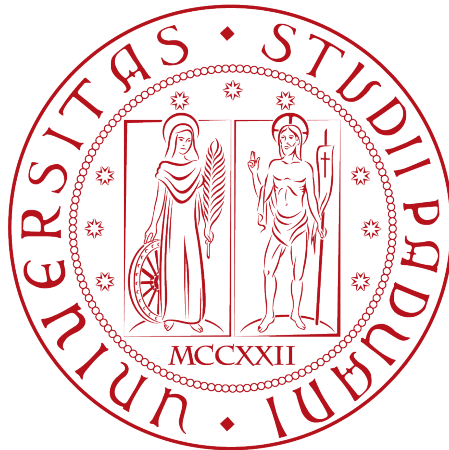


**Università degli Studi di Padova**

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA "

CORSO DI LAUREA IN INFORMATICA



**Front-end di un'applicazione web per la  
distribuzione di materiali didattici**

*Tesi di laurea triennale*

*Relatore*

Prof. Luigi De Giovanni

*Laureando*

Nicola Abbagnato

---

ANNO ACCADEMICO 2018-2019



# Sommario

Questo documento descrive il lavoro svolto durante il periodo di stage avvenuto presso Moku S.r.l. di Roncade, della durata di circa trecento ore, finalizzato alla realizzazione del frontend, in Angular, di una web-app dedicata al supporto del progetto Sports Open Schools (SOS).

L'applicazione ha tre tipologie di utenti, essi sono educatori, che creano unità didattiche (*teaching unit*), professori, che creano delle *classi* a cui associano le *teaching unit* create, e studenti, che fanno parte delle *classi*.

Lo scopo è quindi fornire un flusso di creazione e distribuzione di queste unità.

Dovendo essere utilizzata da utenti che non sono necessariamente familiari con il mondo digitale, inoltre, è stata posta particolare importanza alla semplicità ed intuitività dell'applicazione.



# Ringraziamenti

*Ringrazio i miei genitori, i miei amici e tutti coloro che mi hanno supportato durante i momenti difficili e non.*

*Ringrazio chi lavora presso Moku S.r.l. dato che, nei due mesi di stage, ma anche dopo, mi hanno fatto sentire parte di una seconda famiglia.*

*Padova, Dicembre 2019*

Nicola Abbagnato



# Indice

<b>1</b>	<b>L'azienda</b>	<b>1</b>
1.1	Descrizione generale . . . . .	1
1.2	Metodologie utilizzate . . . . .	1
<b>2</b>	<b>Descrizione dello stage</b>	<b>3</b>
2.1	Obiettivi dello stage . . . . .	3
2.1.1	Notazione . . . . .	3
2.1.2	Obiettivi fissati . . . . .	3
2.2	Introduzione al progetto . . . . .	4
2.3	Pianificazione del lavoro . . . . .	4
2.3.1	Pianificazione iniziale . . . . .	4
2.3.2	Variazioni rispetto alla pianificazione iniziale . . . . .	5
<b>3</b>	<b>Analisi dei requisiti</b>	<b>7</b>
3.1	Funzionalità dell'applicazione . . . . .	7
3.2	Casi d'uso . . . . .	8
3.2.1	UC1: Registrazione . . . . .	8
3.2.2	UC2: Log-in . . . . .	10
3.2.3	UC2.1: Visualizzazione messaggio dati errati . . . . .	10
3.2.4	UC3: Reset password . . . . .	11
3.2.5	UC4: Conferma email . . . . .	11
3.2.6	UC5: Visualizzazione profilo . . . . .	11
3.2.7	UC6: Modifica profilo . . . . .	12
3.2.8	UC7: Logout . . . . .	12
3.2.9	UC8: Visualizzazione lista <i>teaching unit</i> . . . . .	12
3.2.10	UC9: Visualizzazione dettagli <i>teaching unit</i> . . . . .	13
3.2.11	UC10: Visualizzazione <i>teaching unit</i> . . . . .	13
3.2.12	UC10.1: Visualizzazione carosello . . . . .	13
3.2.13	UC11: Visualizzazione forum studenti . . . . .	14
3.2.14	UC12: Creazione <i>teaching unit</i> . . . . .	14
3.2.15	UC12.1: Visualizzazione messaggio d'errore . . . . .	15
3.2.16	UC13: Eliminazione <i>teaching unit</i> . . . . .	15
3.2.17	UC14: Pubblicazione <i>teaching unit</i> . . . . .	15
3.2.18	UC14.1: Visualizzazione messaggio d'errore . . . . .	15
3.2.19	UC15: Modifica <i>teaching unit</i> . . . . .	15
3.2.20	UC15.1: Modifica contenuti <i>teaching unit</i> . . . . .	16
3.2.21	UC15.1: Modifica sezione carosello . . . . .	16
3.2.22	UC16: Visualizzazione forum insegnanti . . . . .	17

