

# Specifiche per il Progetto di Laboratorio di Sistemi Operativi

Tutor: Matteo Trentin - [matteo.trentin2@unibo.it](mailto:matteo.trentin2@unibo.it)

Anno Accademico 2022 - 2023

Questo documento contiene le specifiche del progetto del corso di Sistemi Operativi, per la Laurea in Informatica per il Management dell'Università di Bologna, anno accademico 2022-2023.

Eventuali modifiche o note sulle specifiche verranno comunicate tramite la bacheca ufficiale del corso su Virtuale.

In qualsiasi momento è possibile prenotare un ricevimento con il tutor in presenza o su Teams; il ricevimento va prenotato via email, contattandomi all'indirizzo [matteo.trentin2@unibo.it](mailto:matteo.trentin2@unibo.it).

## 1 Gruppi

I gruppi devono essere costituiti da tre (3) o quattro (4) persone. Gli studenti intenzionati a sostenere l'esame devono comunicare entro il **31 maggio 2023** la composizione del gruppo via email. Questa deve essere inviata dall'indirizzo istituzionale ([@studio.unibo.it](mailto:@studio.unibo.it)).

La mail deve avere come oggetto **[LABSO] FORMAZIONE GRUPPO** e contenere:

- Il nome del gruppo
- Per ogni componente: nome, cognome, numero di matricola
- Un indirizzo email di riferimento a cui inviare notifiche. È responsabilità del referente trasmettere le comunicazioni al resto del gruppo.

To	matteo.trentin2@unibo.it
Cc	
[LABSO] FORMAZIONE GRUPPO	
NOME GRUPPO	
Mario Rossi - 0007654321	
John Doe - 0001234567	
Jimi Hendrix - 0007898223	
mario.rossi@studio.unibo.it	

Esempio di email di formazione gruppo

Chi non riuscisse a trovare un gruppo può inviare una mail con oggetto **[LABSO] CERCO GRUPPO**, specificando:

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

To	matteo.trentin2@unibo.it
Cc	
[LABSO] CERCO GRUPPO	
Mario Rossi - 0007654321	
A causa di lavoro full-time (allego autocertificazione) sono disponibile solo la sera e nei weekend.	

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.

## 1.1 Repository

Il codice sorgente dei progetti dovrà essere caricato come repository su GitHub<sup>1</sup> o GitLab<sup>2</sup>.

Il repository deve essere **privato** e deve avere il nome LABSO\_<NOME GRUPPO>.

Sul repository va caricato **il codice del progetto**; non sono accettati repository contenenti, ad esempio, un file .zip con al suo interno il progetto. Sul repository può essere caricata anche la documentazione del progetto, ma non è obbligatorio.

### 1.1.1 GitHub

1. Ogni membro del gruppo crea un account su GitHub (a meno che non ne abbia già uno).
2. Il referente del gruppo crea un nuovo progetto cliccando su “+” → “**New repository**” nella barra superiore della schermata principale di GitHub. Inserisce LABSO\_<NOME\_GRUPPO> come nome del progetto, imposta il repository come privato e clicca su “**Create repository**”.
3. Il referente aggiunge ogni membro del gruppo al repository. Per fare ciò, dal menu del repository seleziona “**Settings**” → “**Collaborators**” e in seguito clicca su “**Add people**”. Nella schermata di invito membri, il referente cerca ciascun membro col nome utente con cui quest’ultimo è iscritto a GitHub e clicca su “**Add <username> to this repository**”.

