e riaprendo il server con lo stesso percorso come primo argomento, gli utenti devono essere in grado di ritrovare gli stessi file.

2.4 Altri aspetti implementativi

Per essere valido, il progetto deve implementare le funzionalità descritte in Sezione 2.2 e rispettare almeno i requisiti descritti in Sezione 2.3. Ogni altra scelta è a libera discrezione del gruppo. In particolare, sono a discrezione del gruppo le scelte riguardanti:

- Il nome dell'applicazione (divertitevi)
- Come il file server salva e organizza i file su disco
- Eventuali feature aggiuntive, e.g. struttura a cartelle, undo/redo nell'editor, permessi per accedere ai file, etc. **N.B. in caso di estensioni, client e server devono *comunque* essere in grado di rispondere *come richiesto* ai comandi descritti in Sezione 2.3.**
- ...

Questi aspetti non influiscono sulla valutazione del progetto, a meno che le scelte fatte in merito non stravolgano la natura dell'applicazione.

2.5 La documentazione

La documentazione è parte integrante del progetto. Non vi sono vincoli sugli strumenti utilizzati per redigere la documentazione (e.g. LaTeX, MS Word, Google Docs, etc.), l'importante è che al termine della stesura venga consegnato un file **PDF** e che tale file rispetti i requisiti descritti in questa

sezione. La documentazione deve avere una lunghezza di **almeno 10 pagine** (intese come facciate), compresa l'intestazione, e deve essere scritta con **font di grandezza 12pt**. Il limite di pagine è un *lower bound*: non esiste un *upper bound* per la lunghezza della documentazione, che può quindi essere lunga a piacimento.

Struttura della documentazione L'intestazione della documentazione deve avere titolo "Laboratorio di Sistemi Operativi A.A. 2021-22" e deve contenere

- Il nome del gruppo
- L'indirizzo email del referente del gruppo
- Per ogni componente del gruppo:
  - Nome, cognome, matricola

Il corpo della documentazione deve coprire almeno i seguenti argomenti:

1. Descrizione del progetto consegnato:
   (a) Architettura generale: visione **di alto livello** di quali sono le componenti in gioco, di come interagiscono tra loro e delle informazioni che si scambiano per far funzionare il progetto. In questa sezione sono particolarmente utili degli **schemi**.
   (b) Descrizione dettagliata delle singole componenti:
   - Client, server e relativa suddivisione dei compiti
   - Sotto-componenti di client e server: thread, unità logiche, etc.
   - Se necessaria, descrizione delle classi fondamentali e dei loro metodi principali

   (c) Suddivisione del lavoro tra i membri del gruppo
2. Descrizione e discussione del processo di implementazione:
   (a) **Descrizione dei problemi** e degli ostacoli incontrati durante l'implementazione, con discussione e giustificazione delle **soluzioni adottate** e di eventuali **soluzioni alternative**. In particolare:

- Problemi legati alla concorrenza: quali sono le risorse condivise, quando e perché si rende necessaria la mutua esclusione, etc.
- Problemi legati al modello client-server: come vengono instaurate, mantenute e chiuse le connessioni, cosa succede in caso di interruzioni anomale del client o del server, etc.

(b) Descrizione degli strumenti utilizzati per l'organizzazione. In particolare, applicazioni, piattaforme, servizi utilizzati per:

- Sviluppare il progetto (e.g. Eclipse, IntelliJ IDEA, Visual Studio Code, etc.)
- Comunicare tra i membri del gruppo (e.g. Teams, Skype, Discord, etc.)
- Condividere il codice prodotto (e.g. come avete usato GitLab).
- Tenere traccia del lavoro svolto, del lavoro rimasto da svolgere, delle decisioni ad alto livello prese dal gruppo, etc. (e.g. Google Docs, Trello, etc.)

3. Requisiti e istruzioni passo-passo per compilare e usare le applicazioni consegnate

    - Se presenti, descrizione delle estensioni implementate e di come usarle

L'organizzazione delle sezioni non deve per forza rispecchiare esattamente quella appena riportata. Per esempio, è possibile trattare un argomento in più sezioni, o trattare più argomenti nella stessa sezione. È anche possibile aggiungere informazioni non espressamente richieste nell'elenco qualora fossero utili. L'importante è che *almeno* i contenuti elencati siano *facilmente* rintracciabili nel corpo della documentazione.

Scopo della documentazione La documentazione deve puntare a dare al lettore una visione chiara di come funziona il progetto e di come sono stati affrontati gli ostacoli di implementazione, *senza* che il lettore debba conoscere il codice sorgente. I rimandi al codice (e.g. "Vedi `Server.java`, righe 150-155") sono apprezzati, ma il codice sorgente *non* deve sostituire la documentazione. Dall'altro lato, la documentazione non deve essere una

semplice ripetizione delle specifiche. Per la natura del progetto è pressoché scontato che ci sia (ad esempio) una classe del server in cui vengono ricevute le richieste del client, o una classe in cui viene implementato l'editor di testo, etc. Quello su cui la documentazione deve concentrarsi è *cosa* accade quando arriva una richiesta, *come* è implementato l'editor, etc. nella *vostra particolare implementazione* delle specifiche.