3.2.23	UC17: Visualizzazione topic studenti . . . . .	17
3.2.24	UC18: Visualizzazione topic insegnanti . . . . .	18
3.2.25	UC19: Rimozione risposta topic . . . . .	18
3.2.26	UC20: Creazione questionario . . . . .	18
3.2.27	UC21: Modifica questionario . . . . .	19
3.2.28	UC22: Pubblicazione questionario . . . . .	19
3.2.29	UC23: Eliminazione questionario . . . . .	19
3.2.30	UC24: Visualizzazione dettagli questionario . . . . .	20
3.2.31	UC25: Visualizzazione risposte questionario . . . . .	20
3.2.32	UC26: Creazione topic studente . . . . .	20
3.2.33	UC27: Risposta a topic studente. . . . .	21
3.2.34	UC28: Creazione <i>classe</i> . . . . .	21
3.2.35	UC29: Visualizzazione <i>classe</i> . . . . .	21
3.2.36	UC30: Assegnazione <i>teaching unit</i> ad una <i>classe</i> . . . . .	22
3.2.37	UC31: Compilazione questionario . . . . .	22
3.3	Tracciamento requisiti . . . . .	22
3.3.1	Requisiti funzionali . . . . .	23
3.3.2	Requisiti qualitativi . . . . .	24
3.3.3	Requisiti di vincolo . . . . .	25
<b>4</b>	<b>Sviluppo</b>	<b>27</b>
4.1	Tecnologie utilizzate . . . . .	27
4.2	Progettazione . . . . .	29
4.2.1	L'architettura di angular . . . . .	29
4.2.2	Comunicazione con il back-end . . . . .	29
4.2.3	L'architettura dell'applicazione . . . . .	30
4.3	Implementazione . . . . .	31
4.3.1	Shared . . . . .	31
4.3.2	Core . . . . .	34
4.3.3	ClassPage . . . . .	36
4.3.4	ClassesListPage . . . . .	37
4.3.5	EditSurvey . . . . .	38
4.3.6	EditTeachingUnit . . . . .	38
4.3.7	ForgotPassword . . . . .	39
4.3.8	Profile . . . . .	39
4.3.9	ProfileEmailConfirm . . . . .	39
4.3.10	SignIn . . . . .	39
4.3.11	SignUp . . . . .	40
4.3.12	StudentForum . . . . .	40
4.3.13	Survey . . . . .	40
4.3.14	TeachingUnitPageTeacher . . . . .	40
4.3.15	TeachingUnitPageTrainer . . . . .	41
4.3.16	TeachingUnitsList . . . . .	41
4.3.17	Topic . . . . .	42
4.3.18	ViewFilledSurvey . . . . .	43
4.3.19	ViewSurveys . . . . .	43
4.3.20	ViewUnit . . . . .	43
4.4	Problemi di implementazione riscontrati . . . . .	44
4.4.1	Drag and Drop multidirezione . . . . .	44
4.4.2	Youtube video flickering . . . . .	45



<i>INDICE</i>	ix
4.4.3 Breadcrumbs tracking . . . . .	45
<b>5 Conclusioni</b>	<b>47</b>
5.1 Resoconto degli obiettivi raggiunti . . . . .	47
5.2 Valutazione dell'esperienza svolta . . . . .	48
<b>Riferimenti bibliografici e sitografici</b>	<b>51</b>