---

<sup>1</sup><https://github.com/>

<sup>2</sup><https://gitlab.com>

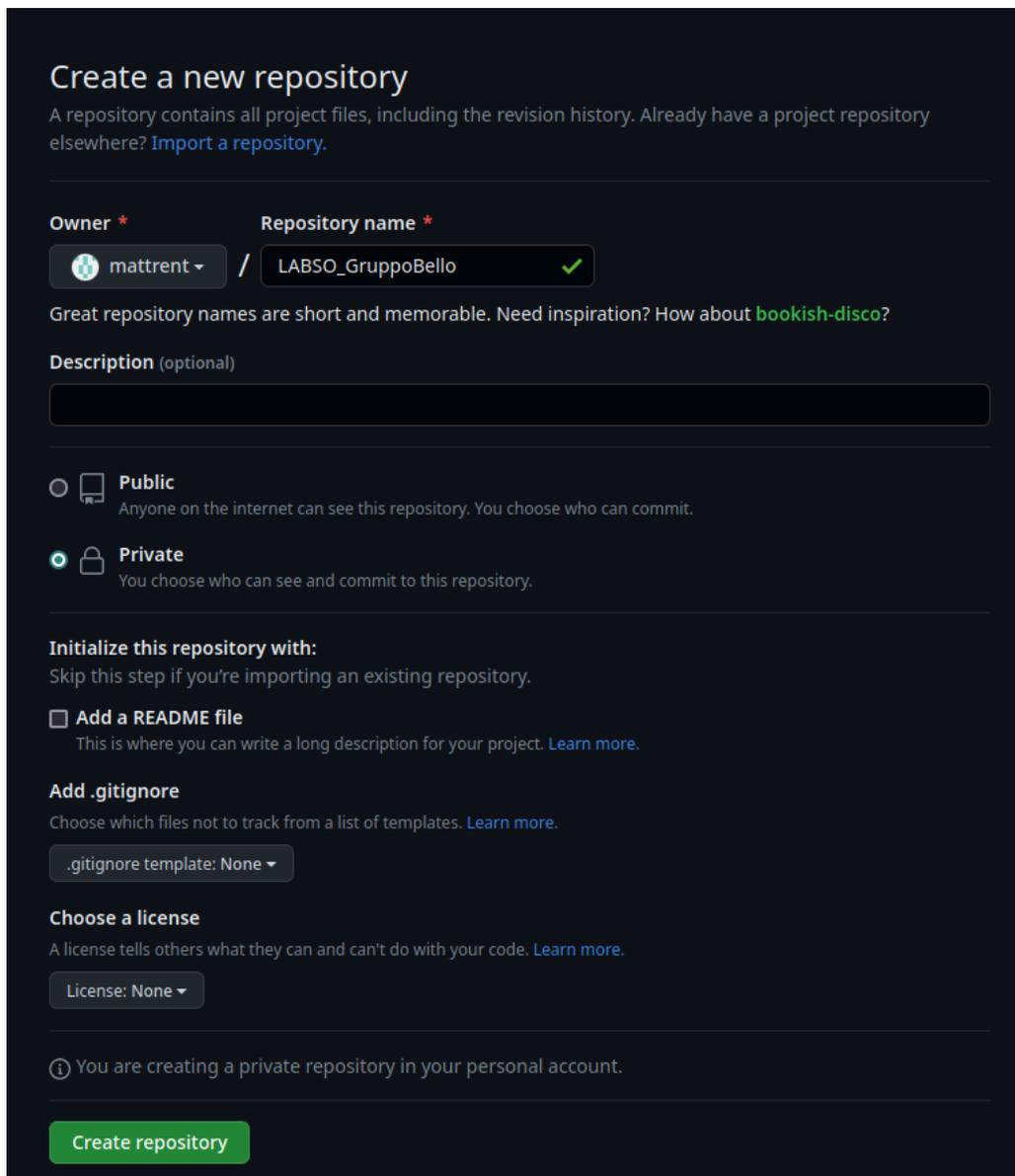
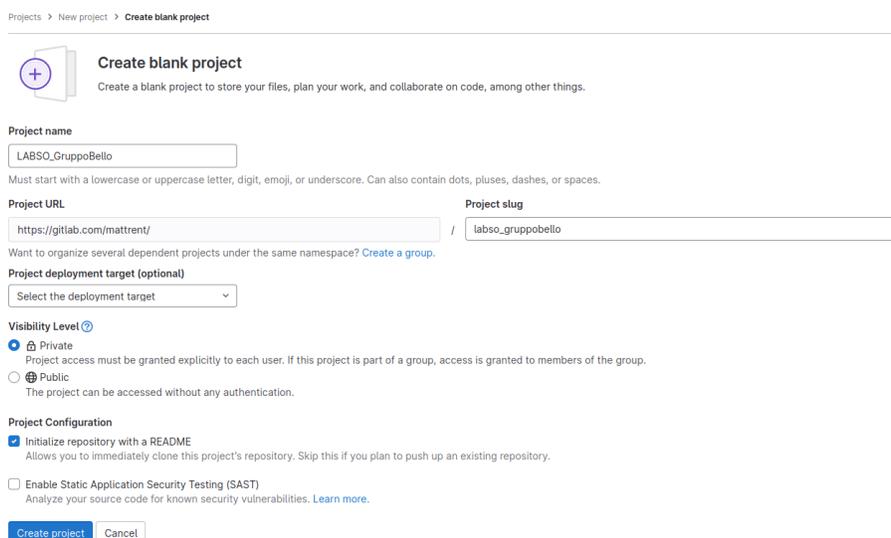