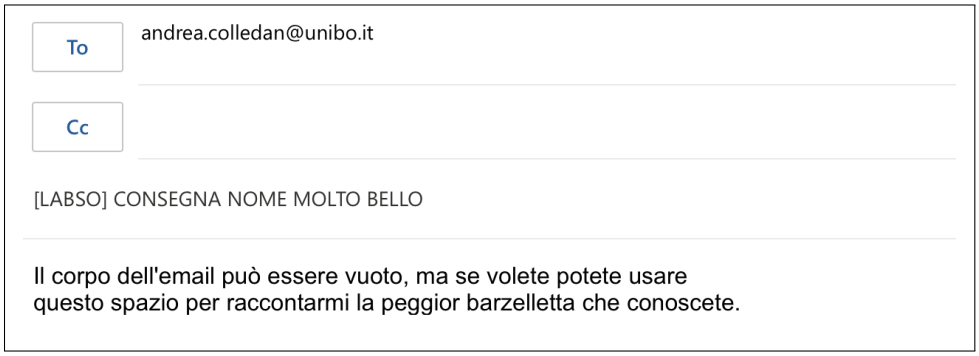
3 Consegna e Discussione

La seguente sezione descrive la procedura di consegna del progetto e le modalità di discussione dello stesso.

3.1 Consegna del Progetto

Al momento della consegna, la repository deve contenere i sorgenti del progetto e la documentazione. Quest'ultima deve essere contenuta in un singolo file PDF di nome `DOCUMENTAZIONE.pdf`. Per effettuare la consegna:

1. Dal menu del progetto, selezionare **"Repository"** → **"Tags"**.
2. Cliccare su **"New tag"**.
3. Inserire `Consegna` come nome del tag e aggiungere un messaggio qualsiasi *non vuoto*. L'aggiunta di un messaggio è essenziale affinché il tag venga registrato con la data e l'ora di creazione.
4. Cliccare su **"Create tag"** per completare la creazione del tag `Consegna`.
5. Una volta creato il tag, notificare la consegna al tutor inviando un'email all'indirizzo `andrea.colledan@unibo.it` con oggetto **[LABSO] CONSEGNA <NOME GRUPPO>**.



Esempio di email di notifica di consegna

L'implementazione e la documentazione del progetto devono essere consegnate assieme. Vi sono tre appelli per la consegna del progetto, con le rispettive scadenze:

- Lunedì **4 Luglio 2022** (04/07/22), ore 23:59.
- Lunedì **3 Ottobre 2022** (03/10/22), ore 23:59.
- Lunedì **28 Novembre 2022** (28/11/22), ore 23:59.

N.B. Per la data e l'ora di consegna del progetto fanno fede la data e l'ora di creazione del tag `Consegna`.

Su AlmaEsami è presente un appello per ciascuna di queste scadenze. Il voto finale è individuale, per cui *tutti* i membri del gruppo sono tenuti ad iscriversi su AlmaEsami all'appello in cui il gruppo intende discutere il progetto. **N.B. La data dell'appello su AlmaEsami corrisponde alla data di scadenza, non alla data di discussione.**

3.2 Discussione del Progetto

In seguito alla consegna, il luogo, la data e l'ora della discussione verranno fissati e comunicati al referente del gruppo. Solitamente, la discussione avviene entro due settimane dalla scadenza di consegna. La discussione consiste in:

- Una breve demo del progetto implementato, che può essere effettuata indifferentemente su rete locale (client e server eseguono sulla stessa rete e/o macchina) o su internet (ad esempio avviando da remoto il server su una macchina di laboratorio).

- Alcune domande ai membri del gruppo sull'implementazione del progetto, l'organizzazione del lavoro e i contributi personali di ciascuno. Durante questa fase verrà richiesto di mostrare e spiegare frammenti di codice sorgente.

Siccome la discussione prevede di dimostrare il proprio progetto e spiegare il proprio codice, è consigliabile (benché non obbligatorio) che il gruppo si presenti in sede di discussione con un proprio portatile.

Al termine della discussione, ad ogni membro del gruppo viene assegnato un punteggio da 0 a 5 in base all'effettivo contributo alla realizzazione del progetto dimostrato in sede di discussione. Questo punteggio si somma alla valutazione del progetto (vedi Sezione 4.4) per determinare il voto finale in trentesimi di ogni singolo membro del gruppo.

4 Griglie di Valutazione

La valutazione del progetto tiene conto del processo di implementazione del progetto, della documentazione e della partecipazione alla comunità del corso. **L'aspetto implementativo e quello documentativo hanno peso uguale**. La valutazione è basata sui criteri contenuti nelle seguenti griglie di valutazione.