# Elenco delle figure

2.1	Diagramma di Gantt della pianificazione del lavoro . . . . .	5
3.1	Casi d'uso . . . . .	9
3.2	Casi d'uso . . . . .	10
4.1	L'architettura di Angular [1] . . . . .	30
4.2	CardPlaceholderComponent appearance . . . . .	32
4.3	DeleteWarningComponent appearance . . . . .	32
4.4	PreviewTeachingUnitComponent appearance . . . . .	34
4.5	ClassPage appearance . . . . .	36
4.6	ClassesListPage appearance . . . . .	37
4.7	CreateClassDialogComponent appearance . . . . .	38
4.8	SignIn appearance . . . . .	40
4.9	SignUp appearance . . . . .	41
4.10	TeachingUnitPageTeacher appearance . . . . .	42
4.11	TeachingUnitPageTrainer appearance . . . . .	43
4.12	TeachingUnitsList appearance . . . . .	44
4.13	ViewFilledSurvey appearance . . . . .	45

# Elenco delle tabelle

2.1	Tabella delle variazioni della pianificazioni . . . . .	6
3.1	Tabella requisiti funzionali . . . . .	24
3.2	Tabella requisiti qualitativi . . . . .	25
3.3	Tabella requisiti di vincolo . . . . .	25

5.1	Tabella degli obiettivi raggiunti a fine stage . . . . .	47
5.2	Tabella delle variazioni della pianificazioni . . . . .	48



# Capitolo 1

## L'azienda

In questo capitolo viene descritta l'azienda presso la quale si è svolto lo stage e le metodologie di lavoro utilizzate per i suoi progetti.

### 1.1 Descrizione generale

Moku è nata come start-up nel 2013, supportata dall'incubatore H-Farm, con alla base un progetto omonimo.

Dopo aver abbandonato tale progetto, l'azienda attualmente si occupa di sviluppo di software su commissione, design grafico e consulenza in ambito IT; ciò che la caratterizza di più però è la metodologia di lavoro, a stretto contatto con il cliente, indipendentemente dalla sua dimensione.

Moku è composta da 12 membri i cui ruoli si dividono in questo modo:

- \* CEO: si occupa di gestire l'azienda dal lato amministrativo oltre che sviluppare relazioni con partner e clienti più importanti;
- \* CTO: si occupa di tenere aggiornato lo stack tecnologico utilizzato dall'azienda;
- \* CFO: si occupa della gestione finanziaria dell'azienda;
- \* Analista: si occupa dello studio dei requisiti e del dominio applicativo;
- \* Web designer: si occupa dell'ideazione dell'applicazione come vista dall'utente;
- \* Sviluppatore: si occupa della codifica dell'applicazione in modo che rispetti i requisiti analizzati e rispecchi il design ideato.

### 1.2 Metodologie utilizzate

Moku utilizza metodologie *agile* [13] al fine di sviluppare software su misura a stretto contatto con il cliente, assicurandosi di creare un prodotto in grado di soddisfare ed eventualmente superare le aspettative.

In particolare, Moku adotta il framework SCRUM [12] che aiuta fortemente nei casi dove risulta difficile pianificare in anticipo, ad esempio quando si ha a che fare con clienti che non sono abituati ad avere a che fare con il mondo dello sviluppo software.

SCRUM si basa su tre principi:

- \* **Trasparenza:** gli aspetti significativi del processo devono essere visibili ai responsabili del lavoro. La trasparenza richiede che quegli aspetti siano definiti da uno standard comune in modo tale che gli osservatori condividano una comune comprensione di ciò che viene visto.
- \* **Ispezione:** chi utilizza Scrum deve ispezionare frequentemente i progressi realizzati verso il conseguimento degli obiettivi prestabiliti, individuando in tal modo precocemente eventuali difformità rispetto a quanto si intende realizzare. La frequenza delle ispezioni non deve essere tale da determinare un'interruzione del lavoro in corso. Le ispezioni devono essere eseguite diligentemente e da ispettori qualificati.
- \* **Adattamento:** se chi ispeziona verifica che uno o più aspetti del processo di produzione sono al di fuori dei limiti accettabili e che il prodotto finale non potrà essere accettato, deve intervenire sul processo stesso o sul materiale prodotto dalla lavorazione. L'intervento deve essere portato a termine il più rapidamente possibile per ridurre al minimo l'ulteriore scarto rispetto agli obiettivi prestabiliti.