Figure 1: Schermata di creazione progetto su GitHub

### 1.1.2 GitLab

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 il repository come privato e clicca su “**Create project**”.
3. Il referente aggiunge ogni membro del gruppo al repository. Per fare ciò, dal menu del 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**”.



Projects > New project > Create blank project

### Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**  / **Project slug**

Want to organize several dependent projects under the same namespace? [Create a group](#).

**Project deployment target (optional)**

Select the deployment target

**Visibility Level** [?](#)

Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

Public  
The project can be accessed without any authentication.

**Project Configuration**

Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more](#).

Figure 2: Schermata di creazione progetto su GitLab

## 2 Progetto

### 2.1 Funzionalità

Il progetto richiede di implementare un semplice servizio per la gestione di trasferimenti di denaro tra conti bancari.

Il servizio permette a più client di connettersi a un singolo server, e di spostare somme di denaro arbitrarie tra due conti a scelta; nello specifico, gli utenti possono:

- Aprire nuovi conti
- Effettuare transazioni di denaro tra un conto e l'altro
- Aprire sessioni interattive in cui spostare denaro tra un conto e l'altro
- Richiedere la lista dei conti aperti, incluso il nome e l'ultima transazione eseguita

Ogni conto ha un nome univoco, definito alla sua creazione e utilizzato per indicarlo nelle transazioni.

Un conto può essere parte solo di una transazione alla volta. Quando un utente sta trasferendo denaro da un conto all'altro, nessuno può effettuare un trasferimento che coinvolga uno dei due conti. Una sessione interattiva viene considerata come una transazione in atto, finché non viene esplicitamente chiusa.

Nel caso in cui un utente provi ad effettuare una transazione su uno (o due) conti già occupati, la richiesta **non termina con un errore**, ma rimane in attesa finché entrambi i conti coinvolti non si sono liberati.

## 2.2 Requisiti implementativi

### 2.2.1 Linguaggio

Il progetto deve essere implementato in Java (ultima versione LTS: Java SE 17). La comunicazione di rete è implementata attraverso i socket. Il progetto deve essere diviso in un'applicazione Client e un'applicazione Server, che implementino le funzionalità descritte nella sezione [2.1](#).

### 2.2.2 Client

Il client viene avviato da linea di comando e richiede come parametri l'indirizzo IP e la porta del server a cui connettersi, ad esempio:

```
java Client 127.0.0.1 9000
```

Se il server non è raggiungibile all'indirizzo e alla porta specificati, il comando restituisce un messaggio di errore. Se la connessione va a buon fine, il client rimane in attesa delle istruzioni dell'utente.

