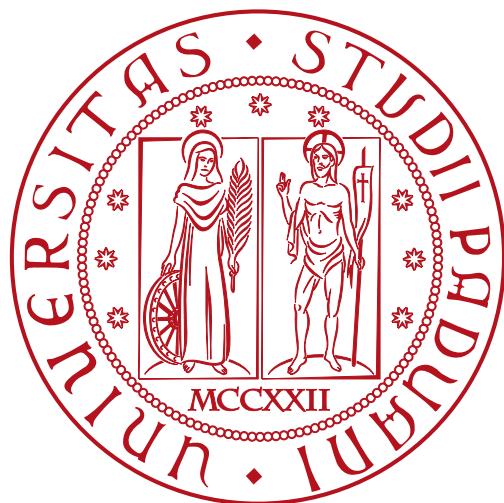


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



WebApp per attività laboratoriali di OpenDay

Tesi di laurea

25 Luglio 2025

Relatrice

Prof.ssa Ombretta Gaggi

Laureando

Orlando Virgilio Maria Ferazzani

Matricola 2058653

A Nonno Enzo, che mi ha insegnato a vivere.

Ai miei genitori, che mi hanno reso l'uomo che sono.

Al mare, che mi ha insegnato il rispetto, perché egli non perdonava.

Tenetevi il mondo e lasciatemi il mare

Sommario

Il seguente elaborato descrive l'attività di tirocinio, della durata complessiva di 313 (trecen-totredici) ore, svolta presso l'Università di Padova. Questa attività è stata portata avanti sotto la guida della Prof.ssa Ombretta Gaggi. Il Prof. Claudio Palazzi ha ricoperto il ruolo di tutor accademico.

L'*Università degli Studi di Padova* durante i suoi OpenDay, utilizza [WebApp](#) interattive per avvicinare i ragazzi delle scuole superiori al corso di laurea in Informatica. Queste consentono di far conoscere le basi della programmazione attraverso giochi che stimolano la logica e la creatività. Tuttavia, queste applicazioni sono spesso molto tediose da utilizzare dato il breve tempo a disposizione per le suddette attività e non sempre riescono a coinvolgere gli studenti, soprattutto chi di programmazione non ha mai intrapreso degli studi di alcun genere.

Il tirocino effettuato mira proprio a risolvere questa problematica, sviluppando una *WebApp* interattiva che permetta di avvicinare i ragazzi al mondo della programmazione in modo divertente e stimolante.

Ringraziamenti

ringraziamento qua

Padova, Luglio 2025

Orlando Virgilio Maria Ferazzani

Indice

Introduzione	2
1.1 Motivazioni e Contesto	2
1.2 Strumenti e processi	3
1.2.1 Suddivisione del lavoro	3
1.3 Struttura del Documento	5
Scopo del tirocinio	7
2.1 Scopo del progetto	7
2.2 Obiettivi prefissati	8
2.3 Prodotti attesi	10
Analisi dei Requisiti	11
3.1 Obiettivo del progetto	11
3.2 Utenti Target	12
3.3 Casi d'uso	12
3.4 Attori	13
3.4.1 Primari	13
3.4.2 Secondari	13
3.5 Definizione dei casi d'uso Utente target	14
3.5.1 UC01: Registrazione Utente (Figura 3.1)	14
3.5.1.1 UC1.1: Visualizzazione messaggio di errore se il nome utente non è rispettoso	16
3.5.1.2 UC1.2: Visualizzazione messaggio di errore se il nome utente è già utilizzato	16

3.5.1.3 UC1.3: Visualizzazione messaggio di errore se non sono stati compilati tutti i campi	17
3.5.2 UC02: Visualizzazione UI (Figura 3.2)	18
3.5.2.1 UC2.1: Visualizzazione messaggio di errore generico	19
3.5.3 UC03: Visualizzazione pagina iniziale laboratorio (Figura 3.3)	20
3.5.3.1 UC3.1: Visualizzazione messaggio di errore se il caricamento dello step corrente non è andato a buon fine	21
3.5.4 UC04: Visualizzazione step progressivi (Figura 3.4)	22
3.5.4.1 UC4.1: Visualizzazione Step 1	24
3.5.4.2 UC4.1.1: Drag and Drop	24
3.5.4.3 UC4.2: Visualizzazione Step 2	25
3.5.4.4 UC4.3: Visualizzazione Step 3	26
3.5.4.5 UC4.3.1: Risposta domande	27
3.5.4.6 UC4.3.1.1: Visualizzazione suggerimento	28
3.5.4.7 UC4.4: Visualizzazione Step 4	28
3.5.4.8 UC04.4.1: Completamento esercizio con menù a tendina	29
3.5.4.9 UC4.5: Visualizzazione Step 5	30
3.5.4.10 UC4.6: Visualizzazione Step 6	31
3.5.5 UC05: Visualizzazione pagina di chiusura laboratorio (Figura 3.5)	33
3.6 Definizione dei casi d'uso Utente Admin	34
3.6.1 UC06: Login Admin (Figura 3.6)	35
3.6.1.1 UC6.1: Visualizzazione messaggio di errore se le credenziali sono errate	36
3.6.2 UC07: Visualizzazione homepage Admin (Figura 3.7)	37
3.6.2.1 UC07.1: Visualizzazione scheda lista utenti registrati	38

3.6.2.2	UC7.1.1: Visualizzazione messaggio per lista vuota	39
3.6.2.3	UC07.2: Visualizzazione tab gestione laboratorio .	40
3.6.2.4	UC07.2.1: Visualizzazione pulsanti della gestione step del laboratorio	40
3.6.2.5	UC07.2.1.1: Visualizzazione pulsante per l'avanzamento ad un nuovo step	41
3.6.2.6	UC07.2.1.2: Visualizzazione pulsante per ritornare allo step precedente	42
3.6.2.7	UC07.2.1.3: Visualizzazione pulsante per azzeramento statistiche	43
3.6.2.8	UC07.2.1.4: Visualizzazione pulsante per il reset degli step	44
3.6.2.9	UC07.2.2: Visualizzazione Tab Grafici Step 3	45
3.6.2.10	UC7.2.2.1: Visualizzazione messaggio di informazione se non ci sono risposte salvate o avviene un errore generico	46
3.6.2.11	UC07.2.3: Visualizzazione Tab Grafici Step 4	47
3.7	Requisiti	48
3.8	Requisiti funzionali	49
3.9	Requisiti di qualità	50
3.10	Requisiti di vincolo	51
Implementazione		53
4.1	Tecnologie: scelte e implementazione	54
4.1.1	TypeScript	55
4.1.2	Axios	56
4.1.3	Firebase	59
4.1.4	GitHub	61
4.1.5	ShadCN	62
4.1.6	NextJS	65

4.1.6.1	Layout.tsx	65
4.1.6.2	page.tsx	67
4.1.6.3	global-error.tsx	67
4.1.6.4	Server e Client components	69
4.1.7	TailwindCSS	69
4.1.8	LucideReact	71
4.1.9	DND Kit	72
4.2	Implementazione del codice e risultati	73
4.2.1	Registrazione	73
4.2.2	Pagina Admin	76
4.2.3	Laboratorio	79
4.2.4	Suggerimento	81
4.2.5	Layout	82
4.2.6	UserProvider	86
4.2.7	Avatar	87
4.2.7.1	Banner Home	89
4.2.8	Responsive Design	90
4.2.9	Drag and Drop	91
4.2.10	Tema chiaro e scuro	93
4.3	Accessibilità	93
4.4	Test e compatibilità	95
4.4.0.1	Risultato finale dei test	96
Conclusioni		98
5.1	Lo scopo e le scelte	98
5.2	Copertura dei requisiti	99
5.3	Sviluppi futuri	100
5.4	Valutazioni personali	100
Glossario		102
Bibliografia		106

Elenco delle Figure

Figura 1.1	vista della Kanban alla seconda settimana	4
Figura 1.2	vista diagramma di Gantt alla seconda settimana	4
Figura 3.1	UC01: diagramma UML	14
Figura 3.2	UC02: Visualizzazione UI	18
Figura 3.3	UC03: Visualizzazione pagina iniziale laboratorio	20
Figura 3.4	UC04: Visualizzazione step progressivi	22
Figura 3.5	UC05: Visualizzazione pagina di chiusura laboratorio	33
Figura 3.6	Schermata di accesso Admin	35
Figura 3.7	Visualizzazione homepage Admin	37
Figura 4.1	Logo di typescript	55
Figura 4.2	Logo di Axios	56
Figura 4.3	Variabili per le richieste Axios	57
Figura 4.4	Controllo del token GitHub	57
Figura 4.5	Istanza base di Axios	58
Figura 4.6	Logo di Firebase	59
Figura 4.7	File `firebase.ts`	60
Figura 4.8	Logo di GitHub	62
Figura 4.9	Esempio di ThemeProvider di ShadCN	63
Figura 4.10	Logo di ShadCN	64
Figura 4.11	Logo di NextJS	65
Figura 4.12	Struttura del file Layout	66
Figura 4.13	global-error.tsx di Thinky	68
Figura 4.14	Pagina di errore globale di Thinky	68
Figura 4.15	Logo di TailwindCSS	70
Figura 4.16	Logo di LucideReact	71

Figura 4.17 Esempio di utilizzo di LucideReact che renderizza una freccia verso l'alto	72
Figura 4.18 Logo di DND Kit	73
Figura 4.19 Tooltip di Thinky	73
Figura 4.20 Form di registrazione di Thinky	74
Figura 4.21 Messaggio di errore per username già utilizzato	74
Figura 4.22 Messaggio di errore per username non rispettoso	75
Figura 4.23 Messaggio di errore per campi mancanti	76
Figura 4.24 Pagina admin di Thinky	76
Figura 4.25 Form di login per la pagina admin	77
Figura 4.26 Messaggio di errore per password errata	77
Figura 4.27 Tab Lista Utenti	78
Figura 4.28 Tab Gestione Laboratorio	78
Figura 4.29 Hint di Thinky	81
Figura 4.30 layout.tsx principale	82
Figura 4.31 layout.tsx della pagina di laboratorio	83
Figura 4.32 layout.tsx delle altre pagine	84
Figura 4.33 UserProvider	86
Figura 4.34 Avatar con username generato: Elefante Verde	87
Figura 4.35 Avatar con username scelto: Avatar personalizzato	88
Figura 4.36 Dropdown dell'avatar	88
Figura 4.37 Banner della Home con username	90
Figura 4.38 Banner della pagina Produttore-Consumatore	90
Figura 4.39 Esempio di Drag and Drop	91
Figura 4.40 SortableItem	92
Figura 4.41 Dizionario delle classi	93

Elenco delle Tabelle

Tabella 3.1 Requisiti funzionali	49
Tabella 3.2 Requisiti di qualità	50
Tabella 3.3 Requisiti di vincolo	51

Elenco del codice

Codice 4.1 Esempio di file ` .env`	57
Codice 4.2 Richiesta GET per recuperare i dati degli utenti	58
Codice 4.3 Utilizzo di `onSnapshot()` per aggiornare i dati in tempo reale	61
Codice 4.4 Esempio di layout.tsx con ThemeProvider	64
Codice 4.5 HTML Esempio	70
Codice 4.6 Esempio di codice CSS per HTML	70
Codice 4.7 Esempio di codice TailwindCSS per HTML	71

Capitolo 1

Introduzione

1.1 Motivazioni e Contesto

Il corso di laurea triennale in *Informatica* che offre l'Ateneo dell'*Università Di Padova* è a numero chiuso, con un massimo di 220 posti, cifra che nel corso degli anni è aumentata visto il grande interesse da parte di studenti delle superiori di intraprendere questo percorso. Tuttavia, le scuole di provenienza delle matricole mostrano una grande disparità, con il numero degli studenti provenienti da istituti ad indirizzo informatico (come istituti tenici o liceo delle scienze applicate) che supera di gran lunga il numero di studenti provenienti da licei tradizionali. In particolare, parliamo di circa il 75% di studenti che proviene da istituti tecnici, con il restante 25% proveniente da altre scuole.

Questa disparità è data sicuramente dalla «credenza» che l'informatica sia solo programmazione e che quindi uno studente proveniente da un Liceo Scientifico non abbia le competenze necessarie per questo affrontare questo percorso.

Tuttavia, di contro a questa «credenza», è ovvio che non mancano le competenze, ma solamente le conoscenze, che è proprio il vuoto che questo corso va a colmare. Se magari uno studente proveniente da un istituto tecnico ha già delle competenze di programmazione, uno studente proveniente da un liceo scientifico ha sicuramente delle competenze matematiche e logiche che sono fondamentali per affrontare questo percorso.

Altro obiettivo molto importante, è quello di aumentare il numero di iscritti di genere femminile, dimostrando, grazie anche a vari testimonial ([Wome-](#)

[nInCS](#)) alle varie studentesse, che la storia dell'informatica è stata fatta in non piccola parte da donne e scenziate.

L'informatica non è solo programmazione, ma è anche progettazione e design. Infatti, il mio progetto si basa sulla creazione di un'applicazione web interattiva, che richiede competenze di progettazione e design oltre a quelle di programmazione.

Personalmente, essendo uno di quei ragazzi timorosi di iniziare il percorso, arrivando da un Liceo Scientifico, mi sono sentito moralmente in dovere di aiutare tutti coloro che si trovano nella posizione in cui io stesso mi sono trovato, anche per questo quindi ho deciso di fare questo tirocinio con la Prof.ssa Ombretta Gaggi.

1.2 Strumenti e processi

Durante il corso del tirocinio, mi sono avvalso di diversi strumenti che ho imparato ad utilizzare nel corso della mia carriera universitaria, e che mi hanno aiutato a portare avanti il mio progetto, come [Git](#) G e [GitHub](#) G, utilissimi per tenere traccia di ogni modifica effettuata al codice sorgente dell'applicazione, nonché per la condivisione di tale codice con la mia relatrice. In ogni caso, tutte le tecnologie saranno discusse nel dettaglio [quarto capitolo](#).

1.2.1 Suddivisione del lavoro

Dovendo presentare un [Piano di Lavoro](#) G per iniziare il mio tirocino, e volendo rispettare gli insegnamenti appresi dal corso di Ingegneria del Software, ho da subito deciso di impostare il mio Way of Working.

Ho quindi dapprima definito tutti gli obiettivi da raggiungere durante il percorso, trovandone 19. A questo punto, ho suddiviso il lavoro da svolgere nelle 8 settimane, potendo quindi definire degli « [sprint](#) G» » . A fine di ogni sprint, controllo di aver completato tutto ciò che mi ero prefissato nel [backlog](#) G di lavoro e, se ci fosse qualcosa che non ho completato, lo sposto nel backlog del prossimo sprint.

Per il tracciamento delle task da completare, ho utilizzato una [Kanban](#) board(1.1), divisa in 3 colonne:

1. ToDo: attività da completare
2. In Progress: attività in corso
3. Done: attività completate

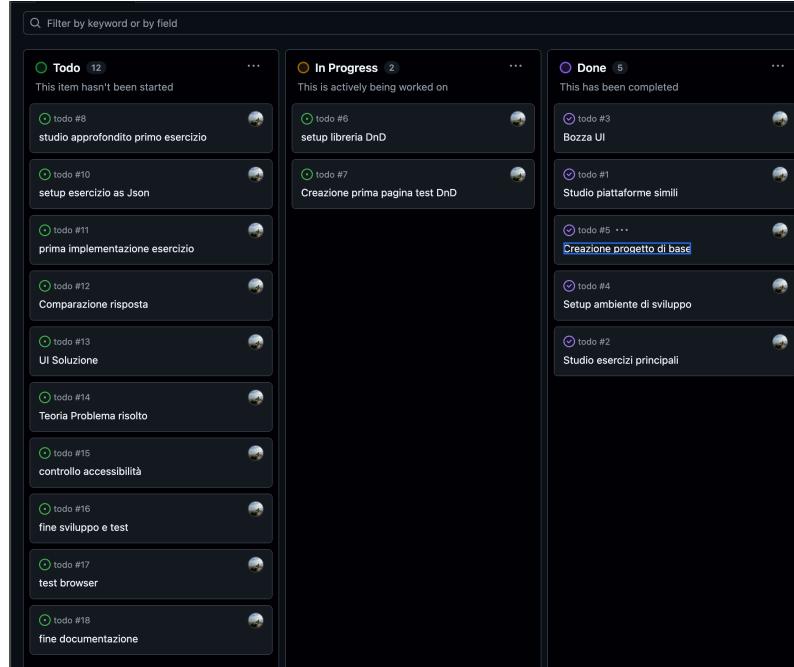


Figura 1.1: vista della Kanban alla seconda settimana

Invece, per visualizzare i tempi di svolgimento previsti ed effettivi di tali task, ho deciso di utilizzare un diagramma di Gantt (1.2).

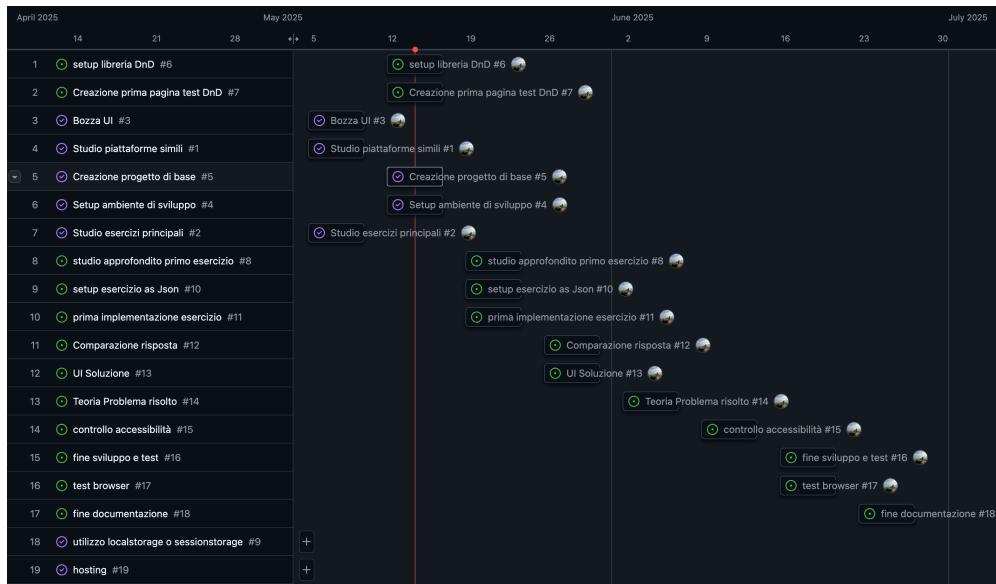


Figura 1.2: vista diagramma di Gantt alla seconda settimana

Tutto questo è conforme al metodo di lavoro [Scrum](#), che prevede una suddivisione del lavoro in sprint e un monitoraggio costante dei progressi. Lo Scrum fa parte della metodologia Agile⁽¹⁾, creata per migliorare lo sviluppo di prodotti software rallentati dalle tediote fasi di analisi e documentazione.

Dato che il progetto è stato svolto in singolo, mi sono impegnato a mantenere un ritmo di lavoro costante, e di fornire aggiornamenti settimanali alla Prof.ssa Ombretta Gaggi, via email o incontri di persona, per discutere i progressi, le difficoltà incontrate e risolvere eventuali problemi prima di andare eccessivamente fuori strada.

1.3 Struttura del Documento

Il secondo capitolo, 'Scopo del Tirocinio' fornisce una panoramica di tutto il progetto, dal suo scopo, a i prodotti attesi, descrivendo nel dettaglio il concetto di webapp interattiva ed elencando gli obiettivi di partenza.

Il terzo capitolo: 'Analisi dei Requisiti' fornisce una panoramica dei requisiti del progetto, partendo dai requisiti funzionali e non funzionali, passando per i vincoli e le limitazioni, fino ad arrivare ai casi d'uso e alle user stories.

Il [quarto capitolo: 'Implementazione'](#) espone come le scelte descritte nel capitolo precedente sono state implementate e perché sono state scelte determinate tecnologie. Inoltre, viene fornita una panoramica del codice sorgente, con i file più significativi e le loro funzionalità, oltre che la descrizione delle caratteristiche di [accessibilità](#) G.

Il [quarto capitolo: 'Conclusioni'](#) fornisce una panoramica dei risultati ottenuti, sia a livello di codice sorgente che di accessibilità. Inoltre, viene fornita una panoramica dei test effettuati e dei risultati ottenuti.

Nel [Glossario](#) sono riportati i termini tecnici e le abbreviazioni utilizzate nel corso del documento.

Oltre alla struttura qui sopra descritta, si adottano anche i seguenti accorgimenti tipografici:

- le abbreviazioni, termini tecnici (o comunque di uso non comune), o in lingua straniera in prima occorrenza nel documento sono definiti nel glossario consultabile alla fine del documento. Ogni termine nel glossario è evidenziato come segue: [Parola](#)G.
- Altri termini che richiedono un'attenzione particolare, ma che non hanno bisogno di essere definiti, saranno evidenziati in corsivo: *Parola*.

Capitolo 2

Scopo del tirocinio

Il prossimo capitolo fornisce una panoramica dettagliata del progetto di stage, partendo dal suo scopo fino ai prodotti attesi.

2.1 Scopo del progetto

Come evidenziato nel capitolo precedente, questo progetto di stage è volto a colmare una grande “disparità” tra gli studenti che si iscrivono al corso di laurea in *Informatica dell’Università di Padova*.

Possiamo comunque distinguere due “sotto-scopi” che convogliano poi in nel progetto finale:

Il primo scopo, definibile *scopo tecnico*, e rivolto al laureando, è quello di portare a limite le proprie abilità, competenze e conoscenze cercando di creare un prodotto che non solo sia funzionale, ma anche facile da utilizzare, accessibile a tutti e che aiuti effettivamente gli studenti. Questo scopo si traduce, all’atto pratico, nello sviluppo della WebApp interattiva.

Il secondo scopo, definibile *scopo sociale*, è rivolto agli studenti delle scuole superiori, e ha come obiettivo quello di mostrare loro che l’informatica non è solo programmazione, ma anche progettazione e design, e che non importa quale sia il loro *background* scolastico per affrontare questo percorso. In particolare, si vuole dimostrare che ogni scuola superiore, anche se in maniera e con contenuti diversi, permette di affrontare questo percorso, chi in un modo e chi in un altro.

Se dovessi dare quindi una definizione sintetica, quasi matematica come il mio corso ha insegnato, dello scopo del progetto, direi che è:

Si definisce il seguente progetto di sviluppo di una WebApp interattiva per l'orientamento agli Open Day di Informatica il processo che, dato un insieme eterogeneo di studenti delle scuole superiori, realizza uno strumento digitale accessibile e inclusivo, volto a ridurre le disparità di background e di genere e a mostrare che l'informatica è disciplina aperta a tutti, indipendentemente dal percorso scolastico precedente.

