Linee guida di progetto

PSM_Project – Password Strength Meter Engine

Corso di Ingegneria del Software – A.A. 2025/2026
Docente: Prof.ssa Marina Mongiello
Team: Belviso M., Vegliante G., Didonna A.

23 novembre 2025

1 Scopo del documento

Questo documento raccoglie le indicazioni operative per sviluppare e consegnare il progetto. Questo file è stato aggiornato in data 21/12/25. Le linee guida servono a mantenere coerenza tra requisiti, modellazione UML, implementazione e valutazione sperimentale, evitando scelte non tracciate o difficili da verificare.

2 Obiettivo e perimetro del sistema

Il progetto prevede un sistema in grado di stimare *in tempo reale* la robustezza di una password e di fornire feedback utile all'utente durante la digitazione. L'obiettivo è andare oltre le sole regole di composizione (ad esempio "metti una maiuscola e un simbolo"), evidenziando anche pattern deboli e scelte prevedibili.

Funzionalità attese

- calcolo di un punteggio e/o livello di robustezza aggiornato a ogni modifica della password;
- feedback comprensibile e immediato (indicatori visivi e suggerimenti pratici);
- supporto a una fase di valutazione sperimentale (confronto con baseline o metriche note).

3 Organizzazione della repository

La repository ufficiale del progetto `PSM_Project` è organizzata in due macro-aree:

- **`src/`**: codice del prototipo e dei moduli software (UI, engine, API, sperimentazione);
- **`docs/`**: documentazione di progetto (specifiche, UML, architettura, relazione, presentazione e risultati).

Questa separazione va mantenuta: ogni nuovo artefatto deve essere collocato nella cartella coerente, così da rendere la verifica rapida e ripetibile.

4 Deliverable e consegna

4.1 Deliverable finali

Alla consegna finale il team deve fornire:

1. **Relazione tecnica**, includendo requisiti (anche non funzionali), principali diagrammi UML, scelte implementative, strategia di test e risultati sperimentali.
2. **Presentazione** (slide) per l'esposizione orale, con una demo e sintesi dei risultati.
3. **Repository** completa e ordinata, con istruzioni chiare per eseguire e verificare il progetto.

4.2 Riproducibilità

Il progetto deve essere eseguibile e verificabile senza passaggi "impliciti".

- Il file `README.md` in radice deve spiegare come eseguire la demo e come trovare documentazione e UML.
- La documentazione (diagrammi, relazione, risultati) deve essere archiviata in `docs/` con struttura coerente.
- **Esecuzione containerizzata (opzionale ma consigliata)**: se prevista o richiesta nel contesto di consegna, deve essere supportata da un `Dockerfile` in radice e comandi riportati nel README.

5 Analisi dei requisiti (FURPS+)

I requisiti vanno raccolti in modo verificabile e non ambiguo, seguendo FURPS+:

- **F – Functionality**: valutazione real-time, suggerimenti, eventuale export/analisi dei risultati.
- **U – Usability**: feedback chiaro, leggibile e coerente con l'intento formativo del meter.
- **R – Reliability**: determinismo (stesso input $\rightarrow$ stesso output), gestione errori.
- **P – Performance**: latenza trascurabile rispetto alla digitazione; efficienza su dataset.
- **S – Supportability**: facilità di manutenzione, estensione e portabilità.
- **+**: vincoli aggiuntivi (privacy, sicurezza dei dati, limiti del contesto didattico).

6 Modellazione UML

La modellazione è parte integrante del progetto e deve restare coerente con l'implementazione:

- **Casi d'uso**: attori, obiettivi e confini del sistema; includere scenari principali e alternativi.
- **Sequenza**: interazione tra UI, engine e moduli (ad es. valutazione, logging, export) con input/output ed errori.
- **Classi e architettura**: struttura modulare (UI, engine, API, sperimentazione) e dipendenze essenziali.

7 Linee guida per l'engine di valutazione

7.1 Requisiti minimi

L'engine deve:

- valutare la password a ogni modifica (real-time);
- produrre un output consistente (punteggio e livello) con soglie documentate;
- individuare pattern deboli ricorrenti (dizionari, ripetizioni, sequenze, informazioni personali quando disponibili);
- essere **separabile dalla UI** (modulo riusabile o interfaccia chiara).

7.2 Comportamento atteso e criteri pratici

Il meter deve combinare **bonus** e **penalità** in modo spiegabile:

- aumentare il punteggio al crescere della lunghezza e della varietà di caratteri;
- ridurre il punteggio in presenza di sequenze, ripetizioni o pattern prevedibili;
- applicare penalità significative per parole comuni/dizionari e per riferimenti facilmente intuibili (ad esempio nomi propri o elementi di "pop culture");
- **quando necessario, imporre cap al punteggio** in presenza di pattern critici (es. password corte o contenenti pattern completi estremamente prevedibili), mantenendo però l'algoritmo coerente e motivato in relazione.

In relazione, va spiegato *perché* un certo pattern è trattato come critico e *come* incide sul punteggio.

7.3 Requisiti non funzionali specifici

- **Usabilità**: messaggi brevi, comprensibili, non contraddittori.
- **Prestazioni**: calcolo rapido, senza blocchi percepibili.
- **Affidabilità**: risultati ripetibili e assenza di comportamenti casuali.
- **Estendibilità**: regole, dizionari e metriche aggiungibili senza riscrivere l'intero sistema.

8 Testing e verifiche

Il testing va pianificato e tracciato:

- **Unit test**: casi mirati sull'engine (input $\rightarrow$ output atteso).
- **Integration test**: flusso end-to-end dalla digitazione al feedback.
- **Regressione**: un set minimo di test per evitare peggioramenti dopo modifiche alle regole.

9 Valutazione sperimentale

La sperimentazione serve a verificare coerenza e utilità del meter.

9.1 Obiettivi

- distinguere password deboli/forti in modo sensato e stabile;
- confrontare risultati con baseline selezionate e discutere differenze;
- analizzare casi limite (sovrastime, false sicurezze) e motivare le scelte correttive.

9.2 Struttura consigliata

1. scelta di una o più baseline (misuratori o metriche note);
2. definizione dataset (reali o sintetici), con attenzione a privacy e tracciabilità;
3. esecuzione degli esperimenti con parametri dichiarati (soglie, dizionari, opzioni);
4. report finale con tabelle/grafici, commento critico e minacce alla validità.

10 Chiusura

Le linee guida mirano a rendere il progetto verificabile e coerente. Ogni modifica sostanziale (soglie, penalità, cap, dataset, metriche) deve essere esplicitata e motivata, così da preservare tracciabilità e qualità del lavoro di gruppo.

Riferimenti bibliografici

[1] B. Ur et al. *How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation.* USENIX Security Symposium, 2012.

[2] D. Wang et al. *No Single Silver Bullet: Measuring the Accuracy of Password Strength Meters.* USENIX Security Symposium, 2023.