Il framework Scrum prevede di dividere il progetto in blocchi rapidi di lavoro, chiamati Sprint, creando un incremento del software al termine di ciascuno di essi.

Ogni sprint, che dura da una a quattro settimane, è preceduto da una riunione di pianificazione in cui vengono definiti gli obiettivi e stimati i tempi; questi obiettivi non possono essere cambiati durante lo sprint, ma soltanto alla successiva riunione di pianificazione.

Nel corso di ogni sprint, il team crea porzioni complete di un prodotto. L'insieme delle funzionalità che vengono inserite in un determinato sprint provengono dal product backlog, che è una lista ordinata di requisiti.

I requisiti del product backlog che vengono scelti come obiettivo per un determinato sprint costituiscono lo sprint backlog. [12]

Moku utilizza il framework Scrum con sprint generalmente lunghi da 10 a 15 giorni; alla fine di ciascuno sprint, viene valutato il progresso raggiunto e pianificato il successivo, stabilendo quali requisiti utilizzare per costituire lo sprint backlog.

Oltre a SCRUM, altre metodologie *agile* utilizzate consistono nel rilascio frequente di versioni intermedie del prodotto e nella particolare attenzione ad avere una comunicazione costante e diretta tra i membri dello stesso team di sviluppo.

## Capitolo 2

# Descrizione dello stage

In questo capitolo verrà descritto lo stage, partendo dagli obiettivi, passando per l'analisi dei requisiti svolta e finendo con la pianificazione del lavoro antecedente lo stage, con i cambiamenti che essa ha subito.

### 2.1 Obiettivi dello stage

Seguono gli obiettivi identificati nel piano di lavoro.

#### 2.1.1 Notazione

Si farà riferimento agli obiettivi secondo le seguenti notazioni:

- \* O per gli obiettivi obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- \* D per gli obiettivi desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- \* F per gli obiettivi facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da un numero sequenziale, identificativo dell'obiettivo.

#### 2.1.2 Obiettivi fissati

- \* Obbligatori:
  - Obbligatori
    - \* O01: studio e formazione sulle tecnologie utilizzate (AngularJS, GraphQL, Apollo framework etc...)
    - \* O02: redazione dell'analisi dei requisiti.

- \* O03: progettazione dell'applicazione frontend secondo i requisiti trovati durante la fase di analisi;
- \* O04: implementazione dell'applicazione frontend;
- \* O05: test del funzionamento dell'applicazione realizzata
- Desiderabili
  - \* D01: aggiunta di funzionalità di comunicazione in tempo reale;
  - \* D02: creazione suite di testing frontend;
  - \* D03: redazione documentazione completa sull'applicazione.
- Facoltativi
  - \* F01: realizzazione di ulteriori funzionalità che potrebbero emergere nella fase di analisi.

## 2.2 Introduzione al progetto

Il progetto consiste nello sviluppo, da parte di Moku, di un'applicazione web per il CUS (Centro Universitario Sportivo) Padova al fine di supportare il progetto SOS (Sports Open Schools). L'obiettivo del progetto SOS consiste nello sviluppo di un metodo di insegnamento innovativo per gli insegnanti di educazione fisica delle scuole superiori e quindi implementare questo nuovo modello nelle 4 scuole partner situate in Italia, Ungheria, Portogallo e Romania [17].

Per raggiungere questo obiettivo, sono state definite quattro tipologie di utenti che l'applicazione deve riconoscere:

Amministratore piattaforma, Educatore, Insegnante, Studente.

Il flusso del progetto SOS consiste nella redazione di specifici moduli didattici multimediali (*teaching unit*) da parte di utenti specifici dell'applicazione, gli educatori; una volta che questi moduli verranno pubblicati, i professori delle scuole partner potranno accedervi ed assegnarli alle proprie *classi*.

Gli studenti infine faranno parte di queste *classi* e potranno consultare questi moduli. Funzionalità importanti dell'applicazione consistono inoltre nella presenza di forum separati per insegnanti/educatori e studenti oltre che la possibilità di somministrare questionari sia a studenti che insegnanti per valutare l'andamento del progetto.