2.2 Obiettivi prefissati

Nelle fasi iniziali di quella che possiamo chiamare “candidatura” al progetto, ho definito ciò che ritenevo fossero gli obiettivi principali del progetto, sapendo che questi comunque sarebbero stati mutevoli nel tempo, essendo aggiornati man mano che il progetto andava avanti sulla base delle esigenze e sull’effettivo progresso del progetto. Gli obiettivi definiti sono poi stati discussi insieme alla mia relatrice, e raggruppati in tre diverse categorie:

- **Obiettivi Obbligatori:** sono gli obiettivi che devono essere raggiunti per considerare il progetto completato e soddisfacente. Questi obiettivi sono stati definiti in modo da garantire che il prodotto finale sia funzionale, accessibile e utile per gli studenti delle scuole superiori.
- **Obiettivi Desiderabili:** sono obiettivi che, se raggiunti, migliorano significativamente il progetto e lo rendono più completo e interessante. Questi obiettivi possono essere considerati come “aggiunte” che arricchiscono l’esperienza utente o aggiungono funzionalità utili, ma non sono strettamente necessari per il completamento del progetto.
- **Obiettivi Facoltativi:** sono obiettivi che, se raggiunti, arricchiscono il progetto e lo rendono più completo e interessante. Questi obiettivi possono essere considerati come “extra” che migliorano l’esperienza utente o aggiungono funzionalità utili.

Per la classificazione degli obiettivi è stata adottata la seguente nomenclatura:

- Obbligatorio: O
- Desiderabile: D

- Facoltativo: F

Con il codice identificativo dell'obiettivo che diventa quindi:

[X] [Y]

con X che rappresenta la categoria dell'obiettivo e Y che rappresenta il numero progressivo dell'obiettivo.

Gli obiettivi definiti inizialmente sono stati i seguenti:

Obbligatori

- O01: implementazione dell'applicativo;
- O02: funzionamento base dello stesso;
- O03: primo problema: produttore - consumatore;
- O04: simulazione della sequenza di operazione eseguita come live demo;
- O05: misurazione delle performance del punto O04;

Desiderabili

- D01: implementare il problema dei filosofi a cena;
- D02: possibilità di inserire una sequenza di operazioni;

Facoltativi

- F01: inserimento di uno username per mostrarlo nella scoreboard (no login);
- F02: inserimento di un punteggio in base alla percentuale di performance;
- F03: visualizzazione di una scoreboard di tutti gli utenti;

Come anticipato, questi obiettivi sono stati in parte rivisti in corso d'opera insieme alla Prof.ssa Ombretta Gaggi, e sono stati aggiunti altri obiettivi, che sono stati classificati come segue:

Obbligatori

- O01: implementazione dell'applicativo;
- O02: funzionamento base dello stesso;
- O03: primo problema: lettore - scrittore;

- O04: Inserimento di username e scuola di provenienza per tenere traccia dell'utilizzo dell'applicativo;

Facoltativi

- F01: implementare il problema del produttore - consumatore;
- F02: possibilità di inserire una sequenza di operazioni;
- F03: Visualizzazione delle statistiche di risposte ai quesiti posti agli utenti;

Desiderabili

- D01: Creazione di un'area riservata per l'amministratore senza necessità di login particolari;
- D02: visualizzazione di una dashboard con tutti gli utenti;

2.3 Prodotti attesi

Al completamento del progetto, è prevista la consegna di un prodotto funzionante che rispetti i requisiti definiti in fase di progettazione.

Sarà consegnato il codice sorgente del progetto, che sarà ben documentato e commentato, in modo da permettere una facile comprensione e manutenzione del codice stesso. Inoltre, sarà fornita una documentazione tecnica che descrive le funzionalità implementate, le tecnologie utilizzate e le modalità di utilizzo del prodotto, oltre che un breve documento che spiega all'utente come utilizzare l'applicativo, e un documento che spiega ad eventuali futuri programmatore come è strutturato il codice, per guiderli in un eventuale manutenzione o potenziamento del progetto.

Oltre che alla documentazione tecnica e al codice sorgente, sarà prodotto questo documenti di tesi, che descrive appunto il progetto sotto ogni punto di vista e ne analizza i risultati ottenuti, le difficoltà incontrate e le soluzioni adottate.

Capitolo 3

Analisi dei Requisiti

Il seguente capitolo riporta l'analisi dei requisiti di Thinky, con i relativi casi d'uso e diagrammi UML.

L'analisi dei requisiti è una fase, e di conseguenza, un documento (o in questo caso, capitolo), fondamentale nel ciclo di vita di un progetto.

Il suo scopo è quello di definire in modo chiaro, preciso e dettagliato le funzionalità che il prodotto finale andrà ad offrire, ossia i requisiti obbligatori e opzionali richiesti dal [proponente G](#).

Nello specifico, questo capitolo si propone di:

- Fornire un'analisi basata direttamente sulle richieste del proponente, in particolare, si basa sugli obiettivi riscontrati negli incontro con la Prof.ssa Ombretta Gaggi, che sono stati riportati nel capitolo precedente.
- Identificare i requisiti e suddividerli in funzionali e non funzionali.
- Validare e verificare i requisiti rispettando la Way of Working adottata.

3.1 Obiettivo del progetto

Thinky è una WebApp progettata per aiutare gli studenti che sono intenzionati ad iscriversi al corso di laurea in *Informatica* dell'*Università di Padova* a approcciarsi al mondo della programmazione durante gli Open Day, mediante un laboratorio interattivo svolto insieme alla Prof.ssa Ombretta Gaggi e al Prof. Claudio Palazzi.

L'app utilizza diversi strumenti per fornire all'utente un'esperienza interattiva e coinvolgente, senza esagerare nella difficoltà, sia di utilizzo che dell'attività da svolgere.

3.2 Utenti Target

Gli utenti target di Thinky sono principalmente studenti delle scuole superiori che sono interessati a iscriversi al corso di laurea in *Informatica* dell'*Università di Padova*. In particolare, si rivolge a:

- Ragazzi e Ragazze di età compresa tra i 17 e i 19 anni, che stanno per diplomarsi e sono interessati a intraprendere un percorso di studi in informatica;
- Studenti provenienti da istituti tecnici e professionali, che potrebbero avere una formazione di base in informatica;
- Studenti provenienti da licei scientifici e classici, che potrebbero avere una formazione più analitico-matematica ma sono interessati a esplorare il mondo della programmazione e dello sviluppo software;

3.3 Casi d'uso

Nella seguente sezione verranno elencati i casi d'uso principali della WebApp, che sono stati identificati in base agli obiettivi del progetto e alle esigenze degli utenti target.

Tutti i casi d'uso adottano la medesima struttura, come segue:

- **Nome Caso D'uso:** Nome del caso d'uso chiaro e descrittivo;
- **Attori:** Gli attori coinvolti nel caso d'uso, che possono essere utenti o sistemi esterni;
- **Precondizione e postcondizione:** Stato del sistema prima e dopo l'esecuzione del caso d'uso;
- **Scenario principale:** descrizione dettagliata dei passi che l'utente deve seguire per completare il caso d'uso;
- **Estensioni:** Eventuali estensioni o variazioni dello scenario principale, che possono includere errori o situazioni particolari che l'utente potrebbe incontrare durante l'esecuzione del caso d'uso;
- **Inclusioni:** Eventuali casi d'uso inclusi che sono necessari per completare il caso d'uso principale;

- **Generalizzazioni:** Eventuali generalizzazioni del caso d'uso, che possono includere casi d'uso più specifici o varianti del caso d'uso principale.
- **User Story:** Una breve descrizione del caso d'uso in forma di user story, che può essere utilizzata per comunicare il caso d'uso in modo più semplice e comprensibile.

È possibile suddividere i casi d'uso del sistema Thinky in due categorie principali:

- I casi d'uso relativi all'utente target, che riguardano le funzionalità principali della WebApp e l'interazione con l'utente;
- I casi d'uso relativi all'utente Admin;

3.4 Attori

3.4.1 Primari

Per attore primario si intende un attore che interagisce direttamente con il sistema per raggiungere un obiettivo specifico. Nel caso di Thinky, l'attore primario è l'utente, ossia lo studente che partecipa al laboratorio interattivo. L'utente ha accesso a tutte le funzionalità della WebApp e può interagire con il sistema per completare le attività previste dal laboratorio.

3.4.2 Secondari

Per attore secondario si intende tutti quei servizi o sistemi esterni che Thinky utilizza a supporto delle sue funzionalità. Sono tutti attori su cui non si ha effettuato alcun tipo di sviluppo e che vengono «contattati» dal sistema, invece di contattare il sistema stesso. Nel caso di Thinky, gli attori secondari sono:

- [Firebase](#), servizio di backend as service che fornisce un database in tempo reale;
- [GitHub](#), piattaforma di hosting per progetti software che utilizza Git come sistema di controllo versione;

3.5 Definizione dei casi d'uso Utente target

3.5.1 UC01: Registrazione Utente (Figura 3.1)

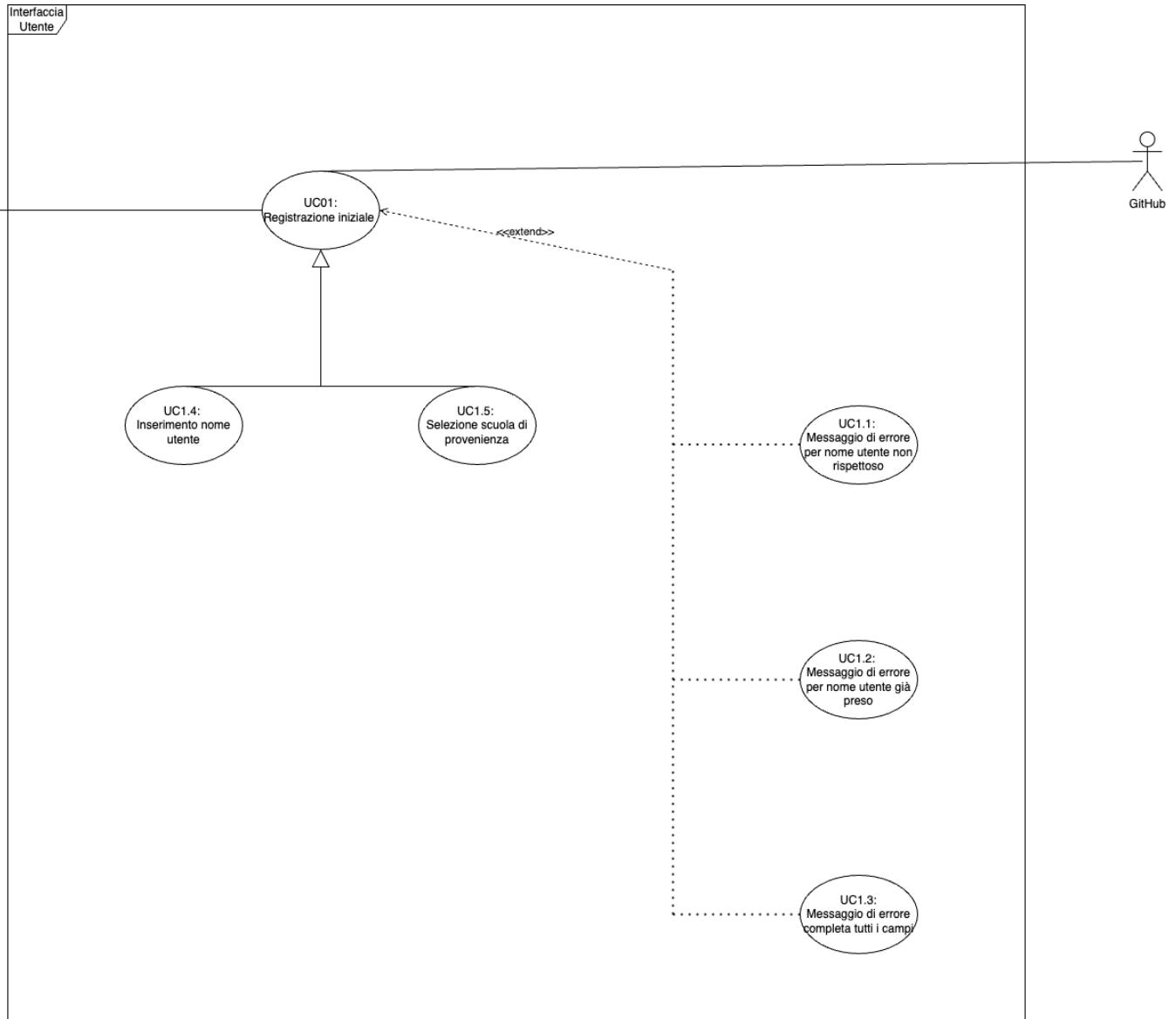


Figura 3.1: UC01: diagramma UML

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente non è registrato nel sistema.
- Le [API G](#) di GitHub sono disponibili e funzionanti.

- **Postcondizioni:**

- L'utente è registrato e può accedere alle funzionalità della WebApp.

Scenario principale

- L'utente apre per la prima volta l'interfaccia di Thinky.
- Il sistema rileva che l'utente non è registrato e mostra il form di registrazione.
- L'utente inserisce un nome utente e seleziona la scuola di provenienza.
- Il sistema verifica la validità dei dati inseriti (vedi estensioni UC1.1 - UC1.3).
- Se i dati sono validi, l'utente viene registrato e il nome viene salvato su GitHub.
- L'utente viene reindirizzato alla homepage della WebApp.

Estensioni

- UC1.1: Visualizzazione messaggio di errore se il nome utente non è rispettoso.
- UC1.2: Visualizzazione messaggio di errore se il nome utente è già stato utilizzato.
- UC1.3: Visualizzazione messaggio di errore se non sono stati compilati tutti i campi.

Generalizzazioni

- UC1.4: Inserimento nome utente.
- UC1.5: Selezione scuola di provenienza.

User story

- Come utente, voglio potermi registrare al sistema per accedere alle funzionalità, senza fornire dati sensibili.

UC1.1: Visualizzazione messaggio di errore se il nome utente non è rispettoso

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente ha inserito un nome utente non rispettoso.
 - Le API di GitHub sono disponibili e configurate correttamente.
- **Postcondizioni:**
 - Il sistema visualizza un messaggio di errore.

Scenario principale

- L'utente inserisce un nome utente non rispettoso.
- Il sistema visualizza un messaggio di errore.
- L'utente corregge il nome utente e ripete la registrazione.

User story

- Quando inserisco un nome utente non rispettoso, il sistema mostra un messaggio di errore così posso correggerlo e ripetere la registrazione.

UC1.2: Visualizzazione messaggio di errore se il nome utente è già utilizzato

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - Il nome utente inserito è già utilizzato da un altro utente.

- Le API di GitHub sono disponibili e configurate correttamente.
- **Postcondizioni:**
 - Il sistema visualizza un messaggio di errore.

Scenario principale

- L'utente inserisce un nome utente già utilizzato.
- Il sistema visualizza un messaggio di errore.
- L'utente corregge il nome utente e ripete la registrazione.

User story

- Quando inserisco un nome utente già utilizzato, il sistema mostra un messaggio di errore così posso correggerlo e ripetere la registrazione.

UC1.3: Visualizzazione messaggio di errore se non sono stati compilati tutti i campi

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente ha inserito un nome utente ma non ha compilato il campo scuola di provenienza.
 - Le API di GitHub sono disponibili e configurate correttamente.
- **Postcondizioni:**
 - Il sistema visualizza un messaggio di errore.

Scenario principale

- L'utente inserisce un nome utente ma lascia vuoto il campo scuola di provenienza.
- Il sistema visualizza un messaggio di errore.
- L'utente completa tutti i campi e ripete la registrazione.

User story

- Quando non compilo tutti i campi richiesti, il sistema mi avvisa con un messaggio di errore così posso completare i dati mancanti.

3.5.2 UC02: Visualizzazione UI (Figura 3.2)

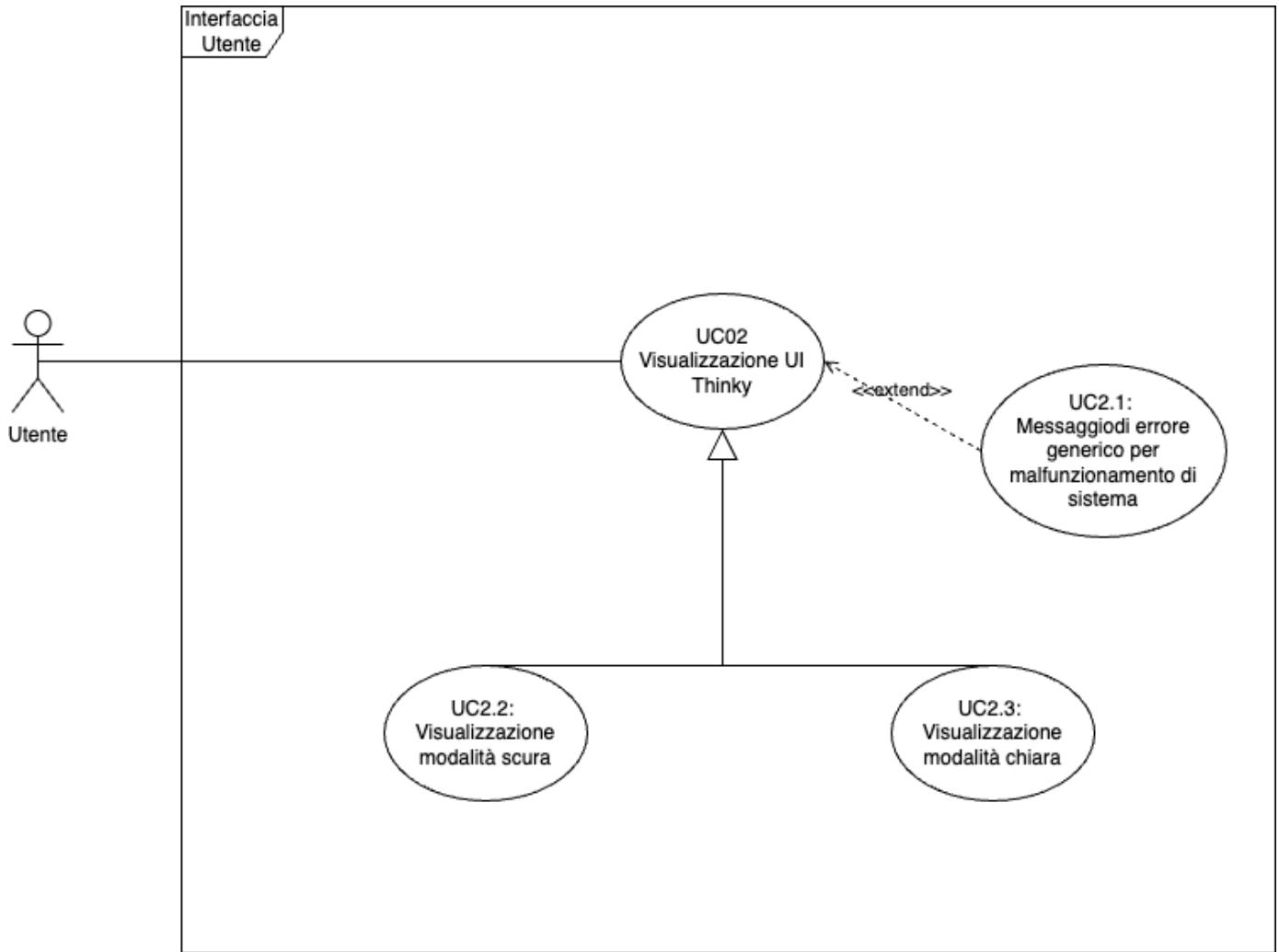


Figura 3.2: UC02: Visualizzazione UI

Attori coinvolti

- **Attori Primari:** Utente

Precondizioni e Postcondizioni

- **Precondizioni:**
 - L'utente è registrato nel sistema.

- Il sistema è connesso e funzionante.
- **Postcondizioni:**
 - L'utente visualizza l'interfaccia utente della WebApp.

Scenario principale

- L'utente accede alla WebApp.
- Il sistema visualizza l'interfaccia utente.

Estensioni

- UC2.1: Visualizzazione messaggio di errore generico in caso di malfunzionamento del sistema.

Generalizzazioni

- UC2.2: Visualizzazione interfaccia in modalità scura.
- UC2.3: Visualizzazione interfaccia in modalità chiara.

User story

- Come utente, voglio visualizzare l'interfaccia della WebApp per interagire con essa e scegliere la modalità del tema.

UC2.1: Visualizzazione messaggio di errore generico

Attori coinvolti

- **Attori Primari:** Utente

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - Si è verificato un malfunzionamento.
- **Postcondizioni:**
 - Il sistema visualizza un messaggio di errore generico.

Scenario principale

- L'utente accede alla WebApp.
- Si verifica un malfunzionamento.

- Il sistema mostra un messaggio di errore generico.

User story

- In caso di malfunzionamento, il sistema mostra un messaggio di errore generico così l'utente è informato e può riprovare.