I comandi richiesti per il client sono i seguenti:

- Il comando `list` restituisce un elenco di tutti i conti presenti. Ogni voce dell'elenco deve contenere:
  - Il nome del conto
  - Il bilancio del conto
  - I dati dell'ultima transazione, cioè: data, ammontare di denaro, e mittente (o destinatario) della transazione
- Il comando `open` crea un nuovo conto sul server. Il comando richiede due argomenti: `<Account>` e `<Amount>`, rispettivamente il nome del conto che sta venendo creato, e il bilancio iniziale del conto. Ad esempio:

```
open ExampleAccount 1000
```

Se un conto con lo stesso nome esiste già, il comando restituisce un messaggio di errore.

- Il comando `transfer` effettua una transazione di denaro da un conto all'altro. Il comando richiede tre argomenti: `<Amount>`, `<Account1>` e `<Account2>`, rispettivamente l'ammontare di denaro da trasferire, il conto mittente e il conto destinatario. Ad esempio:

```
transfer 500 ExampleAccount1 ExampleAccount2
```

Per semplicità, si può imporre che `Amount` sia sempre positivo. Se la transazione è effettuabile (ovvero se entrambi i conti esistono e se il mittente contiene denaro a sufficienza) allora l'ammontare specificato viene prelevato da `Account1` e depositato in `Account2`; in caso contrario, il comando restituisce un errore.

- Il comando `transfer_i` è un modo alternativo per trasferire denaro tra due conti. Il comando richiede due argomenti: `<Account1>` e `<Account2>`, rispettivamente il conto mittente e destinatario della transazione. Se entrambi i conti esistono, il comando avvia una *sessione di transazione* tra i due conti; quando il client si trova in questa sessione, sono disponibili i seguenti comandi:
  - Il comando `:move` trasferisce denaro da `Account1` ad `Account2`; richiede un argomento, `<Amount>`, che rappresenta la quantità di denaro spostata. Si può di nuovo assumere che `Amount` sia positivo. Se il bilancio di `Account1` non è sufficiente a sostenere la transazione, il comando restituisce un errore, ma la sessione **non viene chiusa**.
  - Il comando `:end` termina la sessione di transazione interattiva, riportando il client in modalità normale e liberando i due conti.

Quando il client è in sessione di transazione, i comandi standard sono disabilitati fino al suo termine.

Un esempio di utilizzo di questo comando e della sessione di transazione:

```
transfer_i ExampleAccount1 ExampleAccount2
> :move 500
...
> :end
```

### 2.2.3 Server

Il server viene avviato da linea di comando, e accetta come unico parametro la porta su cui restare in ascolto, ad esempio:

```
java Server 9000
```

Il server accetta solo il seguente comando:

- `quit`, disconnette tutti i client ancora connessi e successivamente termina il server.

### 2.2.4 Requisito opzionale (1 Punto)

Implementare il comando `close <Account>`, che rimuove un conto corrente dal server, o restituisce un errore se il conto è inesistente. L'implementazione di questo comando **non è necessaria** per ottenere il voto massimo (30L).

## 2.3 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.

### 2.3.1 Struttura della documentazione

L'intestazione della documentazione deve avere titolo "Laboratorio di Sistemi Operativi A.A. 2022-23" 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.

### 2.3.2 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 vengono gestite le transazioni, etc. Quello su cui la documentazione deve concentrarsi è *cosa* accade quando arriva una richiesta, *come* sono gestite le transazioni, etc. nella *vostra particolare implementazione* delle specifiche.

## 3 Consegna del progetto

Al momento della consegna, occorre creare un tag di nome **Consegna** all’interno del proprio repository. Le modalità di creazione del tag sono definite ai seguenti indirizzi:

- Per progetti caricati su **GitHub**: <https://docs.github.com/en/repositories/releasing-projects-on-github/managing-releases-in-a-repository>, nella sezione “**Creating a release**”. La release sarà anch’essa chiamata **Consegna**.
- Per progetti caricati su **GitLab**: <https://docs.gitlab.com/ee/user/project/repository/tags/>, nella sezione “**Create a tag**”. È essenziale aggiungere un messaggio qualsiasi *non vuoto* per annotare il tag con la data e l’ora della creazione.