## 2.3 Pianificazione del lavoro

Questa sezione mostra in quale modo è stato organizzato il lavoro durante lo stage, come questa organizzazione si sia poi scostata dalla pianificazione iniziale ed infine le motivazioni per cui questi scostamenti sono avvenuti.

### 2.3.1 Pianificazione iniziale

Anteriormente all'inizio dello stage è stata svolta un'operazione di pianificazione del lavoro da svolgere assieme all'azienda, il risultato di questa è stato poi presentato nel piano di lavoro necessario al fine di iniziare lo stage.

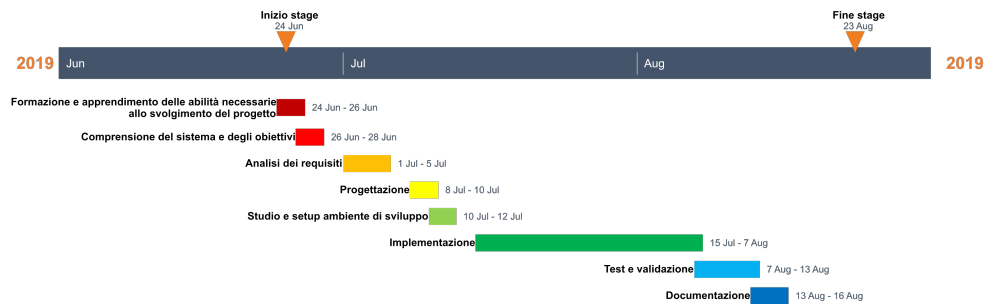
Il lavoro da svolgere è stato inizialmente suddiviso nel seguente modo:

- \* la prime 20 ore, mezza settimana circa, sono dedicate alla formazione necessaria all'utilizzo delle tecnologie e framework da utilizzare per il progetto;



- \* le successive 20 ore, per concludere la prima settimana, saranno dedicate alla comprensione del prodotto da realizzare e agli obiettivi da raggiungere nei due mesi di stage;
- \* la settimana successiva (40 ore) è dedicata all'analisi dei requisiti, con lo scopo di ottenere una lista finale delle funzionalità e caratteristiche del prodotto finito;
- \* nella terza settimana, le prime 20 ore, sono dedicate alla progettazione del prodotto;
- \* le successive 20 ore invece sono dedicate al setup dell'ambiente di sviluppo in modo da poter iniziare l'effettivo sviluppo a partire dalla settimana successiva;
- \* a partire dalla quarta settimana 150 ore (poco meno di quattro settimane) sono dedicate allo sviluppo e all'implementazione del prodotto;
- \* 30 (mezza settimana più il termine della settimana precedente) sono dedicate al testare il prodotto nella sua completezza e ad assicurarsi che rispetti i requisiti trovati in precedenza;
- \* le ultime 20 ore sono dedicate alla redazione della documentazione relativa al progetto.

È stata tenuta in considerazione, inoltre, una settimana aggiuntiva in caso di imprevisti. La Figura 2.1 rappresenta il diagramma di Gantt presentato nel piano di lavoro:



**Figura 2.1:** Diagramma di Gantt della pianificazione del lavoro

### 2.3.2 Variazioni rispetto alla pianificazione iniziale

Il primo scostamento rispetto alla pianificazione si è verificato prima ancora dell'inizio dello stage, per problemi burocratici infatti lo stage è dovuto iniziare la settimana successiva rispetto a quanto pianificato, è stata spostata quindi anche la data di fine dello stage in avanti di una settimana, al 31 di agosto.

Il secondo cambiamento di questo tipo è stato dovuto al fatto che precedentemente all'inizio dello stage non era stato comunicato che per una settimana ad agosto l'azienda sarebbe rimasta chiusa; questo ha portato all'utilizzo della nona settimana, pianificata per imprevisti o problemi vari.

Mentre la prima settimana di formazione e comprensione del problema è andata come pianificato, l'analisi dei requisiti, pianificata per la settimana successiva ha subito un grosso intoppo: il cliente non aveva ancora riflettuto su alcune importanti funzionalità

della webapp, ad esempio la composizione o visualizzazione dei contenuti delle *teaching unit*.