3.5.3 UC03: Visualizzazione pagina iniziale laboratorio (Figura 3.3)

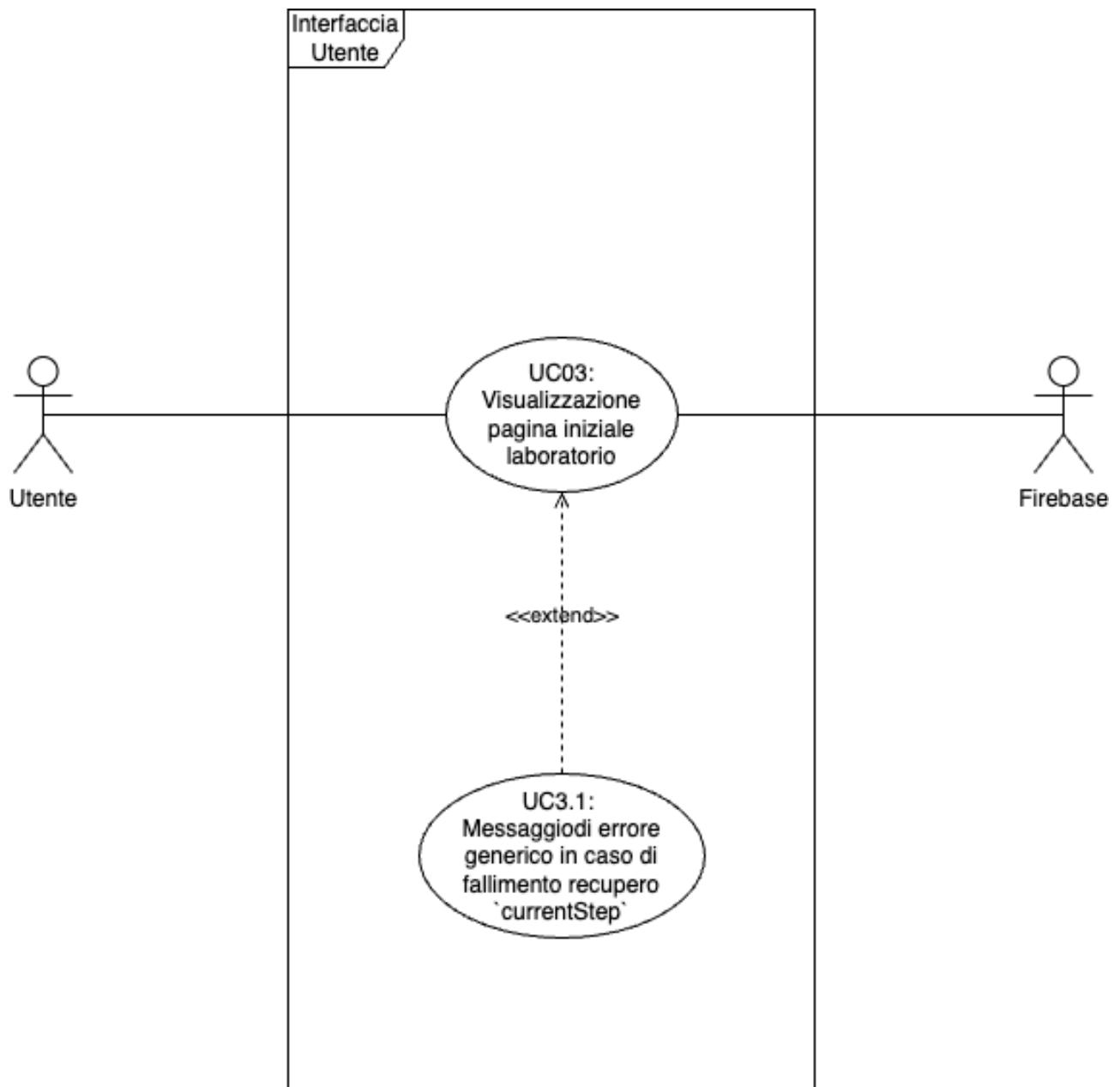


Figura 3.3: UC03: Visualizzazione pagina iniziale laboratorio

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

• Precondizioni:

- Il sistema è connesso e funzionante.
- Le API di Firebase sono disponibili e configurate correttamente.
- L'utente è registrato.
- L'utente ha cliccato sul pulsante «Vai al laboratorio».

• Postcondizioni:

- L'utente visualizza la pagina iniziale del laboratorio.

Scenario principale

- L'utente clicca sul pulsante «Vai al laboratorio».
- Il sistema carica la pagina iniziale del laboratorio.

Estensioni

- UC3.1: Visualizzazione messaggio di errore se il caricamento dello step corrente fallisce.

User story

- Come utente, voglio visualizzare la pagina iniziale del laboratorio per iniziare l'attività.

UC3.1: Visualizzazione messaggio di errore se il caricamento dello step corrente non è andato a buon fine

Attori coinvolti

- **Attori Primari**
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

• Precondizioni:

- L'utente è registrato.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema è connesso ma la richiesta dello step corrente fallisce.
- **Postcondizioni:**
 - Il sistema mostra un messaggio di errore.

Scenario principale

- L'interfaccia utente invia una richiesta API a Firebase per la variabile dello step corrente, ma la richiesta fallisce.

User story

- Se il caricamento dello step corrente fallisce, il sistema mostra un messaggio di errore per informare l'utente e consentirgli di riprovare.

3.5.4 UC04: Visualizzazione step progressivi (Figura 3.4)

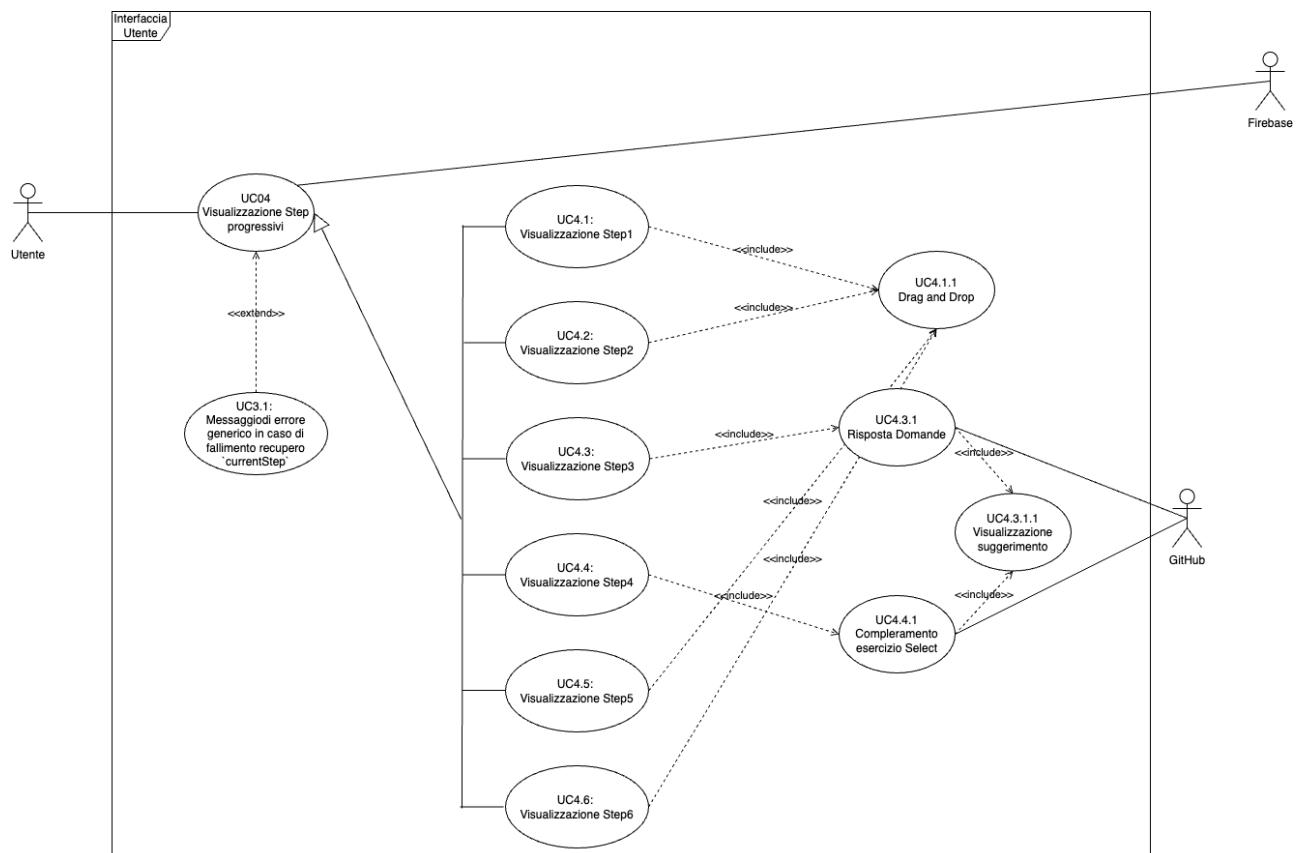


Figura 3.4: UC04: Visualizzazione step progressivi

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

• Precondizioni:

- Il sistema è connesso e funzionante.
- Le API di Firebase sono disponibili e configurate correttamente.
- L'utente è registrato.
- L'utente ha visualizzato la pagina iniziale del laboratorio.
- L'utente admin ha correttamente avanzato lo step.
- Il sistema ha caricato lo step corrente correttamente.

• Postcondizioni:

- L'utente visualizza lo step progressivo del laboratorio.

Scenario principale

- L'utente attende che l'admin carichi lo step successivo.
- Il sistema visualizza lo step progressivo del laboratorio.
- L'utente può interagire con esso.

Estensioni

- UC3.1: Visualizzazione messaggio di errore se il caricamento dello step corrente fallisce.

Generalizzazioni

- UC4.1: Visualizzazione Step 1
- UC4.2: Visualizzazione Step 2
- UC4.3: Visualizzazione Step 3
- UC4.4: Visualizzazione Step 4
- UC4.5: Visualizzazione Step 5
- UC4.6: Visualizzazione Step 6

User story

- Come utente, voglio visualizzare lo step progressivo del laboratorio per interagire con esso e completare l'attività.

UC4.1: Visualizzazione Step 1

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

• Precondizioni:

- Il sistema è connesso e funzionante.
- Le API di Firebase sono disponibili e configurate correttamente.
- L'utente è registrato.
- L'admin ha caricato lo step 1.
- Il sistema ha caricato lo step corrente correttamente.

• Postcondizioni:

- L'utente visualizza lo step 1.

Scenario principale

- L'utente attende che l'admin carichi lo step 1.
- Il sistema visualizza lo step 1.
- L'utente può interagire con esso e completare l'esercizio sul produttore con le istruzioni mancanti.

Inclusioni

- UC4.1.1: Drag and Drop

User story

- Come utente, voglio visualizzare lo step 1 del laboratorio per interagire con il Drag and Drop e completare l'esercizio sul Produttore con le istruzioni mancanti.

UC4.1.1: Drag and Drop

Attori coinvolti

- **Attori Primari:** Utente

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente è registrato.
- L'utente ha visualizzato lo step 1.

- **Postcondizioni:**

- L'utente può trascinare e rilasciare gli elementi dello step 1.

Scenario principale

- L'utente visualizza lo step 1.
- Il sistema permette il drag and drop degli elementi.
- L'utente interagisce con gli elementi trascinati e rilasciati.

User story

- Come utente, voglio poter trascinare e rilasciare gli elementi.

UC4.2: Visualizzazione Step 2

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- Le API di Firebase sono disponibili e configurate correttamente.
- L'utente è registrato.
- L'admin ha caricato lo step 2.
- Il sistema ha caricato lo step corrente correttamente.

- **Postcondizioni:**

- L'utente visualizza lo step 2.
- L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 2.
- Il sistema visualizza lo step 2.
- L'utente può interagire con esso e completare l'esercizio sul consumatore con le istruzioni mancanti.

Inclusioni

- UC4.1.1: Drag and Drop

User story

- Come utente, voglio visualizzare lo step 2 del laboratorio per interagire con il Drag and Drop e completare l'esercizio sul Consumatore con le istruzioni mancanti.

UC4.3: Visualizzazione Step 3

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'admin ha caricato lo step 3.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
- **Postcondizioni:**
 - L'utente visualizza lo step 3.
 - L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 3.
- Il sistema visualizza lo step 3.
- L'utente può interagire con esso.

Inclusioni

- UC4.3.1: Risposta domande

User Story

- Come utente, voglio visualizzare lo step 3 del laboratorio per rispondere alle le domande completando l'attività.

UC4.3.1: Risposta domande

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente ha visualizzato lo step 3.
 - Le API di GitHub sono disponibili e configurate correttamente.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
- **Postcondizioni:**
 - L'utente ha risposto alle domande dello step 3.
 - Il sistema ha salvato le risposte su GitHub.

Scenario principale

- L'utente visualizza lo step 3.
- Il sistema mostra le domande dello step 3.
- L'utente risponde alle domande.
- Il sistema salva le risposte su GitHub.

Inclusioni

- UC4.3.1.1: Visualizzazione suggerimento

User story

- Come utente, voglio rispondere alle domande di teoria riguardanti il problema del lettore-scrittore per completare l'attività.

UC04.3.1.1: Visualizzazione suggerimento

Attori coinvolti

- **Attori Primari:** Utente

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente ha visualizzato lo step 3.
- Il sistema ha caricato lo step corrente correttamente.
- L'utente ha cliccato sul pulsante «Suggerimento».

- **Postcondizioni:**

- Il sistema visualizza il suggerimento per rispondere alle domande dello step 3.

Scenario principale

- L'utente visualizza lo step 3.
- Il sistema mostra le domande dello step 3.
- L'utente clicca sul pulsante «Suggerimento».
- Il sistema visualizza il suggerimento per rispondere alle domande dello step 3.

User story

- Come utente, voglio visualizzare un suggerimento sulla domanda a cui sto rispondendo per completare l'attività.

UC4.4: Visualizzazione Step 4

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'admin ha caricato lo step 4.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema ha caricato lo step corrente correttamente.

- **Postcondizioni:**

- L'utente visualizza lo step 4.
- L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 4.
- Il sistema visualizza lo step 4.
- L'utente può interagire con esso e completare l'esercizio di teoria con i menù a tendina sul problema del lettore-scrittore.

Inclusioni

- UC4.4.1: Completamento esercizio dei lettori-scrittori con i menù a tendina.

User Story

- Come utente voglio visualizzare lo step 4 del laboratorio per completare l'esercizio sul problema dei lettori-scrittori con i menù a tendina e interagire con esso completando l'attività.

COMMENTO: lo spazio tra le parole è dato dall'evidenziatura in giallo, tolta quella la formattazione torna normale.

UC04.4.1: Completamento esercizio con menù a tendina

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.

- L'utente ha visualizzato lo step 4.
- Le API di GitHub sono disponibili e configurate correttamente.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema ha caricato lo step corrente correttamente.
- **Postcondizioni:**
 - L'utente ha completato l'esercizio Select dello step 4.
 - Il sistema ha salvato le risposte su GitHub.

Scenario principale

- L'utente visualizza lo step 4.
- Il sistema mostra l'esercizio Select dello step 4.
- L'utente completa l'esercizio Select.
- Il sistema salva le risposte su GitHub.

Inclusioni

- UC4.3.1.1: Visualizzazione suggerimento

User story

- Come utente, voglio poter interagire con i menù a tendina e completare l'attività sul problema dei lettori-scrittori.

UC4.5: Visualizzazione Step 5

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'admin ha caricato lo step 5.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
- **Postcondizioni:**
 - L'utente visualizza lo step 5.

- L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 5.
- Il sistema visualizza lo step 5.
- L'utente può interagire con esso.

Inclusioni

- UC4.1.1: Drag and Drop

User story

- Come utente, voglio visualizzare lo step 5 del laboratorio per interagire con il Drag and Drop e completare l'esercizio sul Lettore con le istruzioni mancanti.

UC4.6: Visualizzazione Step 6

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'admin ha caricato lo step 6.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
- **Postcondizioni:**
 - L'utente visualizza lo step 6.
 - L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 6.
- Il sistema visualizza lo step 6.
- L'utente può interagire con esso.

Inclusioni

- UC4.1.1: Drag and Drop

User story

- Come utente, voglio visualizzare lo step 1 del laboratorio per interagire con il Drag and Drop e completare l'esercizio sullo Scrittore con le istruzioni mancanti.

3.5.5 UC05: Visualizzazione pagina di chiusura laboratorio (Figura 3.5)

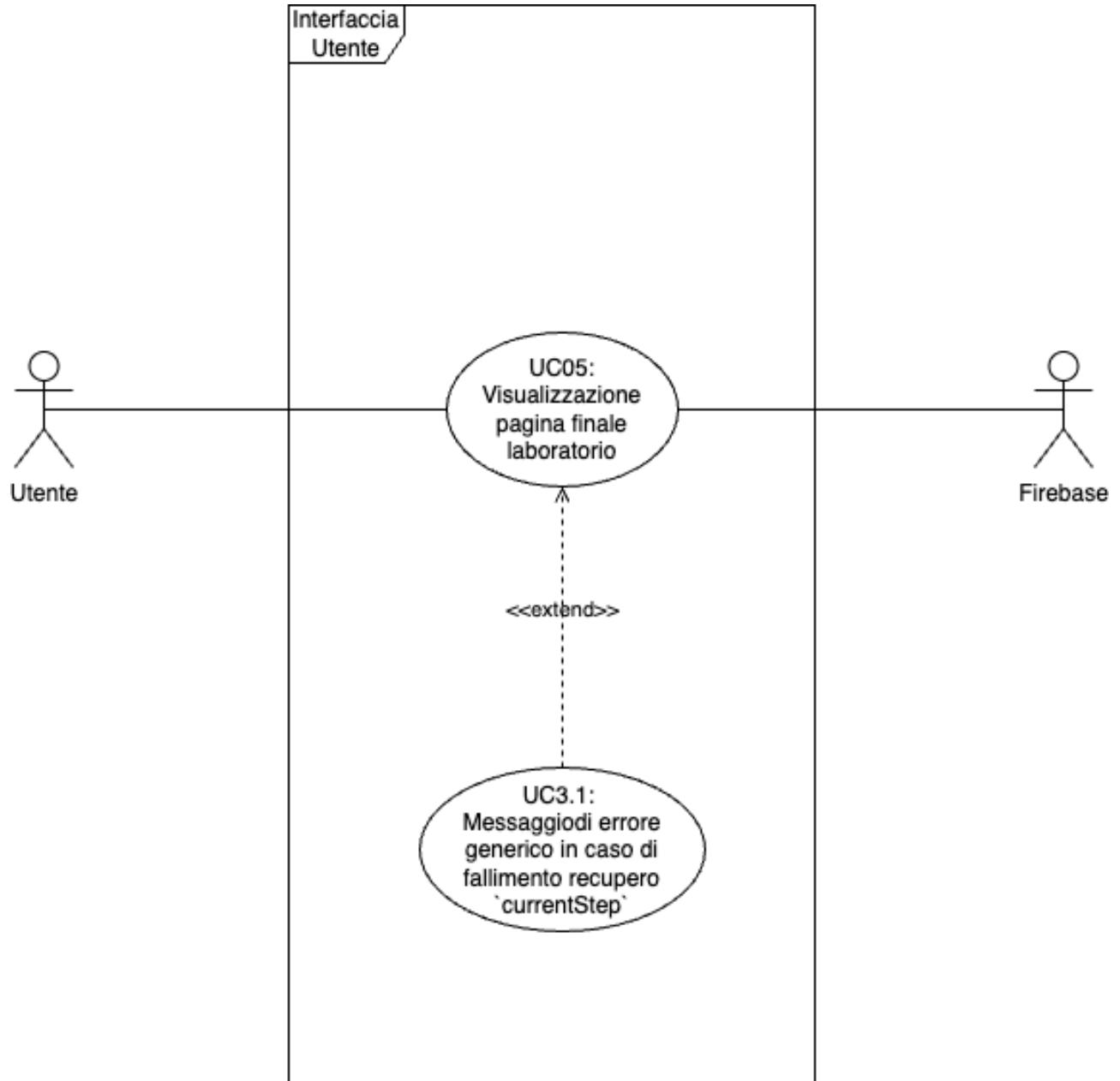


Figura 3.5: UC05: Visualizzazione pagina di chiusura laboratorio

Attori coinvolti

- **Attori Primari:** Utente
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'admin ha caricato lo step 7.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema ha caricato lo step corrente correttamente.

- **Postcondizioni:**

- L'utente visualizza lo step 7.
- L'utente può interagire con esso.

Scenario principale

- L'utente attende che l'admin carichi lo step 7.
- Il sistema visualizza lo step 7.
- L'utente può interagire con esso.

User Story

- Come utente, voglio visualizzare lo step 7 del laboratorio per finire l'attività laboratoriale.

3.6 Definizione dei casi d'uso Utente Admin

L'utente Admin è considerabile come una generalizzazione dell'utente target, in quanto ha accesso a tutte le funzionalità della WebApp, ma con privilegi aggiuntivi che gli consentono di gestire il sistema. L'utente Admin, tuttavia, non necessita di passare per la registrazione avendo già le credenziali (in questo caso salvate sul file `.env`). Si considerino quindi solo i seguenti casi d'uso, specifici per l'utente Admin, da sommare ai casi d'uso appena elencati.

3.6.1 UC06: Login Admin (Figura 3.6)

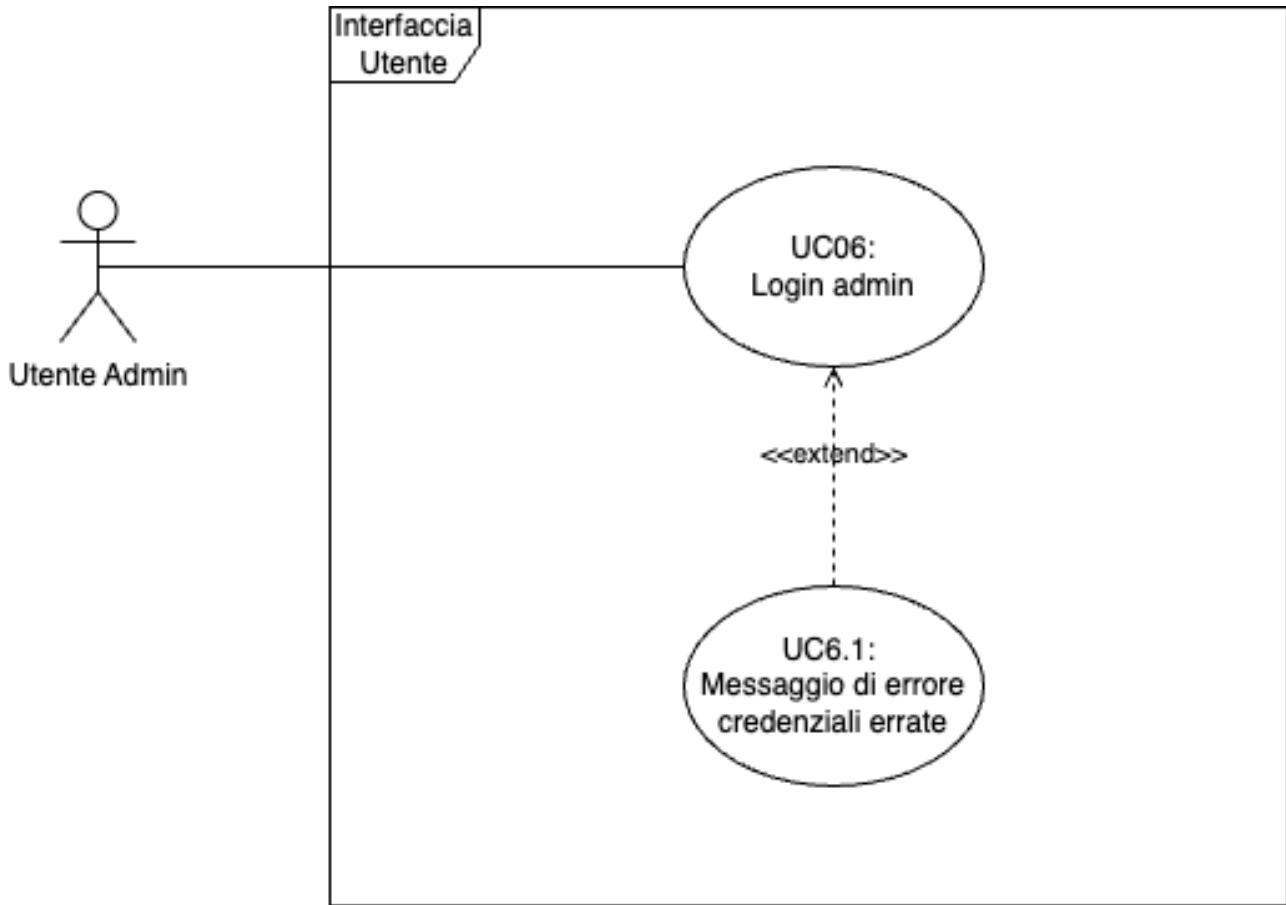


Figura 3.6: Schermata di accesso Admin

Attori coinvolti

- **Attori Primari:** Utente Admin

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin ha le credenziali di accesso.
- **Postcondizioni:**
 - L'utente Admin è autenticato e può accedere alle funzionalità di amministrazione.