4.1 Implementazione

La valutazione dell'implementazione del progetto si basa sull'analisi del codice Java, sull'implementazione corretta delle specifiche e sull'uso dei costrutti del linguaggio per la creazione di soluzioni efficienti e tolleranti ai guasti. Da questo punto di vista, l'obiettivo fondamentale del progetto è quello di dimostrare che i membri del gruppo sono in grado di:

- Utilizzare il multithreading per gestire situazioni in cui molti processi (e.g. più letture contemporanee) devono poter eseguire simultaneamente.

- Riconoscere quando una struttura dati è una risorsa condivisa, ovvero quando viene acceduta concorrentemente da più thread, e individuare i problemi di concorrenza associati.

- Adottare di conseguenza i costrutti di mutua esclusione e sincronizzazione adeguati al caso.

Un progetto in cui non viene fatto uso di alcun costrutto di sincronizzazione è pertanto automaticamente insufficiente, così come un progetto che evita i problemi di concorrenza eseguendo tutta la logica applicativa su un solo thread (nonostante nella vita reale questa sia una strada assolutamente percorribile).

La valutazione tiene conto anche di aspetti esterni alla programmazione intesa in senso stretto, come la ripartizione e l'organizzazione del lavoro all'interno del gruppo, la qualità del processo di implementazione e la tracciabilità degli artefatti di sviluppo. In particolare, una ripartizione precisa dei compiti all'interno del gruppo è importante. Questo non significa che più membri del gruppo non possono collaborare o non devono sapere nulla l'uno del codice degli altri, ma significa che ciascun componente è l'esperto di una (o più) parti circoscritte del progetto (e.g. gestione della connessione client-server, gestione dei comandi dell'utente, etc.) e se ne assume la responsabilità. Per ciascuno degli aspetti riportati nella seguente tabella viene assegnato un punteggio da 0 (insufficiente) a 5 (ottimo).

CRITERIO	DESCRIZIONE
Rispetto delle specifiche	Il progetto implementa correttamente le funzionalità descritte in Sezione 2.2 e rispetta i requisiti di Sezione 2.3.
Qualità del codice	Il gruppo usa correttamente i costrutti e le strutture dati offerti da Java per gestire concorrenza e distribuzione. Gestione adeguata di eccezioni e casi limite. Il codice è leggibile e ben commentato.
Organizzazione del lavoro nel gruppo	La distribuzione del lavoro all'interno del gruppo è ben delineata ed omogenea. Ciascun membro del gruppo è capace di indicare e spiegare i propri contributi. Il processo di implementazione è ben delineato e tracciabile.

4.2 Documentazione

La valutazione della documentazione verte sull'analisi dello scritto e sulla sua capacità di descrivere con chiarezza il prodotto consegnato, i problemi riscontrati durante l'implementazione e le soluzioni adottate, **soprattutto grazie all'uso di esempi**. Per ciascuno degli aspetti riportati nella seguente tabella viene assegnato un punteggio da 0 (insufficiente) a 5 (ottimo).

CRITERIO	DESCRIZIONE
Qualità dell'informazione	La documentazione fornisce una descrizione chiara e completa del progetto implementato e del processo di implementazione.
Uso di esempi	Presenza di esempi (narrativi, grafici, etc.) utili alla comprensione delle scelte implementative del gruppo o di scenari d'uso specifici.
Analisi delle scelte implementative	Individuazione e descrizione dei problemi incontrati durante l'implementazione, con particolare enfasi sui problemi legati alla concorrenza e alla distribuzione. Discussione delle soluzioni adottate e di soluzioni alternative valide.

4.3 Partecipazione alla comunità del corso

Durante la realizzazione del progetto, i vari gruppi sono incoraggiati a esporre i propri dubbi, a fare domande e a rispondere alle domande altrui sui canali di comunicazione messi a disposizione dal corso. La collaborazione tra gruppi in questi termini è valutata positivamente. Nello specifico, viene assegnato un punteggio da 0 (insufficiente) a 5 (ottimo) al seguente aspetto dello sviluppo del progetto:

CRITERIO	DESCRIZIONE
Grado di partecipazione alla comunità	Presenza di domande e risposte significative sui canali di comunicazione offerti dal corso, richieste di ricevimento e delucidazioni, bug fixing del materiale messo a disposizione dal docente.

N.B. A scanso di equivoci, il plagio *non* rientra tra i casi di collaborazione valutati favorevolmente.

4.4 Voto di progetto e voto finale

I punteggi relativi a ciascuno degli aspetti descritti in questa sezione vengono sommati per ottenere un punteggio di base che va da 0 a 35. Terminata la discussione, a questo punteggio si somma il punteggio individuale ottenuto da ciascun membro del gruppo. Il risultato è un punteggio da 0 a 40 per ogni membro, che corrisponde al suo voto individuale di progetto. Un punteggio pari o superiore a 31/40 corrisponde a 30 e Lode.

Il voto finale del corso viene calcolato a partire dal voto di progetto e dal voto dello scritto. Per maggiori dettagli sulla modalità di calcolo e di verbalizzazione del voto finale, fare riferimento alla pagina web del Prof. Sangiorgi: `http://www.cs.unibo.it/~sangio/`.