Il tutor dovrà essere aggiunto come Collaborator (se il progetto è su GitHub) o Reporter (se il progetto è su GitLab) del repository, seguendo la stessa modalità descritta in sezione 1.1. Questa operazione può anche essere

svolta subito prima della consegna, ma è necessaria per rendere accessibile il codice.

Una volta fatto ciò, notificare la consegna al tutor inviando un'email a `matteo.trentin2@unibo.it` con oggetto **[LABSO] CONSEGNA < NOME GRUPPO >**. La mail dovrà includere:

- L'indirizzo del repository
- In allegato, un file PDF di nome `DOCUMENTAZIONE.pdf`, contenente la documentazione del progetto

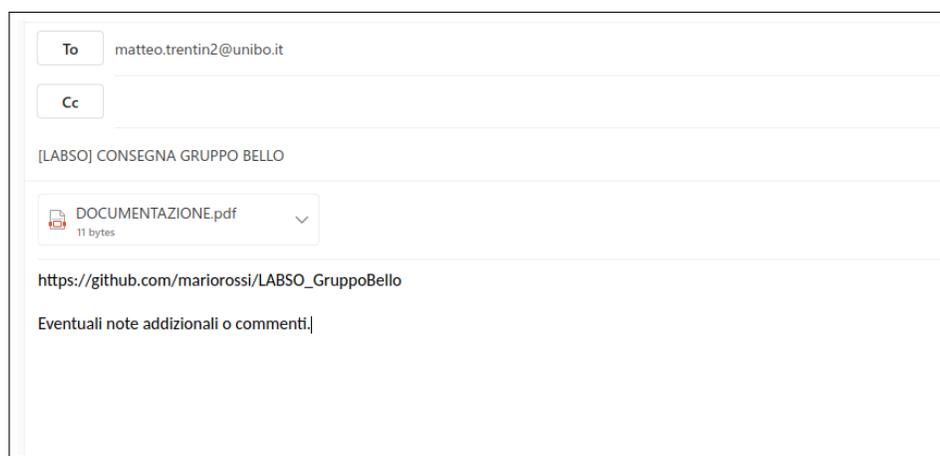


Figure 3: Esempio di email di notifica di consegna

Sono presenti tre appelli per la consegna del progetto, con le rispettive scadenze:

- Lunedì **26 Giugno 2023** (26/06/2023), ore 23:59.
- Lunedì **25 Settembre 2023** (25/09/2023), ore 23:59.
- Lunedì **27 Novembre 2023** (27/11/2023), ore 23:59.

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.**

## 4 Discussione

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 5.3) per determinare il voto finale in trentesimi di ogni singolo membro del gruppo.

## 5 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.

### 5.1 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 6 (ottimo).

CRITERIO	DESCRIZIONE
Rispetto delle specifiche	Il progetto implementa correttamente le funzionalità descritte in Sezione 2.1 e rispetta i requisiti di Sezione 2.2.
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.

## 5.2 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 6 (ottimo).

<b>CRITERIO</b>	<b>DESCRIZIONE</b>
<b>Qualità dell'informazione</b>	La documentazione fornisce una descrizione chiara e completa del progetto implementato e del processo di implementazione.
<b>Uso di esempi</b>	Presenza di esempi (narrativi, grafici, etc.) utili alla comprensione delle scelte implementative del gruppo o di scenari d'uso specifici.
<b>Analisi delle scelte implementative</b>	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.

### 5.3 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 36; a questo viene poi sommato 1 punto per l'eventuale implementazione del requisito opzionale (2.2.4). 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 42 per ogni membro, che corrisponde al suo voto individuale di progetto. Un punteggio pari o superiore a 31/42 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/~sangiorgi/>.