Scenario principale

- L'utente Admin apre l'interfaccia di Thinky direttamente alla pagina di accesso.
- Il sistema mostra il form di accesso.
- L'utente Admin inserisce le credenziali di accesso.
- Il sistema verifica le credenziali.
- Le credenziali sono valide, l'utente Admin viene autenticato.

Estensioni

- UC6.1: Visualizzazione messaggio di errore se le credenziali sono errate.

User Story

- Come utente Admin, voglio poter accedere al sistema per gestire le funzionalità di amministrazione.

UC6.1: Visualizzazione messaggio di errore se le credenziali sono errate

Attori coinvolti

- **Attori Primari:** Utente Admin

Precondizioni e Postcondizioni

- **Precondizioni:**

- ▶ Il sistema è connesso e funzionante.
- ▶ L'utente Admin ha inserito credenziali errate.

- **Postcondizioni:**

- ▶ Il sistema visualizza un messaggio di errore.

Scenario principale

- L'utente Admin apre l'interfaccia di Thinky direttamente alla pagina di accesso.
- Il sistema mostra il form di accesso.
- L'utente Admin inserisce le credenziali errate.
- Il sistema verifica le credenziali.
- Le credenziali non sono valide, il sistema visualizza un messaggio di errore.

User Story

- Quando inserisco credenziali errate, il sistema mostra un messaggio di errore così posso correggerle e ripetere l'accesso.

3.6.2 UC07: Visualizzazione homepage Admin (Figura 3.7)

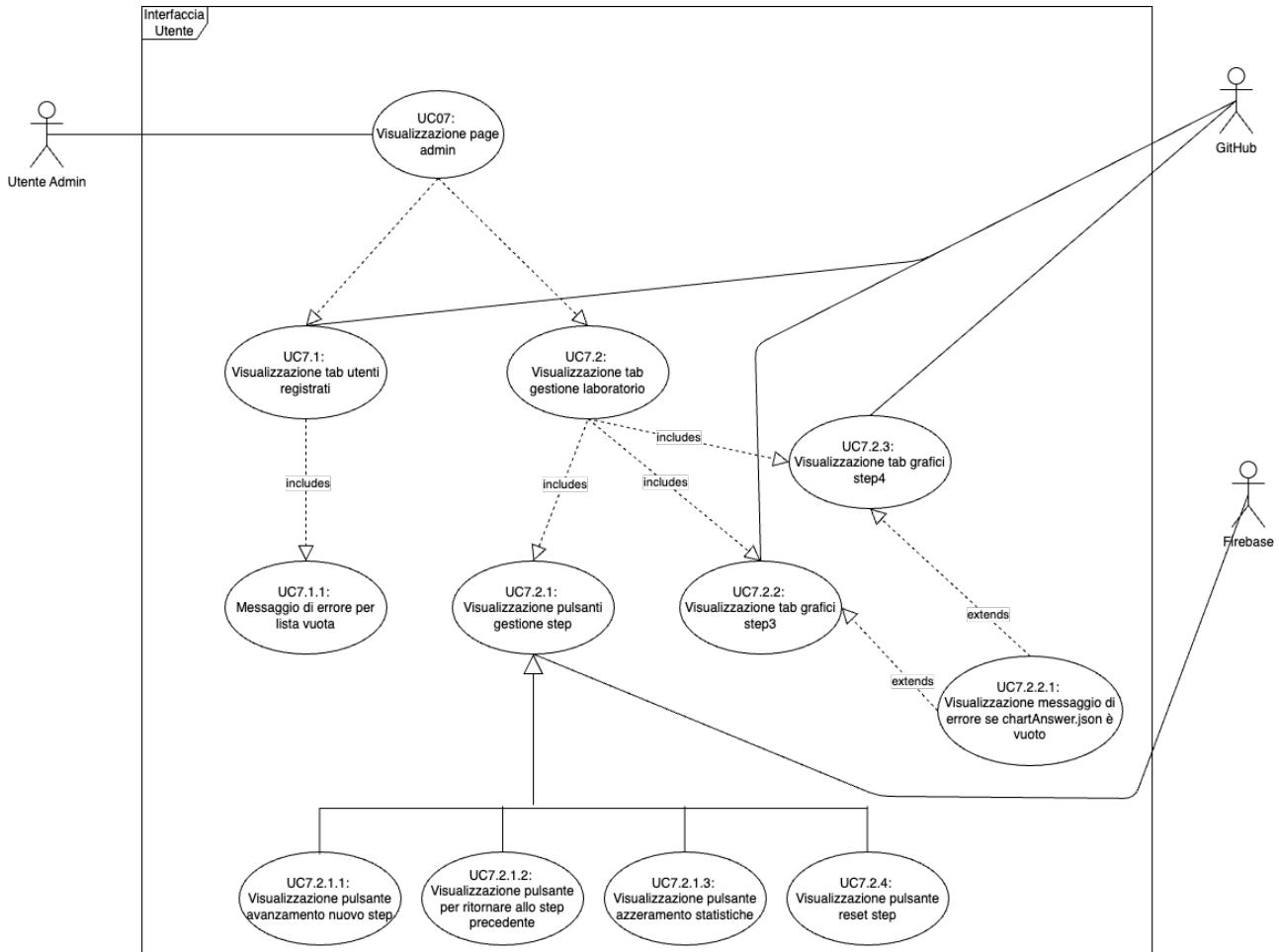


Figura 3.7: Visualizzazione homepage Admin

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub, Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.

- L'utente Admin è autenticato.
- Le API di Firebase sono disponibili e configurate correttamente.
- Le API di GitHub sono disponibili e configurate correttamente
- **Postcondizioni:**
 - L'utente Admin visualizza la homepage di amministrazione.

Scenario principale

- L'utente ha effettuato l'accesso al sistema.
- Il sistema visualizza la homepage di amministrazione.

Inclusioni

- UC07.1: Visualizzazione tab lista utenti registrati.
- UC07.2: Visualizzazione tab gestione laboratorio.

User Story

- Come utente Admin, voglio visualizzare la homepage di amministrazione per gestire le funzionalità del sistema.

UC07.1: Visualizzazione scheda lista utenti registrati

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin è autenticato.
 - Le API di GitHub sono disponibili e configurate correttamente.
- **Postcondizioni:**
 - L'utente Admin visualizza la tabella con la lista degli utenti registrati.

Scenario principale

- L'utente Admin accede alla homepage di amministrazione.
- Il sistema visualizza la tabella con la lista degli utenti registrati.

Estensioni

- UC7.1.1: Visualizzazione messaggio per lista vuota.

User Story

- Come utente Admin, voglio visualizzare la lista degli utenti registrati per gestire gli utenti del sistema.

UC7.1.1: Visualizzazione messaggio per lista vuota

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin è autenticato.
 - Le API di GitHub sono disponibili e configurate correttamente.
 - Non ci sono utenti registrati nel sistema.
- **Postcondizioni:**
 - Il sistema visualizza un messaggio che informa l'utente che la lista è vuota.

Scenario principale

- L'utente Admin accede alla homepage di amministrazione.
- Il sistema verifica se ci sono utenti registrati.
- Non ci sono utenti registrati, il sistema visualizza un messaggio che informa l'utente che la lista è vuota.

User Story

- Quando non ci sono utenti registrati, il sistema mostra un messaggio che informa l'utente Admin che la lista è vuota.

UC07.2: Visualizzazione tab gestione laboratorio

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase, GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin è autenticato.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Le API di GitHub sono disponibili e configurate correttamente.
- **Postcondizioni:**
 - L'utente Admin visualizza la Tab con le informazioni di laboratorio.

Scenario principale

- L'utente Admin accede alla homepage di amministrazione.
- Il sistema visualizza la Tab con le informazioni di laboratorio.

Inclusioni

- UC07.2.1: Visualizzazione pulsanti della gestione step del laboratorio.
- UC07.2.2: Visualizzazione Tab Grafici Step 3
- UC07.2.3: Visualizzazione Tab Grafici Step 4

User Story

- Come utente Admin, voglio visualizzare la Tab con le informazioni di laboratorio per gestire gli step del laboratorio e i grafici delle risposte date nei laboratori.

UC07.2.1: Visualizzazione pulsanti della gestione step del laboratorio

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase, GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- Le API di Firebase sono disponibili e configurate correttamente.
- Le API di GitHub sono disponibili e configurate correttamente.

- **Postcondizioni:**

- L'utente Admin visualizza i pulsanti per la gestione degli step del laboratorio.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza i pulsanti per la gestione degli step del laboratorio .
- L'utente Admin può interagire con i pulsanti per gestire gli step del laboratorio.

Generalizzazioni

- UC07.2.1.1: Visualizzazione pulsante per l'avanzamento ad un nuovo step.
- UC07.2.1.2: Visualizzazione pulsante per ritornare allo step precedente.
- UC07.2.1.3: Visualizzazione pulsante per azzeramento statistiche.
- UC07.2.1.4: Visualizzazione pulsante per il reset degli step.

User Story

- Come utente Admin, voglio visualizzare i pulsanti per la gestione degli step del laboratorio per poterli gestire in modo efficiente.

UC07.2.1.1: Visualizzazione pulsante per l'avanzamento ad un nuovo step

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema ha caricato lo step corrente correttamente.

- **Postcondizioni:**

- L'utente Admin visualizza il pulsante per l'avanzamento ad un nuovo step.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza il pulsante per l'avanzamento ad un nuovo step.
- L'utente Admin può interagire con il pulsante per avanzare ad un nuovo step.

User Story

- Come utente Admin, voglio visualizzare il pulsante per l'avanzamento ad un nuovo step per poter gestire gli step del laboratorio in modo efficiente.

UC07.2.1.2: Visualizzazione pulsante per ritornare allo step precedente

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- Le API di Firebase sono disponibili e configurate correttamente.
- Il sistema ha caricato lo step corrente correttamente.
- step corrente ≥ 1

- **Postcondizioni:**

- Il pulsante per tornare indietro di uno step è cliccabile.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza il pulsante per tornare indietro di uno step.
- L'utente Admin può interagire con il pulsante per tornare indietro di uno step.

User Story

- Come utente Admin, voglio visualizzare il pulsante per tornare indietro di uno step per poter gestire gli step del laboratorio in modo efficiente.

UC07.2.1.3: Visualizzazione pulsante per azzeramento statistiche

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase, GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin è autenticato.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Le API di GitHub sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
 - L'utente Admin ha cliccato sul pulsante per l'azzeramento delle statistiche.
- **Postcondizioni:**
 - Il pulsante per l'azzeramento delle statistiche resetta gli utenti salvati e il conteggio delle domande.

Scenario Principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza il pulsante per l'azzeramento delle statistiche.

- L'utente Admin può interagire con il pulsante per azzerare le statistiche.
- gli utenti salvati e il conteggio delle domande vengono resettati.

User Story

- Come utente Admin, voglio visualizzare il pulsante per l'azzeramento delle statistiche per poter resettare le statistiche del laboratorio in modo rapido senza dover accedere al servizio di *information storage* dove sono contenuti i file.

UC07.2.1.4: Visualizzazione pulsante per il reset degli step

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** Firebase

Precondizioni e Postcondizioni

- **Precondizioni:**
 - Il sistema è connesso e funzionante.
 - L'utente Admin è autenticato.
 - Le API di Firebase sono disponibili e configurate correttamente.
 - Il sistema ha caricato lo step corrente correttamente.
 - L'utente Admin ha cliccato sul pulsante per il reset degli step.
- **Postcondizioni:**
 - Il pulsante per il reset degli step resetta step corrente a 0.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza il pulsante per il reset degli step.
- L'utente Admin può interagire con il pulsante per resettare gli step in maniera rapida.
- step corrente viene resettato a 0.

User Story

- Come utente Admin, voglio visualizzare il pulsante per il reset degli step per poter resettare gli step del laboratorio in modo rapido senza dover accedere al servizio di *information storage* dove è contenuto il file *step.json*.

UC07.2.2: Visualizzazione Tab Grafici Step 3

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

• Precondizioni:

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- Le API di GitHub sono disponibili e configurate correttamente.
- Il sistema ha caricato correttamente i dati delle domande salvate.

• Postcondizioni:

- L'utente Admin visualizza la Tab con i grafici delle risposte date dagli utenti nello step 3.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza la Tab con i grafici delle risposte date dagli utenti nello step 3.
- L'utente Admin può interagire con i grafici per visualizzare le risposte date dagli utenti nello step 3.

Estensioni

- UC7.2.2.1: Visualizzazione messaggio di informazione se non ci sono risposte salvate o avviene un errore generico.

User Story

- Come utente Admin, voglio visualizzare la Tab con i grafici delle risposte date dagli utenti nello step 3 per analizzare le risposte degli utenti e migliorare l'esperienza del laboratorio.

UC7.2.2.1: Visualizzazione messaggio di informazione se non ci sono risposte salvate o avviene un errore generico

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

- **Precondizioni:**

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- non ci sono risposte salvate o si verifica un errore durante il caricamento dei dati.
- Le API di GitHub sono disponibili e configurate correttamente.verifica un errore durante il caricamento dei dati.

- **Postcondizioni:**

- Il sistema visualizza un messaggio di informazione che indica che non ci sono risposte salvate.

Scenario principale

- L'utente Admin accede alla Tab con i grafici delle risposte date dagli utenti nello step 3.
- Il sistema tenta di caricare i dati dnon ci sono risposte salvateci sono risposte salvate o si verifica un errore durante il caricamento dei dati.
- Il sistema visualizza un messaggio di informazione che indica che non ci sono risposte salvate.

User Story

- Quando non ci sono risposte salvate o si verifica un errore durante il caricamento dei dati, il sistema mostra un messaggio di informazione per informare l'utente Admin che non ci sono dati disponibili per lo step 3.

UC07.2.3: Visualizzazione Tab Grafici Step 4

Attori coinvolti

- **Attori Primari:** Utente Admin
- **Attori Secondari:** GitHub

Precondizioni e Postcondizioni

• Precondizioni:

- Il sistema è connesso e funzionante.
- L'utente Admin è autenticato.
- Le API di GitHub sono disponibili e configurate correttamente.
- Il sistema ha caricato correttamente i dati delle domande salvate.

• Postcondizioni:

- L'utente Admin visualizza la Tab con i grafici delle risposte date dagli utenti nello step 4.

Scenario principale

- L'utente Admin accede alla Tab con le informazioni di laboratorio.
- Il sistema visualizza la Tab con i grafici delle risposte date dagli utenti nello step 4.
- L'utente Admin può interagire con i grafici per visualizzare le risposte date dagli utenti nello step 4.

Estensioni

- UC7.2.2.1: Visualizzazione messaggio di informazione se non ci sono risposte salvate o avviene un errore generico.

User Story

- Come utente Admin, voglio visualizzare la Tab con i grafici delle risposte date dagli utenti nello step 4 per analizzare le risposte degli utenti e migliorare l'esperienza del laboratorio.

3.7 Requisiti

I requisiti sono stati individuati in base ai casi d'uso e alle user story precedentemente definiti, oltre che in base alle funzionalità richieste. I requisiti sono stati definiti in modo incrementale, partendo da quelli più generali e arrivando a quelli più specifici, e sono variati e aumentati nel tempo.

Ogni requisito è stato identificato con un codice univoco, che ne facilita la consultazione e la gestione. I requisiti sono stati classificati in tre categorie principali: requisiti funzionali, requisiti di qualità e requisiti di vincolo. Questi possono essere consultati nella tabella sottostante, che riporta il codice del requisito, la sua descrizione e la sua priorità, che si distingue in:

- **Obbligatorio:** requisito che deve essere implementato per il corretto funzionamento del sistema e che non può essere omesso.
- **Desiderabile:** requisito che migliora significativamente l'esperienza utente, ma che può essere omesso senza compromettere il funzionamento del sistema.
- **Opzionale:** requisito che può essere implementato se il tempo lo consente per arricchire le funzionalità. Questi obiettivi possono essere considerati come "extra" che migliorano l'esperienza utente o aggiungono funzionalità utili

I requisiti di questo progetto sono divisi in:

- **Requisiti funzionali:** Indicano le funzionalità specifiche del sistema, e sono identificati con il prefisso *RF*. Corrispondono agli obiettivi identificati al capitolo [Obiettivi del progetto](#) e ai casi d'uso (e sotto casi d'uso) relativi. Sono definiti il **cosa** del sistema.

- **Requisiti di qualità:** Indicano le caratteristiche qualitative del sistema, e sono identificati con il prefisso *RQ*. Sono definiti il **come** il sistema deve funzionare per garantire la miglior **UX**.
- **Requisiti di Vincolo:** Indicano i vincoli esterni al sistema, e sono identificati con il prefisso *RC*. Sono definiti le limitazioni del sistema, tutto ciò che il sistema deve rispettare per mantenere il grado massimo di qualità.

3.8 Requisiti funzionali

Codice	Descrizione	Priorità
RF01	Il prodotto deve essere implementato correttamente ed essere funzionante	Obbligatorio
RF02	Il prodotto deve rappresentare il problema dei lettori-scrittori	Obbligatorio
RF03	L'utente deve potersi registrare senza utilizzo di dati personali (UC01)	Obbligatorio
RF04	L'utente deve visualizzare un messaggio che lo informa che il nome utente scelto non è rispettoso UC1.1	Obbligatorio
RF05	L'utente deve visualizzare un messaggio che lo informa che il nome utente scelto è già in uso UC1.2	Obbligatorio
RF06	L'utente deve visualizzare un messaggio che lo informa che deve compilare tutti i campi UC1.3	Obbligatorio
RF07	L'utente deve visualizzare l'applicazione dopo la corretta registrazione (UC02)	Obbligatorio
RF08	L'utente deve poter visualizzare un messaggio di errore generico in caso di problemi	Obbligatorio
RF09	L'utente deve visualizzare la pagina del laboratorio	Obbligatorio
RF10	L'utente deve poter visualizzare gli step progressivi del laboratorio (UC04)	Obbligatorio
RF11	L'utente deve poter visualizzare la pagina di chiusura del laboratorio (UC05)	Obbligatorio

RF12	L'utente Admin deve poter accedere al sistema (UC06)	Obbligatorio
RF13	L'utente Admin deve poter visualizzare la homepage di amministrazione (UC07)	Obbligatorio
RF14	L'utente Admin deve poter visualizzare la lista degli utenti registrati (UC07.1)	Obbligatorio
RF15	L'utente Admin deve poter visualizzare un messaggio di errore se il caricamento della lista degli utenti fallisce (UC07.1.2)	Obbligatorio
RF16	L'utente Admin deve poter visualizzare la Tab con le informazioni di laboratorio (UC07.2)	Obbligatorio
RF17	L'utente Admin deve poter visualizzare i pulsanti per la gestione degli step del laboratorio (UC07.2.1)	Obbligatorio
RF18	L'utente Admin deve poter visualizzare i grafici delle risposte date dagli utenti nello step 3 (UC07.2.2)	Facoltativo
RF19	L'utente Admin deve poter visualizzare i grafici delle risposte date dagli utenti nello step 4 (UC07.2.3)	Facoltativo
RF20	Il sistema deve implementare il problema dei produttori-consumatori	Facoltativo
RF21	Il sistema deve dare all'utente la possibilità di inserire una sequenza di operazioni (UC4.2.1: Drag and Drop)	Facoltativo

Tabella 3.1: Requisiti funzionali

3.9 Requisiti di qualità

Codice	Descrizione	Priorità
RQ01	Il sistema deve essere facilmente utilizzabile e intuitivo per l'utente	Obbligatorio
RQ02	Il sistema deve essere veloce e reattivo	Obbligatorio

RQ03	Il sistema deve avere le minime basi di sicurezza informatica	Obbligatorio
RQ04	Il codice del sistema deve essere di libera consultazione su una singola repo di GitHub	Obbligatorio
RQ05	Il sistema deve essere facilmente manutenibile e aggiornabile da sviluppatori diversi dal Laureando	Obbligatorio
RQ06	Il sistema deve garantire almeno il 90% di copertura di obiettivi completati	Obbligatorio

Tabella 3.2: Requisiti di qualità

3.10 Requisiti di vincolo

Codice	Descrizione	Priorità
RV01	Il sistema deve essere sviluppato in TypeScript	Obbligatorio
RV02	Il sistema deve essere compatibile e funzionare con disinvoltura con la versione 140.0 o superiore di Firefox, che è l'ultima disponibile al momento della stesura di questo documento, e che quindi sia accessibile da ogni sistema operativo che supporta Firefox	Obbligatorio
RV03	Il sistema deve essere compatibile e funzionare con disinvoltura con la versione 137.0.7151.120 o superiore di Chrome che è l'ultima disponibile al momento della stesura di questo documento, e che quindi sia accessibile da ogni sistema operativo che supporta Chrome	Obbligatorio
RV04	Il sistema deve essere compatibile e funzionare con disinvoltura con la versione 18.1.0 o superiore di Safari, che è l'ultima disponibile al momento della stesura di questo documento, e che quindi sia accessibile da ogni sistema operativo che supporta Safari	Obbligatorio

RV05	Il sistema deve rispettare le best practices del web development e rispettare gli standard di accessibilità dettati dallo standard internazionale W3C WCAG 2.1 AAA (Web Content Accessibility Guidelines)	Obbligatorio
RV06	Il sistema deve essere compatibile con diverse dimensioni di schermo, rendendolo completamente responsive G	Obbligatorio
RV07	Il sistema deve essere compatibile con i diversi sistemi operativi (sia per dispositivi mobili che per desktop), come Windows, macOS, Linux, Android e iOS	Obbligatorio

Tabella 3.3: Requisiti di vincolo

Capitolo 4

Implementazione

Espone come le scelte descritte nel capitolo precedente sono state implementate, attraverso una panoramica del codice sorgente, delle tecnologie utilizzate, delle feature di accessibilità e delle best practices attuate durante il processo di sviluppo.

Il progetto Thinky è stato sviluppato seguendo le linee guida e le best practices del *Web Development*, con particolare attenzione all'accessibilità e alla facilità d'uso. In questo capitolo, verranno presentate le tecnologie utilizzate, le scelte progettuali adottate e una panoramica del codice sorgente.

Il progetto ha un architettura a componenti, come da standard di [ReactJS](#). Ogni componente è responsabile di una parte specifica dell'interfaccia utente e della propria logica e può essere riutilizzato in diverse parti dell'applicazione.

Queste scelte sono state fatte tenendo conto del concetto di riuso, ovvero quella pratica di utilizzare parti di codice già esistenti per evitare di riscrivere codice già funzionante, e di modularità, che permette di suddividere il codice in moduli indipendenti e riutilizzabili. Questo approccio consente di mantenere il codice organizzato e facilmente manutenibile, facilitando anche l'aggiunta di nuove funzionalità in futuro. Il riuso durante la progettazione può essere ottemperato in due modi differenti:

- Progettare *con* riuso, ovvero progettare il codice ed utilizzare librerie di componenti già pronti in modo che possa essere facilmente riutilizzato in altre parti dello stesso progetto. Questo approccio è più semplice, in quanto non richiede una conoscenza anticipata di bisogni futuri, ma richiede

comunque una buona organizzazione del codice e una buona documentazione.

- Progettare *per riuso*, ovvero progettare il codice in modo che possa essere facilmente riutilizzato in altri progetti o contesti. Questo approccio è difficile, in quanto implica una conoscenza anticipata di bisogni futuri e di come il codice potrebbe essere utilizzato in altri contesti. Tuttavia, è possibile adottare alcune best practices per facilitare il riuso del codice.

4.1 Tecnologie: scelte e implementazione

Data la natura tecnica del progetto, i suoi scopi (si veda il [capitolo secondo](#)), e tutte le motivazioni elencate sopra, la scelta più ovvia per le tecnologie da utilizzare era quella di optare per i [Framework](#). Questi rendono il processo di sviluppo più veloce e semplice, fornendo una struttura di base su cui costruire l'applicazione. Inoltre, i framework offrono anche funzionalità predefinite e convenzioni da seguire, che semplificano ulteriormente lo sviluppo.

I framework, a livello architettonale e di sviluppo sono considerati:

- [Bottom-up](#): perché partono da una base di codice già esistente e lo estendono, piuttosto che partire da zero.
- [Top-down](#): perché forniscono una struttura architettonale di base su cui costruire l'applicazione, piuttosto che partire da zero e costruire tutto da capo.

Queste tecnologie sono state scelte tenendo conto di conoscenze pregresse acquisite durante il corso di *Ingegneria del Software*, e delle tecnologie più moderne e adatte al contesto del progetto. Inoltre, sono state scelte anche in base alla loro popolarità e diffusione nel mondo dello sviluppo web, in modo da garantire una rapida curva di apprendimento per altri studenti che potrebbero voler contribuire al progetto in futuro. I linguaggi, le librerie e i vari framework utilizzati sono comunque stati approvati dalla Prof.ssa Ombretta Gaggi, che ha supervisionato e guidato le scelte tecniche durante tutto il processo di sviluppo.

4.1.1 Typescript

Typescript⁽²⁾ è alla base del progetto (Figura 4.1). È il linguaggio di programmazione utilizzato per tutto il progetto in ogni sua parte. Questo linguaggio è un *superset* (estensione) di JavaScript che aggiunge una forte tipizzazione statica che permette di evitare errori di tipo durante lo sviluppo, migliorando la qualità del codice e la sua manutenibilità. Inoltre, Typescript offre funzionalità avanzate come le interfacce, le classi e i moduli, che semplificano la scrittura di codice complesso e facilitano il riuso. Il fatto che sia un *superset* di JavaScript permette di utilizzare tutte le librerie e i framework basati su JavaScript, rendendolo estremamente versatile e compatibile con il vasto ecosistema di librerie e strumenti disponibili per lo sviluppo web.

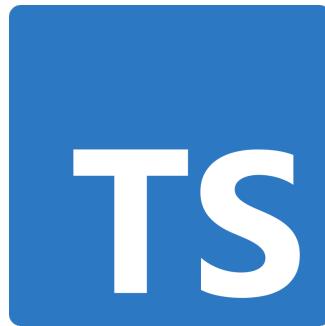


Figura 4.1: Logo di typescript

4.1.2 Axios

Axios^([3]) (Figura 4.2) è una libreria JavaScript per effettuare richieste HTTP. È stata scelta per la sua semplicità e facilità d'uso, oltre alla sua capacità di gestire le richieste in modo asincrono. Axios permette di effettuare chiamate alle API [G](#) in modo semplice e intuitivo, gestendo automaticamente la serializzazione dei dati e la gestione degli errori. Inoltre, supporta anche le richieste cancellabili e la gestione dei token di autenticazione, rendendolo una scelta ideale per il progetto Thinky.



Figura 4.2: Logo di Axios

Durante il progetto Axios è stata usata per effettuare chiamate HTTP alle API [G](#) di GitHub [G](#).

Le richieste HTTP sono il meccanismo con cui un client [G](#) comunica con un server [G](#) per richiedere risorse o inviare dati. Ogni richiesta HTTP consiste in un metodo (come GET, POST, PUT, DELETE) e un URL che identifica la risorsa richiesta. Le richieste possono anche includere intestazioni (headers) e un corpo (body) contenente dati aggiuntivi. Le risposte del server contengono uno stato (status code), intestazioni e, facoltativamente, un corpo con i dati richiesti.

In particolare, le richieste HTTP sono state utilizzate per recuperare i dati dai file JSON presenti nella repository di GitHub, che contengono le informazioni sui laboratori e le risposte degli utenti.

Le HTTP Requests sono state effettuate in modo asincrono, utilizzando le funzionalità di `async/await` di Typescript, che permettono di scrivere codice asincrono in modo più leggibile e comprensibile. Inoltre, Axios gestisce

automaticamente la serializzazione dei dati e la gestione degli errori, semplificando ulteriormente il processo di sviluppo.

Per fare le richieste con Axios, viene creato un file typescript dove vengono definite le variabili (Figura 4.3) da utilizzare nella [query](#). Nel mio caso, ho creato il file `gh.ts` e poi ho creato le seguenti variabili:



```

thinky - gh.ts

1 const REPO: string = "data"; // Nome repo
2 const OWNER: string = "orlifera"; // Proprietario repo
3 const FILE_PATH: string = "data/users.json"; // Path del file users.json
4 const ANS_FILE_PATH = "data/chartAnswer.json"; // Path del file chartAnswer.json
5 const BRANCH: string = "master"; // Branch name
6 const MAX_RETRIES: number = 3; //numero massimo di tentativi in caso di conflitto
7

```

Figura 4.3: Variabili per le richieste Axios

Per fare le richieste su una specifica *Repo* di *GitHub* però, è necessario creare un *secret token* (Codice 4.1) sul proprio profilo di GitHub. Questo verrà usato dai servizi per confermare la propria identità, confermando che sia l'effettivo proprietario della repo a fare queste *HTTP Requests*. Questo token viene poi salvato in un file `.env` che non viene mai caricato su GitHub, per evitare che altri possano utilizzarlo per accedere ai dati della propria repo. Il file `.env` contiene le variabili d'ambiente necessarie per l'autenticazione e la configurazione del progetto.

1	NEXT_PUBLIC_GITHUB_TOKEN=token_segretrissimo
---	--

Codice 4.1: Esempio di file `'.env'`

Per un miglior mantenimento del codice, prima di ogni chiamata di funzione, viene controllato se il token è creato con questa *flag* come mostrato in (Figura 4.4).

```
thinky - gh.ts
1 if (!process.env.NEXT_PUBLIC_GITHUB_TOKEN) {
2   throw new Error("GitHub token is missing. Make sure NEXT_PUBLIC_GITHUB_TOKEN is set.");
3 }
```

Figura 4.4: Controllo del token GitHub

Viene poi creata l'istanza base di Axios (Figura 4.5), che contiene le informazioni di base per effettuare le richieste:

```
thinky - gh.ts
1 // crea un'istanza di axios per l'API di GitHub
2 const githubApi = axios.create({
3   baseURL: "https://api.github.com",
4   headers: {
5     Authorization: `Bearer ${process.env.NEXT_PUBLIC_GITHUB_TOKEN}`,
6     Accept: "application/vnd.github+json",
7   },
8 });
```

Figura 4.5: Istanza base di Axios

Nel caso di `users.json`, viene effettuata una richiesta `GET`(Codice 4.2) per recuperare i dati degli utenti:

```
1 //richiesta GET per recuperare i dati degli utenti
2 export const fetchUsers = async (): Promise<User[]> => {
3   try {
4     const response = await githubApi.get(`repos/${OWNER}/${REPO}/contents/
5     ${FILE_PATH}`);
5     const content = JSON.parse(atob(response.data.content)); // Decode Base64
6     return content.map((user: User) => ({
7       ...user,
8       date: user.date,
```

```

9      });
10     } catch (error) {
11       console.error("Failed to fetch users:", error);
12       throw new Error("Could not fetch users");
13     }
14   };

```

Codice 4.2: Richiesta GET per recuperare i dati degli utenti

Mentre per scrivere sul file i dati degli utenti appena registrati, viene effettuata una richiesta **PUT**.

Tutti i dati mandati e richiesti sono crittografati in **Base64**, per evitare che i dati possano essere letti da chiunque abbia accesso alla repo. Questo è un passaggio fondamentale per garantire la sicurezza dei dati degli utenti e per proteggere la privacy degli stessi. Vengono poi utilizzate le funzioni **btoa** e **atob** per codificare e decodificare i dati in Base64. Queste funzioni sono disponibili in TypeScript e permettono di convertire i dati in una stringa Base64, che può essere facilmente trasmessa attraverso le richieste HTTP.

4.1.3 Firebase

Firebase^([4]) (Figura 4.6) è un servizio di Google che fornisce un **Backend as a Service** (BaaS) per lo sviluppo di applicazioni web e mobili. I servizi BaaS sono una soluzione cloud che fornisce tutte le funzionalità tipiche di un backend, senza la necessità di doverlo creare manualmente e averne l'infrastruttura.



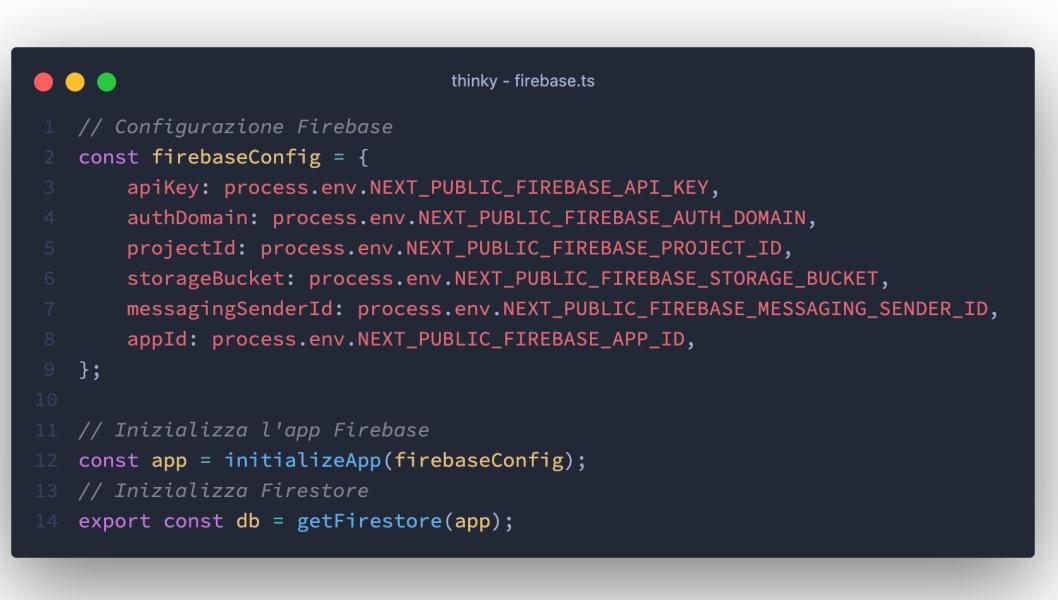
Firebase

Figura 4.6: Logo di Firebase

Nel caso di Thinky, Firebase è stato utilizzato per soppiare alle mancanze di GitHub, che non permetteva di avere un aggiornamento live dei dati. Questo

è dovuto al fatto che GitHub è un servizio di versionamento del codice, e non un database in tempo reale, e inoltre la funzione `fetchStep` è contenuta in un [server-side component](#) di NextJS e che quindi viene eseguito solo una volta al caricamento della pagina. Inoltre, GitHub ha un ritardo sulle richieste dovuto al CDN Caching.

Per creare un database con Firebase, tutto quello che serve è creare un progetto sul sito ufficiale di Firebase e configurare il file `firebase.ts` (Figura 4.7) con le credenziali del progetto. Il file `firebase.ts` contiene le informazioni necessarie per connettersi al database Firebase e per utilizzare i servizi offerti da Firebase, come l'autenticazione e il database in tempo reale.



```

thinky - firebase.ts

1 // Configurazione Firebase
2 const firebaseConfig = {
3     apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
4     authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
5     projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
6     storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
7     messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID,
8     appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID,
9 };
10
11 // Inizializza l'app Firebase
12 const app = initializeApp(firebaseConfig);
13 // Inizializza Firestore
14 export const db = getFirestore(app);

```

Figura 4.7: File `firebase.ts`

Questo file viene automaticamente generato da Firebase quando si crea il nuovo progetto. Per aggiornare i dati in tempo reale, viene utilizzato il database Firestore di Firebase, che permette di avere un database NoSQL in tempo reale, con la possibilità di effettuare query e aggiornamenti in modo semplice e veloce.

Per vedere *live* i dati aggiornati, viene utilizzato il metodo `onSnapshot()` di Firestore, che permette di ascoltare le modifiche ai dati in tempo reale e di

aggiornare l’interfaccia utente di conseguenza. Questo metodo è molto utile per Thinky, perché azzera il *delay* che i test eseguiti con GitHub hanno mostrato nell’aggiornamento dello step, garantendo così la stessa esperienza fluida e divertente ad ogni studente.

```

1  useEffect(() => {
2      const unsubscribe = onSnapshot(
3          doc(db, "lab", "step"),
4          (snap) => {
5              setCurrentStep(snap.data()?.currentStep ?? 0)
6              setError(null)
7          }
8      ),
9      (err) => {
10         setError(err)
11         setCurrentStep(null)
12     }
13 )
14
15     return () => unsubscribe()
16 }, [])

```

Codice 4.3: Utilizzo di `onSnapshot()` per aggiornare i dati in tempo reale

4.1.4 GitHub

GitHub_([5]) (Figura 4.8) è una piattaforma di hosting per il controllo di versione e la collaborazione, che permette di gestire progetti software e di tenere traccia delle modifiche al codice sorgente. È stata scelta per il progetto Thinky per la sua popolarità e per le sue funzionalità avanzate di gestione del codice, ma anche come «backend» fittizio.

È stato deciso di utilizzare GitHub come «backend» fittizio per il progetto Thinky, in quanto non era necessario un vero e proprio backend per il progetto. Avendo inizialmente deciso di non utilizzare un database, GitHub era la scelta più ovvia per gestire i dati del progetto, reperendoli quando necessario attraverso le sue API, dando così un valore aggiunto alla WebApp.

I dati reperiti dalle API di GitHub sono stati:

- Avatar degli utenti, per mostrare le immagini dei profili degli studenti che hanno partecipato al laboratorio nella navbar.

- Risposte degli utenti, per mostrare il conto delle risposte degli utenti nell'area admin.

Tutte le informazioni sono state salvate in file JSON all'interno di una repository di GitHub separata da quella del codice sorgente, che sono stati poi letti e scritti tramite le API di GitHub.

In particolare, per reperire le immagini degli avatar, è stata creata una funzione che prende come parametro il nome utente scelto durante la registrazione e lo inserisce all'interno dell'URL della cartella contenente le immagini degli avatar, in modo che prenda quella corrispondente al nome utente scelto. Le immagini degli avatar sono state salvate in una cartella chiamata avatars all'interno della repository di GitHub.

Per le risposte degli utenti, invece, è stato creato un file JSON chiamato chartAnswer.json che contiene il numero di risposte per ogni domanda. Questo file viene aggiornato ogni volta che l'utente admin avanza di step, in modo da tenere traccia delle risposte degli utenti in tempo reale.

GitHub, come anticipato, è stato utilizzato anche nel suo ruolo principale, quindi di Version Control System. L'utilizzo di GitHub ha permesso di tenere traccia delle modifiche al codice sorgente, di poter tornare indietro nel caso di *breaking changes* e di tenere traccia dei progressi attraverso le Issues.

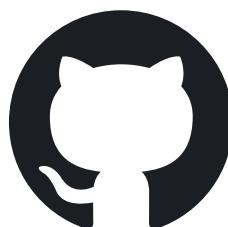


Figura 4.8: Logo di GitHub

4.1.5 ShadCN

ShadCN_([6]) (Figura 4.10) è una libreria di componenti User Interface [open source](#) basata su ReactJS e TailwindCSS progettata per fornire componenti UI moderni, accessibili e facilmente personalizzabili. È stata scelta per Thinky per svariati motivi, primo tra cui l'accessibilità.

Infatti, ShadCN è progettata per essere accessibile e conforme agli standard WCAG (Web Content Accessibility Guidelines), garantendo che tutti gli utenti possano utilizzare l'applicazione senza difficoltà.

Questa era una prerogativa fondamentale del progetto, in quanto volevo garantire il facile utilizzo della piattaforma a tutti gli studenti, senza esclusioni.

Un'altra motivazione importante è stata la facilità di personalizzazione dei componenti con TailwindCSS e il loro adattamento al tema chiaro e a quello scuro. Infatti, ogni componente di ShadCN è predisposto con delle classi CSS generiche, che vengono definite dopo dallo sviluppatore nel file `globals.css`, permettendo così di personalizzare facilmente l'aspetto dei componenti senza dover riscrivere il codice sorgente della libreria.

ShadCN inoltre mette a disposizione una semplice logica per il cambio tema della pagina. Creando un `ThemeProvider` (Figura 4.9), da utilizzare nel layout (Codice 4.4), questo verrà applicato ad ogni pagina, e di conseguenza ad ogni componente di essa, garantendo una coerenza visiva e una facile gestione del tema scelto dall'utente.



```

thinky - ThemeProvider.tsx

1  export function ThemeProvider({
2      children,
3      ...props
4  }: React.ComponentProps<typeof NextThemeProvider>) {
5      return (
6          <NextThemeProvider {...props}>
7              {children}
8          </NextThemeProvider>
9      )
10 }

```

Figura 4.9: Esempio di ThemeProvider di ShadCN

```

1 import { ThemeProvider } from '@/components/ThemeProvider';
2
3
4 export default function RootLayout({
5   children,
6 }: Readonly<{
7   children: React.ReactNode;
8 }>) {
9   return (
10     <ThemeProvider
11       attribute="class"
12       defaultTheme="system"
13       enableSystem
14       disableTransitionOnChange
15     >
16       <div id="main-content" tabIndex={-1}>
17         {children}
18       </div>
19     </ThemeProvider>
20   )
21 }

```

Codice 4.4: Esempio di layout.tsx con ThemeProvider

In questo caso, `{children}` indica tutto ciò che viene passato come contenuto del componente `ThemeProvider`, e che quindi verrà avvolto, in gergo *wrappato* dal tema scelto dall'utente. Questo permette di avere un tema coerente in tutta l'applicazione, senza dover ripetere il codice per ogni pagina o componente. ShadCN ha quindi permesso di avere uno stile coeso e coerente tra tutte le pagine, e grazie alla sua stretta integrazione con TailwindCSS, ha permesso di risparmiare tempo nello sviluppo dell'UI garantendo una maggior concentrazione di risorse nello sviluppo della logica e nello studio dei problemi di sincronizzazione.

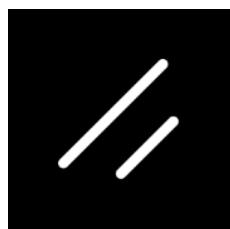


Figura 4.10: Logo di ShadCN

4.1.6 NextJS

NextJS^[7] (Figura 4.11) è un framework di React utilizzato per lo sviluppo *fullstack* di applicazioni web moderne e performanti. Questo framework offre di default molte funzionalità avanzate, utili per migliorare le performance del prodotto. Ad esempio il Server Side Rendering (SSR) delle componenti, che permette di renderizzare una pagina sul server prima di inviarla al client, migliorando le prestazioni e diminuendo il codice che il client deve interpretare, aumentando la velocità di caricamento della pagina stessa. Inoltre, NextJS offre anche la generazione statica delle pagine, che permette di creare pagine statiche a partire dai dati dinamici, migliorando ulteriormente le prestazioni e l'ottimizzazione SEO.

NextJS utilizza una struttura a file e cartelle per organizzare le pagine e i componenti dell'applicazione, che rende facile la navigazione e la gestione del progetto. Mette anche a disposizione dei file di *default* da utilizzare in diverse occorrenze.



Figura 4.11: Logo di NextJS

Layout.tsx

Il file `layout.tsx` (Figura 4.12) è il più importante di tutti, è l'unico file che non può mancare in un progetto (non a caso, Next lo crea automaticamente se viene cancellato). Deve essercene almeno uno, ed è il file che dà la struttura principale a tutte le pagine dell'applicazione. Funge da *blueprint* per tutte le pagine, permettendo di definire la loro struttura. Tutto ciò che viene incluso in un file `layout.tsx` sarà riportato in ogni pagina che fa riferimento a quel file (possono esistere più file `layout.tsx` in un progetto, ma ogni pagina può fare riferimento solo ad uno di essi, in base alla folder structure e alla posizione di suddetta pagina). Questo file è quindi fondamentale per garantire una struttura coerente e uniforme in tutta l'applicazione, e permette di definire le parti comuni a tutte le pagine, come il menu di navigazione, il footer e altre

componenti condivise. Come mostrato prima infatti, il `ThemeProvider` viene inserito all'interno del file `layout.tsx` principale, in modo da applicare il tema scelto dall'utente a tutte le pagine dell'applicazione.

Nel caso di Thinky sono stati creati tre file `layout.tsx`:

- Il file `layout.tsx` principale, che contiene la struttura di base dell'applicazione e il `ThemeProvider`.
- Il file `layout.tsx` per la pagina di laboratorio.
- Il file `layout.tsx` per le altre pagine.

Questi verranno approfonditi nella [sezione apposita](#).

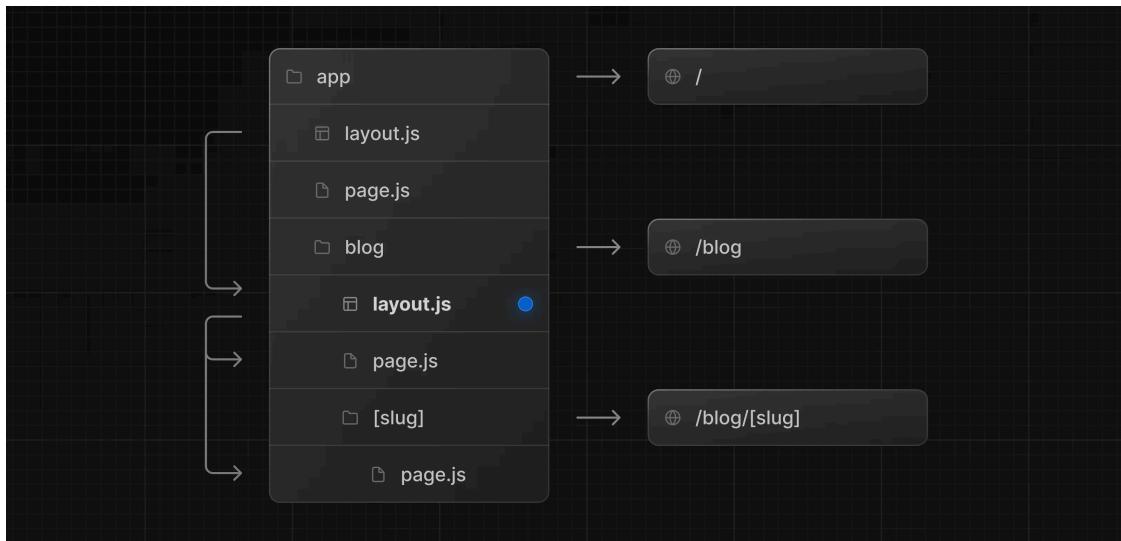


Figura 4.12: Struttura del file Layout

Nell'immagine sopra, il file `layout.tsx` principale è quello che si trova nella Root (`app/` folder) del progetto, e contiene la struttura di base dell'applicazione. I file `layout.tsx` secondari sono quelli che si trovano all'interno delle cartelle delle pagine (in questo caso `Blog/`), e contengono la struttura specifica per quelle pagine. I file `layout.tsx` si «sommano», ovvero ogni file `layout.tsx` si applica a tutte le pagine che si trovano nella sua cartella e nelle sue sottocartelle, inclusi altri file `layout.tsx`, creando una struttura ad albero.

page.tsx

Per creare le varie pagine, NextJS utilizza un sistema di routing *folder-based*, in cui ogni pagina è rappresentata da un file `page.tsx` all'interno di una cartella. Questo sistema permette di creare pagine in modo semplice e intuitivo, senza dover configurare manualmente le rotte dell'applicazione. Ogni file `page.tsx` rappresenta una pagina dell'applicazione e viene automaticamente associato a una rottura corrispondente.

Il file `page.tsx` è il file che contiene il codice della pagina. Ogni pagina dell'applicazione deve avere un file `page.tsx` fatta eccezione per la pagina Home, che è contenuta nella Root del progetto, ed è visualizzata nella barra degli indirizzi con il «/» subito dopo il dominio: <https://nextproject.com/>.

Tutti i file `page.tsx` devono essere esportati come componenti React, in modo da poter essere renderizzati correttamente da NextJS, e si attengono alla struttura definita nel file `layout.tsx` più vicino.

Con «file più vicino» si intende il primo file `layout.tsx` che si trova percorrendo a ritroso il percorso partendo dal file `page.tsx` stesso verso la Root del progetto. Se non viene trovato nessun file `layout.tsx` lungo il percorso, NextJS utilizzerà il file `layout.tsx` principale, che si trova nella Root del progetto.

global-error.tsx

Il file `global-error.tsx` (Figura 4.13) è un file speciale di NextJS che permette di gestire gli errori a livello globale dell'applicazione. Utile per creare una pagina di errore generica che viene visualizzata quando si verifica un errore in una qualsiasi pagina dell'applicazione. Questo file viene utilizzato per gestire gli errori in modo centralizzato, evitando di dover gestire gli errori in ogni singola pagina.

Una particolarità di questo file, è che ridefinisce i tag `<html>` e `<body>` della pagina in quanto sostituisce completamente il contenuto della pagina di errore, «auto-creando» il proprio layout.

Tutti gli «error-components» devono essere marcati come `use client`, ovvero «client-side components», in quanto devono essere eseguiti sul client e non sul server.



```

1  export default function GlobalError({
2    error,
3  }: {
4    error: Error & { name?: string, message?: string, digest?: string }
5 }) {
6   return (
7     <html>
8       <body className="h-screen flex flex-col bg-white items-center justify-center">
9         <div className="flex flex-col items-center">
10           <Alert variant="destructive">
11             <AlertCircle className="h-4 w-4" />
12             <AlertTitle>Error</AlertTitle>
13             <AlertDescription>
14               C'è stato un errore, per favore prova di nuovo.
15             </AlertDescription>
16           </Alert>
17           <Button onClick={() => window.location.reload()} className="mt-4 items-center">
18             Prova di nuovo
19           </Button>
20         </div>
21         <div className="mt-4">
22           More info:
23           <ul>
24             <li>{error.name}</li>
25             <li> {error.digest}</li>
26             <li>{error.message}</li>
27           </ul>
28         </div>
29       </body>
30     </html>
31   )
32 }

```

Figura 4.13: global-error.tsx di Thinky

Quindi, questo componente, verrà visualizzato ogni volta che si verifica un errore in Thinky. In particolare, la pagina mostrata sarà (Figura 4.14):



Figura 4.14: Pagina di errore globale di Thinky

Server e Client components

In NextJS, per standard, tutti i `layout.tsx` e tutti i file `page.tsx` sono considerati *server-side components*, ovvero componenti che vengono eseguiti sul server e che possono accedere ai dati del server, come ad esempio le API o il database. Questi componenti vengono renderizzati sul server e poi inviati al client, dove vengono visualizzati.

I *server-side components*^[8] permettono di effettuare il `fetch` dei dati e di renderizzare parte della UI direttamente sul server, prima di arrivare al Client.

I *client components*^[9] invece, differiscono dai primi in quanto vengono eseguiti sul client e non sul server. Questi componenti possono essere utilizzati per gestire l'interazione con l'utente, come ad esempio la gestione degli eventi o la modifica dello stato dell'applicazione. I *client components* sono marcati con la direttiva `use client` all'inizio del file, e possono essere utilizzati all'interno dei *server-side components*.

4.1.7 TailwindCSS

TailwindCSS^[10] (Figura 4.15) è un framework CSS utility-first G che permette di creare interfacce utente mo-

derne e personalizzabili in modo semplice e veloce. Funziona attraverso l'utilizzo di classi CSS predefinite, che possono essere combinate per creare stili complessi senza dover scrivere codice CSS personalizzato. Questo approccio consente di risparmiare tempo e di evitare la scrittura di codice CSS complesso, rendendo lo sviluppo dell'interfaccia utente più rapido e intuitivo. Dopo l'installazione di Tailwind, si importa il file [globals.css](#), che contiene le classi CSS predefinite di Tailwind, e si possono iniziare ad utilizzare le classi CSS direttamente nei file [page.tsx](#).

È stato scelto per il progetto Thinky per la sua facilità d'uso, che permette allo sviluppatore di non preoccuparsi della specificità delle regole CSS in quanto le classi di Tailwind sono già create, ed esposte nella documentazione, e vengono inserite direttamente nel *markup* HTML o TSX.



Figura 4.15: Logo di TailwindCSS

Creare stili con Tailwind è molto semplice: quando si vuole applicare lo stile ad un elemento, basta aggiungere la classe desiderata al tag stesso dell'elemento, poi il compilatore di Tailwind si occuperà di generare il CSS necessario per applicare lo stile desiderato. Questo approccio permette di creare interfacce utente in modo rapido e intuitivo, senza dover scrivere codice CSS complesso(Codice 4.5).

```
1 <button id="test-button">
2   Click me
3 </button>
```

Codice 4.5: HTML Esempio

Soltamente, in CSS base (Codice 4.6), per applicare degli stili a questo elemento si avrebbe:

```
1 #test-button {
```

```

2 background-color: #3b82f6; /* blu-500 */
3 color: white;
4 font-weight: bold;
5 padding: 0.5rem 1rem; /* py-2 px-4 */
6 border-radius: 0.25rem; /* rounded */
7 }
8 #test-button:hover {
9   background-color: #1d4ed8; /* blu-700 */
10 }
```

Codice 4.6: Esempio di codice CSS per HTML

Mentre con TailwindCSS (Codice 4.7), si avrebbe:

```

1 <button id="test-button" className="bg-blue-500 text-white font-bold
2 py-2 px-4 rounded hover:bg-blue-700">
3 Click me
4 </button>
```

Codice 4.7: Esempio di codice TailwindCSS per HTML

Questo approccio, oltre che ad essere più conciso, permette di applicare lo stile direttamente all'elemento, senza preoccupazioni di sovrascritture da parte di stili applicati ad altri elementi prima di questo.

Nel caso in cui si volesse applicare uno stile unico a diversi elementi dello stesso tipo, ad esempio tutti i button del sito, invece di utilizzare le classi di default di Tailwind, si definiscono delle variabili personalizzate, come si farebbe in CSS, e si applicano a tutti gli elementi che si vogliono stilizzare. Questo permette di avere uno stile coerente in tutta l'applicazione, e in caso di modifica, cambiare solamente la variabile personalizzata invece di dover cambiare la classe ad ogni elemento.

4.1.8 LucideReact

LucideReact_[11] (Figura 4.16) è una libreria di icone open source per ReactJS, che offre una vasta gamma di icone personalizzabili e facilmente integrabili nelle applicazioni web. È stata scelta per il progetto Thinky per la sua facilità d'uso e per la sua compatibilità con TailwindCSS, che permette di personalizzare le icone in modo semplice e veloce.



Figura 4.16: Logo di LucideReact

Per utilizzare LucideReact (Figura 4.17), dopo aver installato la libreria, basta importare l'icona desiderata come un componente ed applicare eventuali stili.

```

● ● ● thinky - BackToTop.tsx
1 <ArrowUp className="h-6 w-6" />

```

Figura 4.17: Esempio di utilizzo di LucideReact che renderizza una freccia verso l'alto

4.1.9 DND Kit

DND Kit_([12]) (Figura 4.18) è una libreria di Drag and Drop per ReactJS che permette di creare interfacce utente interattive e dinamiche, con la possibilità di trascinare e rilasciare elementi all'interno dell'applicazione. È stata scelta per il progetto Thinky per la sua facilità d'uso e per la sua compatibilità con TailwindCSS, che permette di personalizzare gli stili degli elementi trascinabili in modo semplice e veloce.

Inoltre, DND Kit offre di default tutte le caratteristiche di accessibilità compatibili con gli standard WCAG, garantendo che tutti gli utenti possano utilizzare l'applicazione senza difficoltà. Questo è stato un aspetto fondamentale per il progetto Thinky, in quanto si voleva garantire il facile utilizzo della piattaforma a tutti gli studenti, senza esclusioni.



Figura 4.18: Logo di DND Kit

4.2 Implementazione del codice e risultati

In questa sezione verranno presentate le scelte progettuali più importanti fatte durante lo sviluppo di Thinky, spiegando i motivi dietro a queste scelte e come sono state effettivamente implementate.

4.2.1 Registrazione

La prima cosa che uno studente è chiamato a fare quando apre Thinky, è la registrazione. Questo passaggio è fondamentale, sia per lo studente per poter iniziare ad utilizzare l'app, sia per l'utente admin, che potrà poi vedere le risposte degli studenti e gestire il laboratorio.

Le prerogative della registrazione erano che fosse semplice, e che non chiedesse dati personali, così da non dover gestire la privacy degli studenti come da GDPR.

Per questo motivo, è stata data l'opzione all'utente di generare uno username randomico, o di sceglierne uno personalizzato, che non dovesse obbligatoriamente essere il proprio nome. L'utente è informato di questa possibilità tramite un [tooltip](#) che appare al passaggio del mouse sull'icona (o al click sull'inconca se da mobile) delle informazioni al momento della registrazione. Inoltre, in questo *tooltip* (Figura 4.19) l'utente è informato che lo username deve essere rispettoso.



Figura 4.19: Tooltip di Thinky

Per la registrazione è stato creato un form (Figura 4.20) che richiede all'utente due informazioni:

- Il nome utente, che può essere generato randomicamente o scelto dall’utente.
- La scuola di provenienza, utile per tenere traccia degli studenti che partecipano al laboratorio.

Benvenuto! 🎉 Scegli il tuo nome

Nome utente ⓘ

Es. Gatto Rosso

Genera un nome casuale

Scuola di provenienza

La tua scuola

Inizia

Figura 4.20: Form di registrazione di Thinky

Per effettuare la registrazione, e controllare che lo username non sia già stato utilizzato, viene effettuata una richiesta **PUT** al file **users.json** presente nella repository di GitHub. Questo file contiene le informazioni degli utenti registrati, e viene aggiornato ogni volta che un nuovo utente si registra. In questo modo, si garantisce che ogni utente abbia uno username unico e che non ci siano conflitti tra gli utenti. Nel caso in cui lo username scelto dall’utente sia già stato utilizzato, viene mostrato un messaggio di errore e l’utente è invitato a scegliere un altro username. (Figura 4.21)



Figura 4.21: Messaggio di errore per username già utilizzato

Il nome utente inoltre, prima di essere salvato, viene comparato ad una lista di parole scurrili, e se questo corrisponde ad una di esse viene mostrato un messaggio di errore (Figura 4.22) e l’utente è invitato a scegliere un altro username.

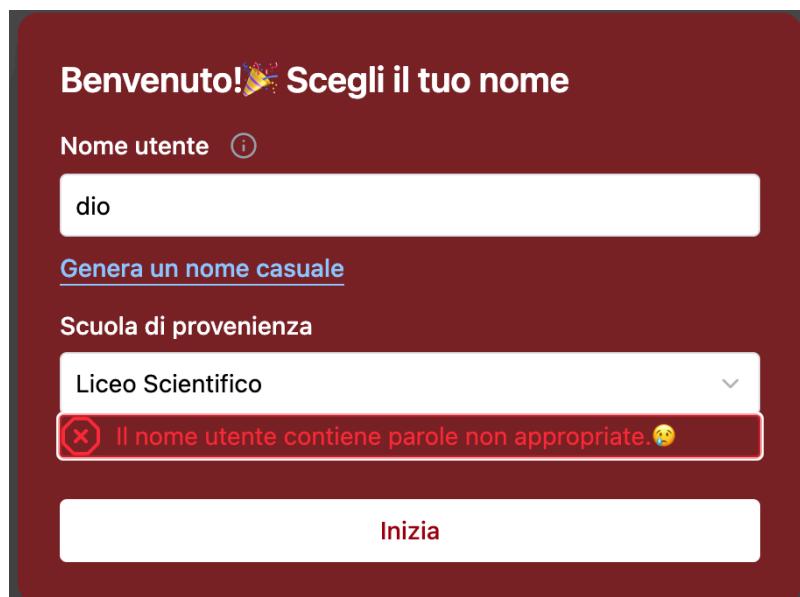


Figura 4.22: Messaggio di errore per username non rispettoso

Ogni campo del form è obbligatorio, e se l'utente non compila uno dei campi viene mostrato un messaggio di errore (Figura 4.23), invitando l'utente a compilare tutti i campi.

Benvenuto! Scegli il tuo nome

Nome utente ⓘ

Es. Gatto Rosso

Genera un nome casuale

Scuola di provenienza

Liceo Scientifico

Compila tutti i campi. 😞

Inizia

Figura 4.23: Messaggio di errore per campi mancanti

4.2.2 Pagina Admin

La pagina admin (Figura 4.24) è la pagina a cui l'organizzatore dell'attività può accedere quando vuole gestire il laboratorio o vedere le risposte degli studenti.

Benvenuto nell'area riservata

Tabella Utenti

Ecco la lista degli utenti registrati:

#	Nome Utente	Scuola	Data di Registrazione
1	Cane Blu	Altro	mar 24 giugno 2025 alle ore 17:42
2	Riccio Lito	Altro	mar 24 giugno 2025 alle ore 17:45
3	Gufo Nero	Liceo Scientifico	mar 24 giugno 2025 alle ore 18:14

Questa tabella mostra gli utenti registrati, la loro scuola e la data di registrazione.

Dipartimento di Matematica
Via Trieste 63, Padova, 35129
Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA
UNIVERSITÀ DEGLI STUDI DI PADOVA

Area riservata

Figura 4.24: Pagina admin di Thinky

Questa pagina è accessibile attraverso un login con password (Figura 4.25). Anche in questo caso, è stato deciso di non chiedere dati personali, e di non utilizzare metodi di autenticazione complessi, optando quindi per una semplice password salvata sul un file `.env`.

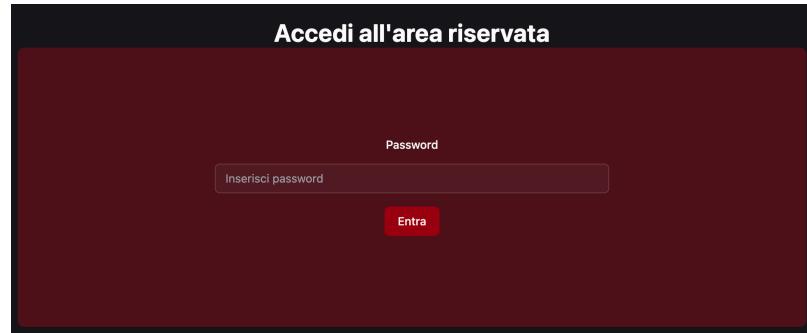


Figura 4.25: Form di login per la pagina admin

Se la password inserita è corretta, l'utente viene reindirizzato alla pagina admin, altrimenti viene mostrato un messaggio di errore (Figura 4.26).

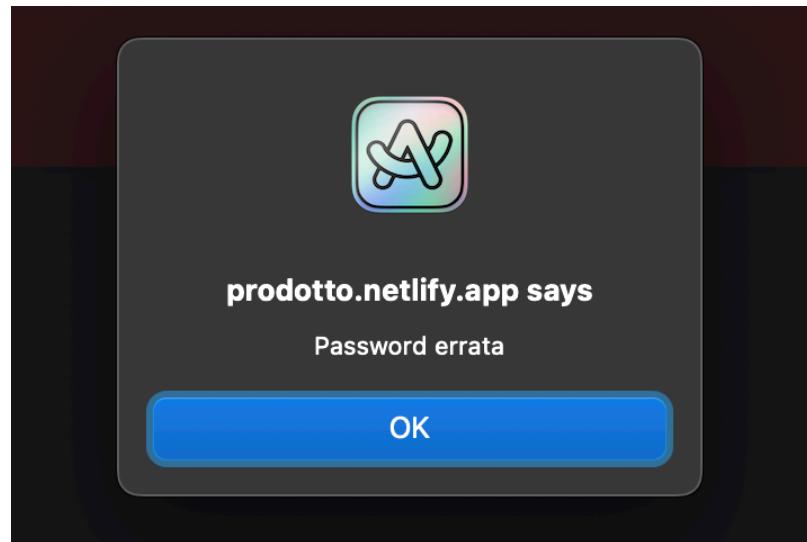


Figura 4.26: Messaggio di errore per password errata

Questa pagina è suddivisa in due sezioni, raggruppate in due tab:

- Lista utenti(Figura 4.27), che mostra la lista di tutti gli utenti registrati, insieme alla data e alla scuola di provenienza.

#	Nome Utente	Scuola	Data di Registrazione
1	Cane Blu	Altro	mar 24 giugno 2025 alle ore 17:42
2	Riccio Lilla	Altro	mar 24 giugno 2025 alle ore 17:46
3	Gufo Nero	Liceo Scientifico	mar 24 giugno 2025 alle ore 18:14

Figura 4.27: Tab Lista Utenti

- Gestione laboratorio(Figura 4.28), che mostra la sezione dedicata alla gestione degli step del laboratorio, e la visualizzazione dei grafici delle risposte.

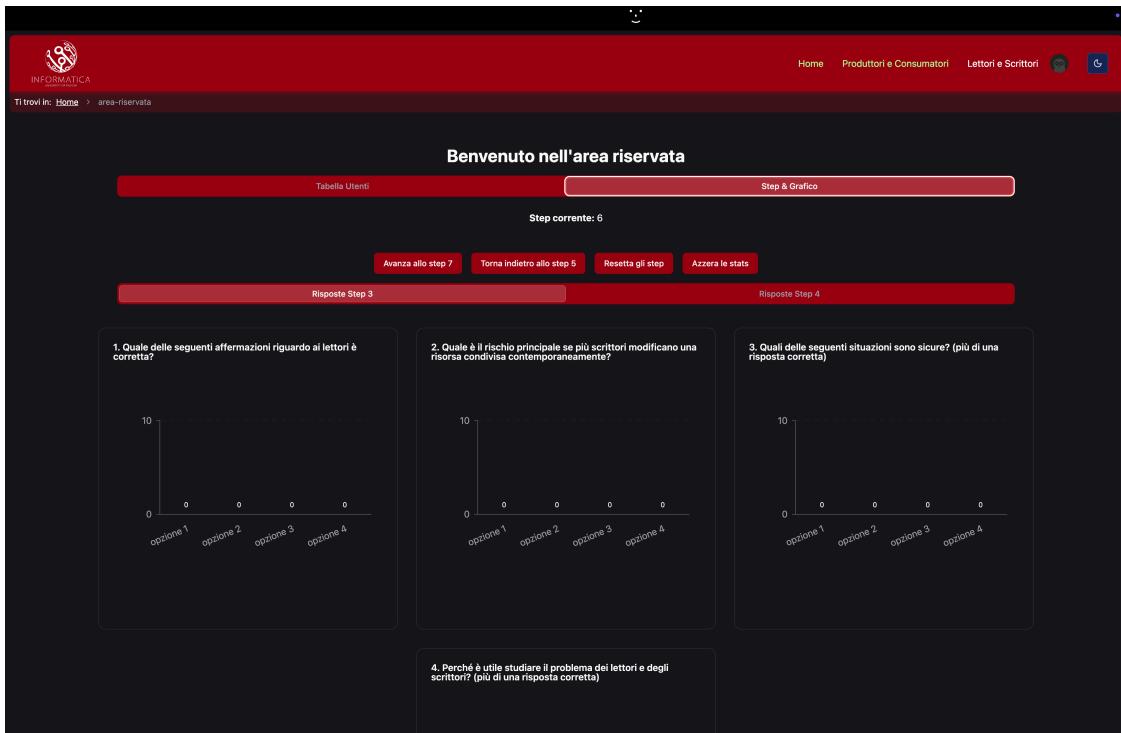


Figura 4.28: Tab Gestione Laboratorio

I bottoni vengono usati per gestire gli step progressivi, in particolare permettono di:

- Avanzare di uno step.
- Tornare indietro di uno step.
- Resettare il laboratorio, cancellando tutti gli utenti e tutte le loro risposte.
- Resettare lo step corrente a 0.

4.2.3 Laboratorio

Il laboratorio è il fulcro centrale dell’esperienza per cui è nata l’idea di Thinky. L’attività, che deve durare circa 1 ora, è svolta insieme ad un docente, che spiega agli studenti man mano i problemi da risolvere, e che può decidere di fermarsi per spiegare meglio un concetto o per rispondere a domande degli studenti. Il laboratorio è suddiviso in step, che corrispondono a problemi da risolvere, e che vengono mostrati agli studenti dinamicamente, in modo che gli studenti non possano accedere agli step successivi prima di aver risolto quello corrente. Questo permette di garantire che gli studenti non saltino i

problemi e che seguano l'ordine corretto. Ogni step viene caricato manualmente dall'utente admin, che può decidere di avanzare di uno step o di tornare indietro, in base alle esigenze del laboratorio.

Gli step di laboratorio sono 6, da sommare a uno «step 0» che è la pagina iniziale in cui si spiega agli studenti cosa andranno a fare, e lo «step 7» che è una pagina che mostra agli studenti che hanno terminato il laboratorio. I 6 step interattivi sono così suddivisi:

- **Step 1:** Drag and Drop relativo al problema del «produttore-consumatore», in cui gli studenti devono trascinare i pezzi di codice nella giusta posizione per completare la struttura del Produttore.
- **Step 2:** Drag and Drop relativo al problema del «produttore-consumatore», in cui gli studenti devono trascinare i pezzi di codice nella giusta posizione per completare la struttura del Consumatore.
- **Step 3:** Domande Teoriche sul problema dei «lettori-scrittori». In questo step gli studenti devono rispondere a 4 domande, due a risposta singola e due a risposta multipla, riguardanti il problema dei lettori-scrittori. Le domande sono state create per verificare la comprensione del problema e per stimolare la riflessione sugli argomenti trattati. Ogni domanda dispone di un [hint](#) (Figura 4.29) che può essere cliccato per mostrare un suggerimento, utile per gli studenti che hanno difficoltà a rispondere.
- **Step 4:** Domande di teoria sul problema dei «lettori-scrittori» in forma di menù a tredina divise in 3 categorie:
 - associazione ruolo-comportamento
 - classificazione comportamento
 - completa la frase
- **Step 5:** Drag and Drop relativo al problema dei «lettori-scrittori», in cui gli studenti devono trascinare i pezzi di codice nella giusta posizione per completare la struttura del Scrittore.

- **Step 6:** Drag and Drop relativo al problema dei «lettori-scrittori», in cui gli studenti devono trascinare i pezzi di codice nella giusta posizione per completare la struttura dello Lettore.

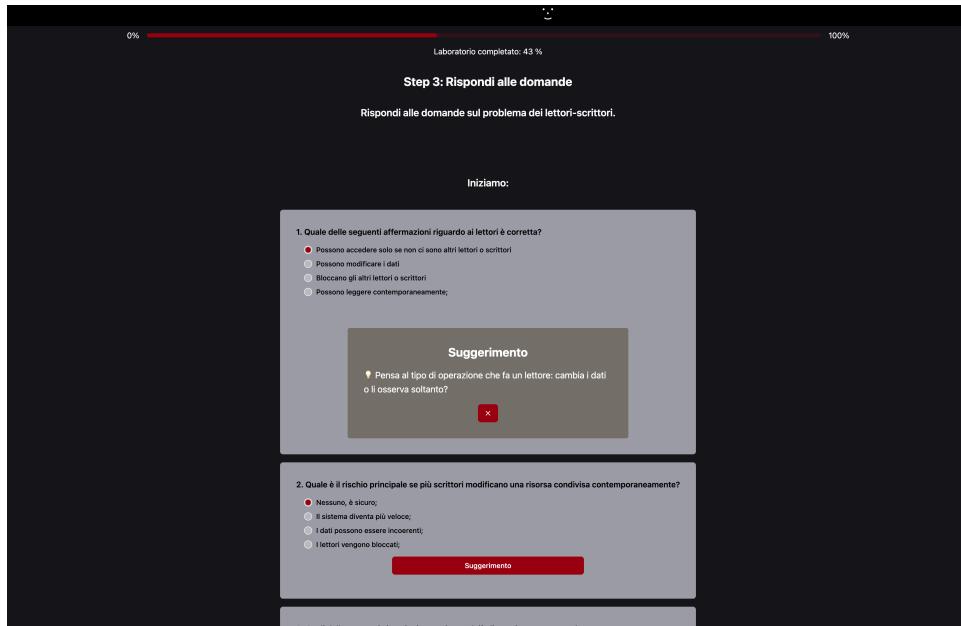


Figura 4.29: Hint di Thinky

4.2.4 Suggerimento

Il suggerimento mostrato sotto ogni domanda teorica è stato creato come un componente aggiuntivo, che viene mostrato dinamicamente in base ad un certo `id` passato come prop al componente stesso. Questo permette di mostrare un suggerimento diverso per ogni domanda, in modo da aiutare gli studenti a rispondere correttamente senza dover creare diversi componenti cambiando il testo.

In particolare, il componente prende come prop l'id della domanda, e poi fa una ricerca nel file `hints.json`, che contiene i suggerimenti per ogni domanda. In questo modo, il suggerimento è facilmente inseribile nella funzione `map` che crea le domande, e viene mostrato solo quando l'utente clicca sul link del suggerimento. Questo permette di mantenere l'interfaccia pulita e di mostrare il suggerimento solo quando necessario, evitando di appesantire la pagina con informazioni non necessarie.

4.2.5 Layout

Come accennato nella sezione dedicata a [NextJS](#), il file `layout.tsx` è il file che definisce la struttura di base dell'applicazione e viene utilizzato per tutte le pagine che fanno riferimento ad esso. Questo file è fondamentale per garantire una struttura coerente e uniforme in tutta l'applicazione, e permette di definire le parti comuni a tutte le pagine, come il menu di navigazione, il footer e altre componenti condivise.

In Thinky, sono stati creati tre file `layout.tsx`:

- Nel file `layout.tsx` principale (Figura 4.30), vengono renderizzate le parti necessarie alla visualizzazione dell'pagina generica di errore, e il link per saltare al contenuto principale. Questo è stato creato per sopperire alla mancanza di un layout nella Root del progetto, che avrebbe portato alla visualizzazione di pagine di errore senza una struttura definita e senza l'applicazione degli stili.

```

1  export default function RootLayout({
2      children,
3  }: Readonly<{
4      children: React.ReactNode;
5  }>) {
6
7      return (
8          <html lang="en" suppressHydrationWarning className="scroll-smooth">
9              <head />
10             <body>
11                 <a
12                     href="#main-content"
13                     tabIndex={0}
14                     className="sr-only focus:not-sr-only
15                     focus:fixed focus:top-4 focus:left-4 focus:z-50
16                     focus:px-4 focus:py-2 focus:bg-white f
17                     ocus:text-black focus:outline-2
18                     focus:outline-blue-500 focus:rounded"
19                     >
20                         Vai al contenuto
21                     </a>
22                     <main id="main-content" tabIndex={-1}>
23                         {children}
24                     </main>
25                 </body>
26             </html>
27         )
28     }

```

Figura 4.30: layout.tsx principale

- Il file `layout.tsx` per la pagina di laboratorio (Figura 4.31), che contiene la struttura specifica per quella pagina. Questo file viene utilizzato per tutte le pagine che fanno riferimento alla pagina di laboratorio, garantendo una struttura coerente e uniforme in tutta l'applicazione. In particolare, data la presenza di pagine con una spiegazione teorica dei problemi che lo studente è chiamato a risolvere, è stato necessario creare un layout per questa pagina che includesse una vista modificata della Navbar, che non permettesse di accedere alle pagine di teoria, di modo che l'utente non potesse accedervi durante lo svolgimento dell'attività.

```

1  export default function LabLayout({
2      children,
3  }: Readonly<{
4      children: React.ReactNode;
5  }>) {
6
7      return (
8          <>
9              <ThemeProvider
10                 attribute="class"
11                 defaultTheme="system"
12                 enableSystem
13                 disableTransitionOnChange
14             >
15                 <UserProvider>
16                     <LabNavbar />
17                     <div id="main-content" tabIndex={-1}>
18                         {children}
19                     </div>
20                     <Footer />
21                 </UserProvider>
22             </ThemeProvider>
23
24         </>
25     )
26 }
27
28
29

```

Figura 4.31: layout.tsx della pagina di laboratorio

- L'ultimo file `layout.tsx` è quello per le altre pagine (Figura 4.32), che contiene la struttura specifica per quelle pagine. Questo file viene utilizzato per tutte le pagine che non fanno riferimento alla pagina di laboratorio, garantendo una struttura coerente e uniforme in tutta l'applicazione. In particolare, questo file contiene la struttura della Navbar e del Footer, che sono presenti in tutte le pagine dell'applicazione.



```
thinky - layout.tsx

1  export default function PageLayout({
2    children,
3  }: Readonly<{
4    children: React.ReactNode;
5  }>) {
6
7    return (
8      <>
9        <ThemeProvider
10          attribute="class"
11          defaultTheme="system"
12          enableSystem
13          disableTransitionOnChange
14        >
15          <UserProvider>
16            <div className="md:hidden block">
17              <Header />
18            </div>
19            <Navbar />
20            <div id="main-content" tabIndex={-1}>
21              {children}
22            </div>
23            <BackToTop />
24            <Footer />
25          </UserProvider>
26        </ThemeProvider>
27
28      </>
29    )
30  }
31
32
33
```

Figura 4.32: layout.tsx delle altre pagine

Si noti come il layout del laboratorio e quello delle pagine generiche abbiano in comune i componenti `<UserProvider>` e `<ThemeProvider>`.

4.2.6 UserProvider

Mentre è stata già spiegata la funzione e l'implementazione di `<ThemeProvider>` nella sezione dedicata a ShadCN, il componente `<UserProvider>` è stato creato per gestire lo stato dell'utente all'interno dell'applicazione, permettendo di accedere alle informazioni dell'utente in modo semplice e veloce. In particolare, questo componente permette di gestire lo stato dell'utente, e di visualizzarne le informazioni in tutti gli altri componenti che sono *wrapped* all'interno di esso.

Lo `<UserProvider>` (Figura 4.33) è quindi incluso nel layout desiderato, e al suo interno sono inclusi tutti i restanti componenti. Nella sua definizione, viene creato il `context` e un hook `useUser` per accedere alle informazioni dell'utente in modo semplice e veloce. Questo permette di evitare di dover passare le informazioni dell'utente come `props` a tutti i componenti, semplificando la gestione dello stato dell'utente all'interno dell'applicazione.



```

thinky - UserContext.tsx

1  export function UserProvider({ children }: { children: React.ReactNode }) {
2      const [user, setUser] = useState<User | null>(null);
3
4      useEffect(() => {
5          const saved = sessionStorage.getItem('user');
6          if (saved) {
7              setUser(JSON.parse(saved));
8          }
9      }, []);
10
11     return (
12         <UserContext.Provider value={{ user, setUser }}>
13             {children}
14         </UserContext.Provider>
15     );
16 }
17
18 export default function useUser() {
19     const context = useContext(UserContext);
20     if (!context) throw new Error('useUser must be used within a UserProvider');
21     return context;
22 }

```

Figura 4.33: UserProvider

`useUser` viene utilizzato in tutti i componenti che hanno bisogno di accedere alle informazioni dell’utente, come ad esempio la Navbar, o le pagine admin.

4.2.7 Avatar

Gli avatar sono stati implementati in Thinky per mostrare le immagini dei profili degli studenti che hanno partecipato al laboratorio nella navbar. Per fare ciò, è stato creato un componente `<Avatar>` che si occupa di visualizzare l’immagine dell’avatar dell’utente, partendo da un componente della libreria ShadCN.

Questo è poi inserito nella Navbar, che utilizzando `useUser` come menzionato sopra, passa le informazioni al componente `<Avatar>` per visualizzare l’immagine dell’avatar dell’utente.

L’immagine è visualizzata in base all’username, che è di due tipi:

- Generato randomicamente in base ad una lista di nomi disponibili.
- Deciso dall’utente.

In ambo i casi, l’immagine dell’avatar viene recuperata dalla cartella `avatar` all’interno della repository di GitHub, cambia però quale immagine viene visualizzata.

Gli avatar sono quindi stati creati con l’intelligenza artificiale in modo da distinguere questi due casi.

Infatti, se l’username viene generato randomicamente, allora seguirà il tipo Animale Colore, quindi la logica del componente reperirà da GitHub l’immagine con il nome corrispondente all’username passato come Prop al componente, e visualizzerà l’immagine corrispondente (Figura 4.34).

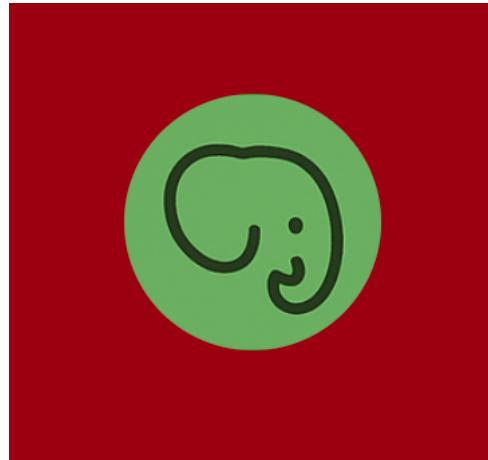


Figura 4.34: Avatar con username generato: Elefante Verde

Se invece l'username viene scelto dall'utente, la logica del componente reperirà da GitHub un'immagine random tra una lista dedicata di avatar (Figura 4.35).

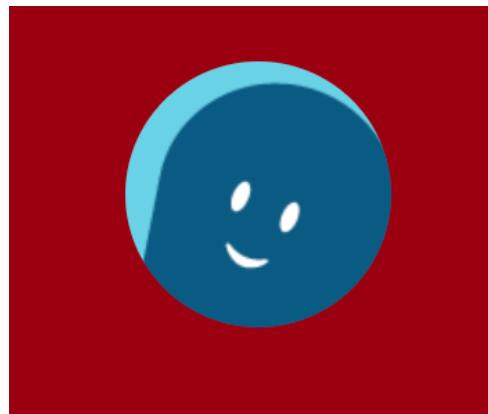


Figura 4.35: Avatar con username scelto: Avatar personalizzato

Inoltre, l'avatar fungerà da menù dropdown (Figura 4.36), che mostrerà all'utente le informazioni inserite in fase di registrazione e un link per visitare la pagina informativa del corso.

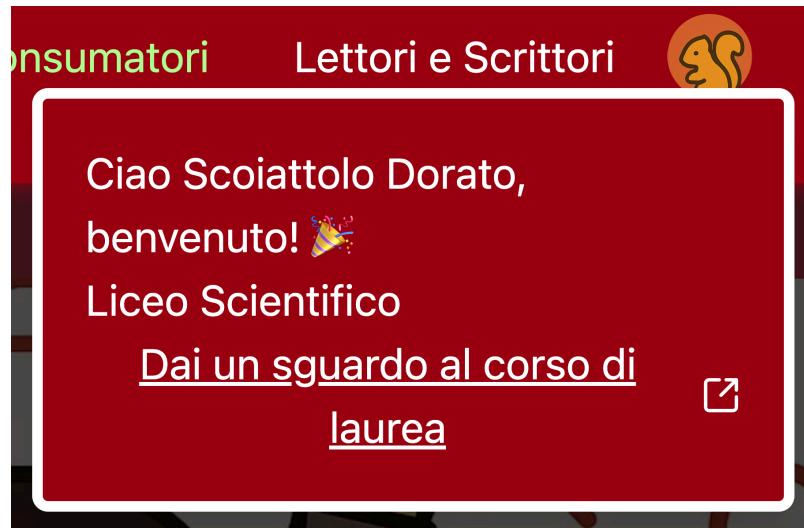


Figura 4.36: Dropdown dell'avatar

Tutte le informazioni degli user sono inoltre persistenti attraverso tutta l'applicazione grazie all'utilizzo del `sessionStorage`. È stato deciso di utilizzare questo tipo di memoria perché non era necessario mantenere le informazioni degli utenti una volta terminato il laboratorio. In questo modo, una volta chiusa la scheda del browser, tutte le informazioni degli utenti vengono cancellate, garantendo così la privacy degli studenti e la sicurezza dei loro dati.

Banner Home

Anche il banner utilizza `<useUSer>` per visualizzare il nome utente in prima pagina, così da rendere l'esperienza utente più personalizzata e accogliente. Il banner è stato creato come un componente a sé stante, che può essere utilizzato in qualsiasi pagina dell'applicazione, che accetta diversi prop:

- `source`: source dell'immagine da visualizzare nel banner.
- `title`: titolo del banner.
- `text`: testo del banner.
- `username`: unico prop opzionale, che permette di visualizzare il nome utente nel banner se esiste.

Inoltre, nel caso in cui il prop `username` venga passato (Figura 4.37), il banner visualizzerà il nome utente all'interno di un tag `<h2>` con un `background-color`

impostato allo stesso colore del nome utente (se del tipo Animale Colore), o al colore di default (se del tipo Avatar personalizzato).

Se invece il prop username non è pervenuto, il banner visualizzerà normalmente tutti gli altri props (Figura 4.38).



Figura 4.37: Banner della Home con username



Figura 4.38: Banner della pagina Produttore-Consumatore

Per fare questo, è bastato prendere il prop username, separare il nome dell'animale dal colore, e applicare il colore come `background-color` del tag `<h2>` che contiene il nome utente. Nel caso in cui l'username sia del tipo Avatar personalizzato, il colore di default è stato impostato a `bg-red-500`, ovvero il rosso di TailwindCSS.

4.2.8 Responsive Design

Tutto il progetto è stato sviluppato tenendo a mente il principio del *responsive design*, ovvero la capacità di adattarsi a diverse dimensioni dello schermo e dispositivi. Questo è stato fatto utilizzando le classi modificatrici di TailwindCSS, che permettono di definire stili specifici per diverse dimensioni dello schermo, garantendo così una buona esperienza utente su tutti i dispositivi.

Nonostante il progetto sia stato sviluppato con la consapevolezza che verrà utilizzato sui computer desktop del laboratorio del Dipartimento di Matematica, far sì che l'applicazione fosse responsive permette di garantire una ottima esperienza a tutti gli utenti se in futuro dovessero utilizzare Thinky su dispositivi diversi, come smartphone o tablet.

4.2.9 Drag and Drop

La funzionalità Drag and Drop (Figura 4.39) che offre il progetto, è il core dell'attività di laboratorio, in quanto permette agli studenti di interagire a livello massimo con l'applicazione stessa, spronandoli al ragionamento e al pensiero critico. Per implementare questa funzionalità, è stata utilizzata la libreria DND Kit, che permette di creare interfacce utente interattive e dinamiche, con la possibilità di trascinare e rilasciare elementi all'interno dell'applicazione.

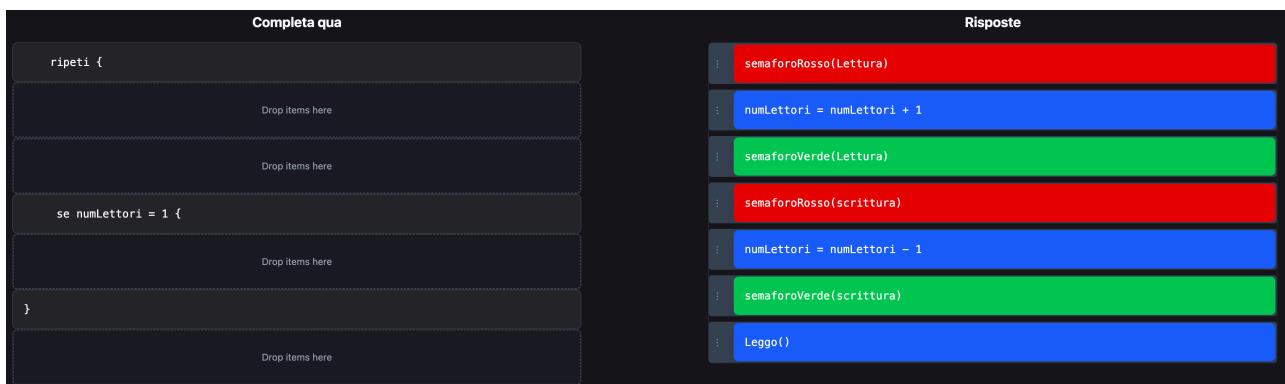


Figura 4.39: Esempio di Drag and Drop

Questa libreria è stata scelta per la sua facilità d'uso e per la sua compatibilità con TailwindCSS, che permette di personalizzare gli stili degli elementi trascinabili in modo semplice e veloce. Inoltre, DND Kit offre di default tutte le caratteristiche di accessibilità compatibili con gli standard WCAG.

Per ottenere un risultato modulare, sono stati creati due componenti principali:

- `DroppableContainer`, che rappresenta l'area in cui gli elementi possono essere trascinati e rilasciati.

- SortableItem, che rappresenta gli elementi che possono essere trascinati e rilasciati all'interno del DroppableContainer.

All'interno del componente `<DroppableContainer>`, viene utilizzato il `useDroppable` di DND Kit per gestire gli eventi di trascinamento e rilascio degli elementi. Questo hook permette di definire le proprietà del contenitore, come ad esempio il tipo di elementi che possono essere trascinati e rilasciati, e le azioni da eseguire quando un elemento viene trascinato o rilasciato. Ogni `<DroppableContainer>` può contenere al più un `<SortableItem>` (Figura 4.40), così da garantire che la risposta sia univoca per ogni container. In figura



```

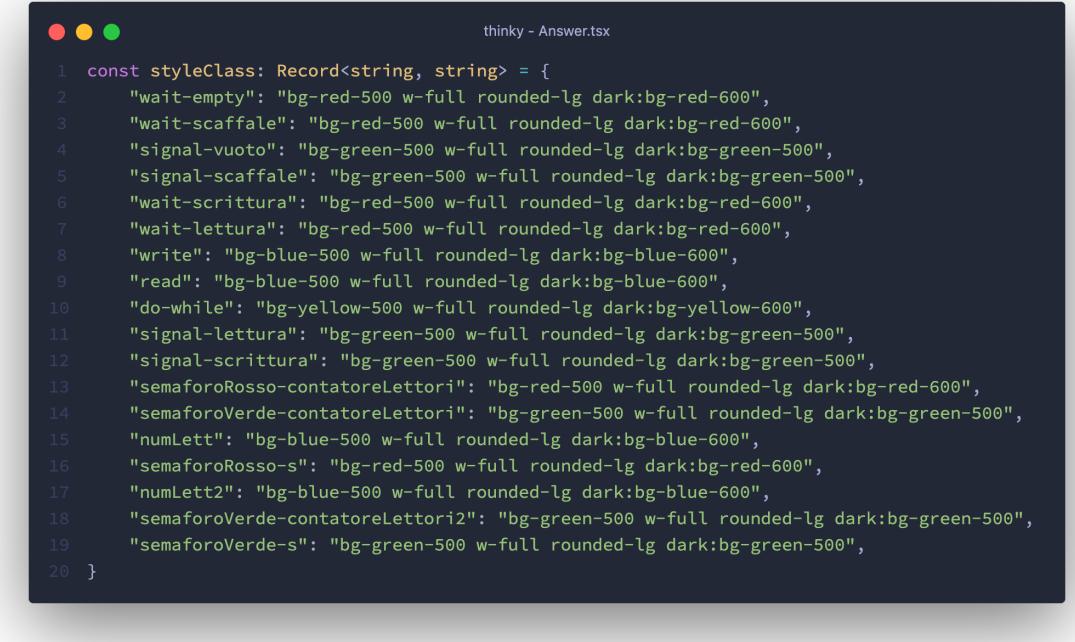
thinky - DroppableContainer.tsx
1 <SortableContext items={items.map((item) => item.id)} strategy={verticalListSortingStrategy}>
2   <ul className="flex flex-col gap-2 w-full">
3     {items.map((item) => (
4       <Answer
5         key={item.id}
6         id={item.id}
7         content={item.content}
8       />
9     ))}
10   </ul>
11 </SortableContext>

```

Figura 4.40: SortableItem

All'interno del componente `<SortableItem>`, viene utilizzato il componente `<Markdown>`, importato dalla libreria `RactMarkdown`, per visualizzare il testo dell'elemento trascinabile. Questo permette di visualizzare il testo formattato per apparire come una stringa di codice. A questo componente, viene poi passato un prop opzionale, comodamente chiamato `className` che permette di passare delle classi di TailwindCSS per personalizzare l'aspetto dell'elemento trascinabile. Questo comportamento è stato utile per cambiare il colore di sfondo dei vari `<SortableItem>` in base al contenuto, per rendere più intuitivo l'esercizio. Ad esempio, i `semaforoVerde()` sono stati colorati di verde, e i `semaforoRosso()` di rosso, mentre le altre istruzioni in blu o arancione.

Dato che però i vari Items erano renderizzati dinamicamente, in base ad un oggetto definito nella logica del componente, non era possibile passare le classi direttamente al componente `<SortableItem>` in fase di renderizzazione. Per sopperire a questo problema, il componente `<SortableItem>` accetta un prop chiamato `id`, che contiene l'id dell'elemento renderizzato. Questo id, viene poi utilizzato per recuperare la classe corrispondente all'elemento all'interno di un dizionario (Figura 4.41), e quella viene passata al componente `<Markdown>` per visualizzare l'elemento con lo stile corretto.



```

thinky - Answer.tsx

1 const styleClass: Record<string, string> = {
2   "wait-empty": "bg-red-500 w-full rounded-lg dark:bg-red-600",
3   "wait-scaffale": "bg-red-500 w-full rounded-lg dark:bg-red-600",
4   "signal-vuoto": "bg-green-500 w-full rounded-lg dark:bg-green-500",
5   "signal-scaffale": "bg-green-500 w-full rounded-lg dark:bg-green-500",
6   "wait-scrittura": "bg-red-500 w-full rounded-lg dark:bg-red-600",
7   "wait-lettura": "bg-red-500 w-full rounded-lg dark:bg-red-600",
8   "write": "bg-blue-500 w-full rounded-lg dark:bg-blue-600",
9   "read": "bg-blue-500 w-full rounded-lg dark:bg-blue-600",
10  "do-while": "bg-yellow-500 w-full rounded-lg dark:bg-yellow-600",
11  "signal-lettura": "bg-green-500 w-full rounded-lg dark:bg-green-500",
12  "signal-scrittura": "bg-green-500 w-full rounded-lg dark:bg-green-500",
13  "semaforoRosso-contatoreLettori": "bg-red-500 w-full rounded-lg dark:bg-red-600",
14  "semaforoVerde-contatoreLettori": "bg-green-500 w-full rounded-lg dark:bg-green-500",
15  "numLett": "bg-blue-500 w-full rounded-lg dark:bg-blue-600",
16  "semaforoRosso-s": "bg-red-500 w-full rounded-lg dark:bg-red-600",
17  "numLett2": "bg-blue-500 w-full rounded-lg dark:bg-blue-600",
18  "semaforoVerde-contatoreLettori2": "bg-green-500 w-full rounded-lg dark:bg-green-500",
19  "semaforoVerde-s": "bg-green-500 w-full rounded-lg dark:bg-green-500",
20 }

```

Figura 4.41: Dizionario delle classi

4.2.10 Tema chiaro e scuro

4.3 Accessibilità

Come già scritto, l'accessibilità è stata una delle priorità principali del progetto Thinky. Si è cercato di garantire che tutti gli studenti potessero utilizzare la piattaforma senza difficoltà, in ogni parte del sistema, dalla più banale alla più complessa. Per raggiungere questo obiettivo, sono state adottate diverse best practices e strumenti per misurare il grado di accessibilità.

Per essere sicuri che l'applicazione fosse accessibile a tutti gli utenti, sono stati seguiti gli standard WCAG (Web Content Accessibility Guidelines), che forniscono linee guida per rendere i contenuti web più accessibili a persone con disabilità. Queste linee guida coprono vari aspetti dell'accessibilità, come la navigazione, la leggibilità, l'uso di colori e contrasti, e l'interazione con i contenuti.

Sono stati utilizzati diversi strumenti per testare l'accessibilità dell'applicazione, tra cui:

- Total validator, messo a disposizione dall'ateneo. È uno strumento che permette di verificare l'accessibilità dei siti web, controllando se rispettano gli standard WCAG. Total Validator esegue una serie di test automatici per identificare eventuali problemi di accessibilità e fornisce un report dettagliato con le problematiche riscontrate e le relative soluzioni.
- Accessible Web Chrome Extension, un'estensione per il browser Google Chrome che permette di scansionare ogni pagina del sito e identificare eventuali problemi, evidenziandoli così da rendere più semplice la loro risoluzione. Questa estensione è molto utile per testare l'accessibilità in tempo reale, mentre si sviluppa l'applicazione, e permette di identificare rapidamente eventuali problemi di accessibilità.
- Screen Reader NVDA_([13]) (lettore di schermo), un software che legge ad alta voce il contenuto di una pagina web, permettendo agli utenti con disabilità visive di interagire con l'applicazione. I lettori di schermo sono uno strumento fondamentale per testare l'accessibilità dell'applicazione, in quanto permettono di verificare se il contenuto è leggibile e comprensibile anche per gli utenti con disabilità visive.

Durante lo sviluppo inoltre, sono state applicate tutte le best practices per garantire l'accessibilità, come ad esempio:

- Utilizzare colori con un contrasto sufficiente tra il testo e lo sfondo, in modo da garantire una buona leggibilità per tutti gli utenti.

- Utilizzare etichette chiare e descrittive per i campi di input, in modo da rendere chiaro il loro scopo e facilitare la comprensione da parte degli utenti.
- Utilizzare testi alternativi per le immagini, in modo da garantire che gli utenti con disabilità visive possano comprendere il contenuto delle immagini.
- Utilizzare una struttura semantica corretta per il markup HTML, in modo da garantire che i lettori di schermo possano interpretare correttamente il contenuto della pagina e navigare facilmente tra le sezioni.
- Utilizzare un linguaggio semplice e chiaro, evitando termini tecnici o complessi.

4.4 Test e compatibilità

Per garantire la consegna di un prodotto funzionante e di massima qualità sono stati svolti molteplici test manuali. Insieme alla Prof.ssa Ombretta Gaggi, è stato deciso di non utilizzare test automatici **sulla code coverage** dato che l'applicazione sarebbe stata testata molteplici volte durante lo sviluppo da più utenti, anche di tipo diverso.

Più precisamente, l'applicazione è stata testata in toto da utenti target, e utenti esperti.

Con utente target si intende la tipologia di utente che si vuole raggiungere con il prodotto, più precisamente il tipo di utente descritto al [capitolo terzo](#).

Con utente esperto invece si intende un utente con conoscenze del prodotto tecniche e teoriche molto avanzate, che sa come funziona l'applicazione stessa.

Far testare l'applicazione solamente ad uno dei due gruppi di utenti avrebbe iniziato il concetto stesso di testing: infatti, far eseguire i test solamente all'utente target avrebbe portato a non considerare le problematiche tecniche e teoriche che un utente esperto avrebbe potuto riscontrare, mentre far eseguire i test solamente all'utente esperto avrebbe portato a non considerare le

problematiche di usabilità e accessibilità che un utente target avrebbe potuto riscontrare.

I test inoltre sono stati effettuati su diversi browser e dispositivi, in modo tale da garantire la massima compatibilità attraverso ogni piattaforma.

I browser testati sono stati:

- Google Chrome
- Mozilla Firefox
- Safari

I dispositivi testati sono stati:

- Computer desktop
- Computer portatili
- Smartphone
- Tablet

Risultato finale dei test

I ripetuti test hanno portato a diverse modifiche nel tempo, che hanno migliorato l'esperienza utente e la stabilità dell'applicazione. Tutto questo ha fatto sì che l'applicazione fosse pronta per essere presentata al pubblico, e che potesse essere utilizzata da tutti gli studenti senza problemi.

I test con utenti target hanno evidenziato come certi aspetti del laboratorio non fossero chiari e che fosse necessario semplificare gli esercizi e le istruzioni, in modo da renderli più comprensibili e facili da seguire. Inoltre, sono stati riscontrati alcuni problemi di usabilità, come ad esempio la difficoltà nel trovare alcune funzionalità o nel comprendere il funzionamento di alcune parti dell'applicazione.

I test con utenti esperti hanno invece evidenziato alcune problematiche tecniche, come ad esempio il contrasto dei colori in alcuni elementi non fosse abbastanza accentuato e come, in particolare nel Drag and Drop, fosse necessario aggiungere dei colori alle istruzioni per rendere più chiaro lo scopo dell'esercizio. Alcuni test hanno anche rilevato mancata compatibilità

con certi dispositivi, più nel particolare quelli Android, e come questi non supportassero alcune funzionalità avanzate come il drag and drop. Questi problemi sono stati risolti con modifiche al codice e all'interfaccia utente, in modo da garantire la massima compatibilità e usabilità su tutti i dispositivi e browser.

Capitolo 5

Conclusioni

Riassume le considerazioni finali, riassumendo gli scopi, le scelte e le lezioni apprese, e propone possibili sviluppi futuri.

5.1 Lo scopo e le scelte

Thinky è un progetto di stage universitario nato con l’obiettivo di sostituire i software attualmente utilizzati per le attività di OpenDay dell’Università di Padova, creando una WebApp interattiva proprietaria dell’ateneo con cui gli studenti potessero approcciarsi al mondo dell’informatica risolvendo alcuni problemi canonici, sotto la guida dei professori del corso.

Il progetto vuole inoltre essere un monito per gli studenti che si trovano a dover scegliere un percorso di studi non in linea con il loro curriculum da studente, dimostrando come anche senza conoscenze specifiche, ma con la voglia di imparare e di mettersi in gioco, si possa intraprendere un percorso da informatico.

La scelta di realizzare un progetto di questo tipo è stata per me importante, in quanto io stesso mi sono trovato a scegliere questo percorso di studi senza una pregressa conoscenza del mondo dell’informatica, e senza sapere se fosse la scelta giusta per me. Mi è sembrato quindi importante realizzare un progetto che possa essere utile a tutti gli studenti futuri che si vogliono “gettare” in questo mondo ma insicuri perché mancano di conoscenze specifiche.

Tutte le scelte atte durante lo svolgimento del progetto sono state tutte mirate non solo al raggiungimento degli obiettivi prefissati, e quindi al completamento del mio percorso (accademico e di stage), ma anche a garantire che un

prodotto di qualità fosse consegnato, e soprattutto con la consapevolezza che la qualità che il prodotto che ho realizzato avrebbe avuto un impatto positivo su studenti che, come me in passato, si trovano a dover compiere una scelta tutt'altro che semplice, in un campo in cui la competenza, ma non la voglia di fare e le conoscenze, viene meno.

Scegliere un percorso di studi, un lavoro, o qualsiasi altra attività, è un passo importante, e spesso difficile da compiere. Per questo motivo, ho cercato di realizzare un prodotto che potesse essere utile a chiunque si trovasse in questa situazione.

Con questo progetto, e di conseguenza le scelte (tecniche e non), ho cercato di rappresentare concretamente questa difficoltà, spingendomi in ogni cosa al di fuori della mia *comfort zone*, adottando tecnologie e pratiche con cui non avevo mai avuto a che fare, o con le quali non avevo esperienza. Questo ha portato ad un prodotto che, sebbene potrebbe non risultare perfetto, è stato realizzato con cura e attenzione ai dettagli, e con lo scopo di dimostrare che tutto è possibile, se l'impegno è abbastanza forte.

5.2 Copertura dei requisiti

Alla fine dello sviluppo del progetto, è stato effettuato un controllo della copertura dei requisiti, per verificare che tutti i quelli definiti fossero stati implementati correttamente. La copertura dei requisiti è stata calcolata come rapporto tra il numero di requisiti implementati e il numero totale di requisiti definiti.

Considerando NR il numero totale di requisiti definiti e NI il numero di requisiti implementati, la copertura dei requisiti è stata calcolata come segue:

$$\text{Copertura Requisiti} = \left(\frac{\text{NR}}{\text{NI}} \right) * 100 \quad (5.1)$$

Dove:

- NR = 9, obiettivi richiesti, come dal capitolo [Obiettivi del progetto](#).
- NI = 9, obiettivi effettivamente implementati.

È stata raggiunta una copertura del 100%, in quanto tutti i requisiti definiti sono stati implementati correttamente. Questo garantisce che il sistema soddisfi le aspettative minime e le esigenze degli utenti, come definito nei casi d'uso e nelle user story, oltre che ad avere tutti gli “extra” menzionati.

5.3 Sviluppi futuri

Come mostrato nella sezione precedente, il progetto ha raggiunto tutti gli obiettivi prefissati, e quindi non sono previsti sviluppi futuri a terminazione di questo progetto. Tuttavia, ci sono sicuramente funzionalità aggiuntive che potrebbero essere implementate in futuro, per migliorare ulteriormente l'esperienza utente e le funzionalità del sistema, come ad esempio:

- Implementazione di un sistema di feedback per raccogliere le opinioni degli utenti e migliorare il sistema in base alle loro esigenze.
- Aggiunta di funzionalità di personalizzazione del profilo utente.
- Nuovi problemi da risolvere, per rendere il laboratorio sempre più interessante e stimolante per gli studenti.

5.4 Valutazioni personali

Durante queste settimane di stage, ho avuto l'opportunità di lavorare su un progetto che mi ha permesso di mettere in pratica le conoscenze acquisite durante il mio percorso di studi, e di imparare nuove tecnologie e pratiche che non avevo mai affrontato prima.

Ho imparato molto e non solamente dal punto di vista tecnico, ma anche a livello personale.

Ho imparato ad organizzarmi, a gestire il mio tempo e a lavorare in modo efficace ed efficiente.

Ho imparato a non essere frettoloso, a non avere paura di chiedere aiuto quando ne ho bisogno, e a non arrendermi di fronte alle difficoltà.

Ho avuto la fortuna di avere una Tutor che mi ha supportato e guidato durante tutto il percorso, e che mi ha permesso di crescere e migliorare, mantenendo sempre alta la qualità del lavoro svolto.

Infine, ritengo che il prodotto realizzato sia di qualità, e che possa essere considerato un ottimo strumento per le attività di OpenDay dell’Università di Padova, e per gli studenti che si trovano a dover scegliere un percorso di studi.

Glossario

API: Interfaccia di programmazione delle applicazioni. Permette la comunicazione tra diverse applicazioni o servizi. 15, 56

Backend as a Service: Backend as Service. Servizio che fornisce un backend completo per le applicazioni, inclusi database, autenticazione e hosting, consentendo agli sviluppatori di concentrarsi sullo sviluppo del frontend.

59

Bottom-up: Approccio Bottom-up. Approccio di sviluppo in cui si parte da una base di codice già esistente e lo si estende, piuttosto che partire da zero. Questo approccio consente di riutilizzare il codice esistente e di costruire su di esso. 54

Firebase: Piattaforma di sviluppo di applicazioni web e mobili che offre servizi come database in tempo reale, autenticazione e hosting. 13

Framework: Framework. Insieme integrato di componenti software e strumenti che forniscono una struttura di base per lo sviluppo di applicazioni. I framework semplificano e accelerano il processo di sviluppo, fornendo funzionalità predefinite e convenzioni da seguire. 54

Git: Sistema di controllo versione distribuito. Permette di tenere traccia delle modifiche apportate a file e cartelle nel tempo. 3

GitHub: Piattaforma di hosting per progetti software che utilizza Git come sistema di controllo versione. Permette la collaborazione tra sviluppatori e la gestione del codice sorgente. 3, 13, 56

Kanban: Sistema di gestione del lavoro che utilizza schede per visualizzare il flusso di lavoro e le attività in corso. 4

Piano di Lavoro: Piano di lavoro. Documento che descrive le attività e gli obiettivi di un tirocinio o di un progetto. 3

ReactJS: Libreria JavaScript per la creazione di interfacce utente. Permette di costruire applicazioni web complesse e reattive in modo efficiente. 53

Scrum: Framework Agile per la gestione dei progetti. Si basa su iterazioni brevi (sprint) e su riunioni regolari per monitorare i progressi. 5

Top-down: Approccio Top-down. Approccio di sviluppo in cui si fornisce una struttura architettonica di base su cui costruire l'applicazione, piuttosto che partire da zero e costruire tutto da capo. Questo approccio consente di avere una visione d'insieme del progetto fin dall'inizio. 54

UX: Esperienza utente. Rappresenta l'esperienza complessiva di un utente nell'interagire con un prodotto o un servizio, in questo caso Thinky. 49

WebApp: abbr. Applicazione Web iv

accessibilità: Accessibilità. Pratica di progettazione e sviluppo che mira a rendere i contenuti e le funzionalità di un sito web o di un'applicazione utilizzabili da tutti, comprese le persone con disabilità. 6

backlog: Backlog. Elenco di attività o funzionalità da completare in un progetto. In questo caso, si riferisce all'elenco delle funzionalità e dei bug da risolvere nel progetto. 3

blueprint: Progetto di base. In questo caso, si riferisce al file layout.tsx che fornisce la struttura principale a tutte le pagine dell'applicazione. 65

client: Client. In un contesto di rete, il client è un dispositivo o un software, solitamente un browser o un'applicazione, che richiede servizi o risorse da un server. Il client invia richieste al server e riceve risposte contenenti i dati richiesti. 56

hook: Hook. Funzione speciale in React che consente di utilizzare lo stato e altre funzionalità senza dover scrivere una classe. Gli hook semplificano la gestione dello stato e degli effetti collaterali nelle applicazioni React. 86

open source: Open source. Software il cui codice sorgente è reso disponibile al pubblico, consentendo a chiunque di utilizzarlo, modificarlo e distribuirlo liberamente. 62

proponente: Colui che propone un progetto o un'idea. In questo caso, si riferisce all'Università di Padova 11

props: Proprietà. In React, le props sono un modo per passare dati e funzioni dai componenti genitori ai componenti figli. L'equivalente dei parametri per le funzioni in altri linguaggi. I props consentono di personalizzare il comportamento e l'aspetto dei componenti. 86

query: Query. In informatica, una query è una richiesta di informazioni da un database o da un sistema di gestione dei dati. Le query vengono utilizzate per recuperare, inserire, aggiornare o eliminare dati in un database. 57

responsive: Responsive. Design reattivo. Approccio al design web che consente a un sito web di adattarsi a diverse dimensioni di schermo e dispositivi, garantendo una buona esperienza utente su desktop, tablet e smartphone. 52

server: Server. In un contesto di rete, il server è un dispositivo o un software che fornisce servizi o risorse ai client. Il server elabora le richieste dei client e restituisce le risposte contenenti i dati richiesti. 56

server-side component: Componenti lato server. Permettono di eseguire il rendering delle pagine sul server prima di inviarle al client, migliorando le prestazioni e l'ottimizzazione SEO. 60

sprint: Sprint. Periodo di tempo definito in cui un team di sviluppo (o in questo caso, un solo dev) lavora per completare un insieme specifico di attività o obiettivi. 3

tooltip: Tooltip. Piccola finestra di testo che appare quando si passa il mouse su un elemento, fornendo informazioni aggiuntive o suggerimenti sull'elemento stesso. 73

utility-first: Utility-first. Approccio di sviluppo CSS in cui si utilizzano classi predefinite per applicare stili specifici agli elementi HTML, semplificando la creazione di interfacce utente reattive e personalizzabili. 69

Bibliografia

- [1] «Manifesto tetet Agile Software Development». [Online]. Disponibile su: <http://agilemanifesto.org/>
- [2] «TypeScript Documentation». [Online]. Disponibile su: <https://www.typescriptlang.org/docs/>
- [3] «Axios Documentation». [Online]. Disponibile su: <https://axios-http.com/docs/intro>
- [4] «Firebase Documentation». [Online]. Disponibile su: <https://firebase.google.com/docs>
- [5] «GitHub Documentation». [Online]. Disponibile su: <https://docs.github.com/en>
- [6] «ShadCN Documentation». [Online]. Disponibile su: <https://ui.shadcn.com/docs>
- [7] «Next.js Documentation». [Online]. Disponibile su: <https://nextjs.org/docs>
- [8] «Server-Side Components». [Online]. Disponibile su: <https://react.dev/reference/rsc/server-components>
- [9] «Client-Side Components». [Online]. Disponibile su: <https://react.dev/reference/rsc/use-client>
- [10] «TailwindCSS Documentation». [Online]. Disponibile su: <https://tailwindcss.com/docs>
- [11] «Lucide Icons Documentation». [Online]. Disponibile su: <https://lucide.dev/>
- [12] «DND Kit Documentation». [Online]. Disponibile su: <https://docs.dndkit.com/>

[13] «NVDA Documentation». [Online]. Disponibile su: <https://www.nvda.it/node/242>