Si è scelto quindi di lasciare in sospeso alcune funzionalità e iniziare con la progettazione e lo sviluppo di quelle già complete.

Quando poi i clienti sono stati in grado di darci una risposta per terminare le funzionalità in sospeso abbiamo concluso l'analisi e osservato che: non solo la progettazione già effettuata si è rivelata adatta ad includere, senza cambiamenti importanti, le "nuove" funzionalità ma anche che questa situazione non ha causato particolari ritardi rispetto alla tabella di marcia (Tabella 2.1).

Attività	Durata pianificata nel piano di lavoro	Durata ripianificata
Formazione e apprendimento delle abilità necessarie allo svolgimento del progetto	20 ore	20 ore
Comprensione del sistema e degli obiettivi	20 ore	20 ore
Analisi dei requisiti	40 ore	32 ore
Progettazione	20 ore	12 ore
Studio e setup ambiente di sviluppo	20 ore	20 ore
Termine progettazione e analisi dei requisiti	0 ore	16 ore
Implementazione	150 ore	150 ore
Test e validazione	30 ore	30 ore
Documentazione	20 ore	20 ore
<b>Totale</b>	<b>320 ore</b>	<b>320 ore</b>

**Tabella 2.1:** Tabella delle variazioni della pianificazioni

## Capitolo 3

# Analisi dei requisiti

Questo capitolo contiene l'analisi dei requisiti effettuata durante lo svolgimento dello stage.

### 3.1 Funzionalità dell'applicazione

Le attività che le tipologie di utenti della piattaforma possono effettuare tramite l'applicazione sono le seguenti:

- \* gli amministratori:
  - creano utenti educatori e insegnanti;
  - vedono i dati di monitoraggio della piattaforma;
- \* gli educatori:
  - definiscono i propri moduli formativi contenenti le seguenti tipologie di dati:
    - \* il *pilastr*o di appartenenza del modulo, questi *pilastr*i sono gli argomenti attorno ai quali si costruisce il modulo, limitati alle seguenti opzioni:
      - *Fairplay*,
      - *Life skills*,
      - *Physical activities and health*;
    - \* una descrizione;
    - \* testi inerenti la pratica o la teoria dell'argomento trattato;
    - \* foto;
    - \* video (link youtube);
  - definiscono questionari insegnanti e studenti (risposte multiple chiuse);
  - partecipano al forum, rispondendo a domande degli insegnanti;
- \* gli insegnanti:
  - partecipano al forum;
  - vedono tutti i moduli formativi;
  - rispondono alle schede di monitoraggio;

- hanno più *classi* alla cui creazione viene generato un codice che viene dato agli studenti per potersi iscrivere e effettuare l'associazione;
- \* gli studenti:
  - possono iscriversi alla piattaforma utilizzando il proprio *codice classe*;
  - rispondono alle schede di monitoraggio;
  - possono partecipare al forum:
    - \* il forum è legato ai moduli formativi a cui hanno partecipato ed è pubblico;
    - \* il forum consente una messaggistica semplice (solo testo);
    - \* il forum è separato da quello educatori / insegnanti.

## 3.2 Casi d'uso

In questa sezione saranno elencati i casi d'uso principali dell'applicazione, senza entrare particolarmente nel dettaglio se non per quelli non banali. Sono stati identificati cinque attori principali nell'applicazione:

- \* Utente non autenticato;
- \* Utente autenticato;
- \* Educatore;
- \* Insegnante;
- \* Studente.

La suddivisione è tale in quanto educatore, insegnante e studente sono i tre tipi di utente dell'applicazione; questi attori derivano poi da utente autenticato che partecipa in tutti i casi d'uso rappresentanti le funzionalità dell'applicazione garantite a tutti gli utenti indipendentemente dal tipo.

Infine l'utente non autenticato è presente per rappresentare tutte le funzionalità dell'applicazioni presenti precedentemente all'identificazione dell'utente e della sua tipologia.

La Figure 3.1 e 3.2 consistono nella rappresentazione grafica dei casi d'uso; questi sono stati separati per semplificare la visualizzazione i casi d'uso a cui può partecipare più di una categoria di utente (Figura 3.2).

### 3.2.1 UC1: Registrazione

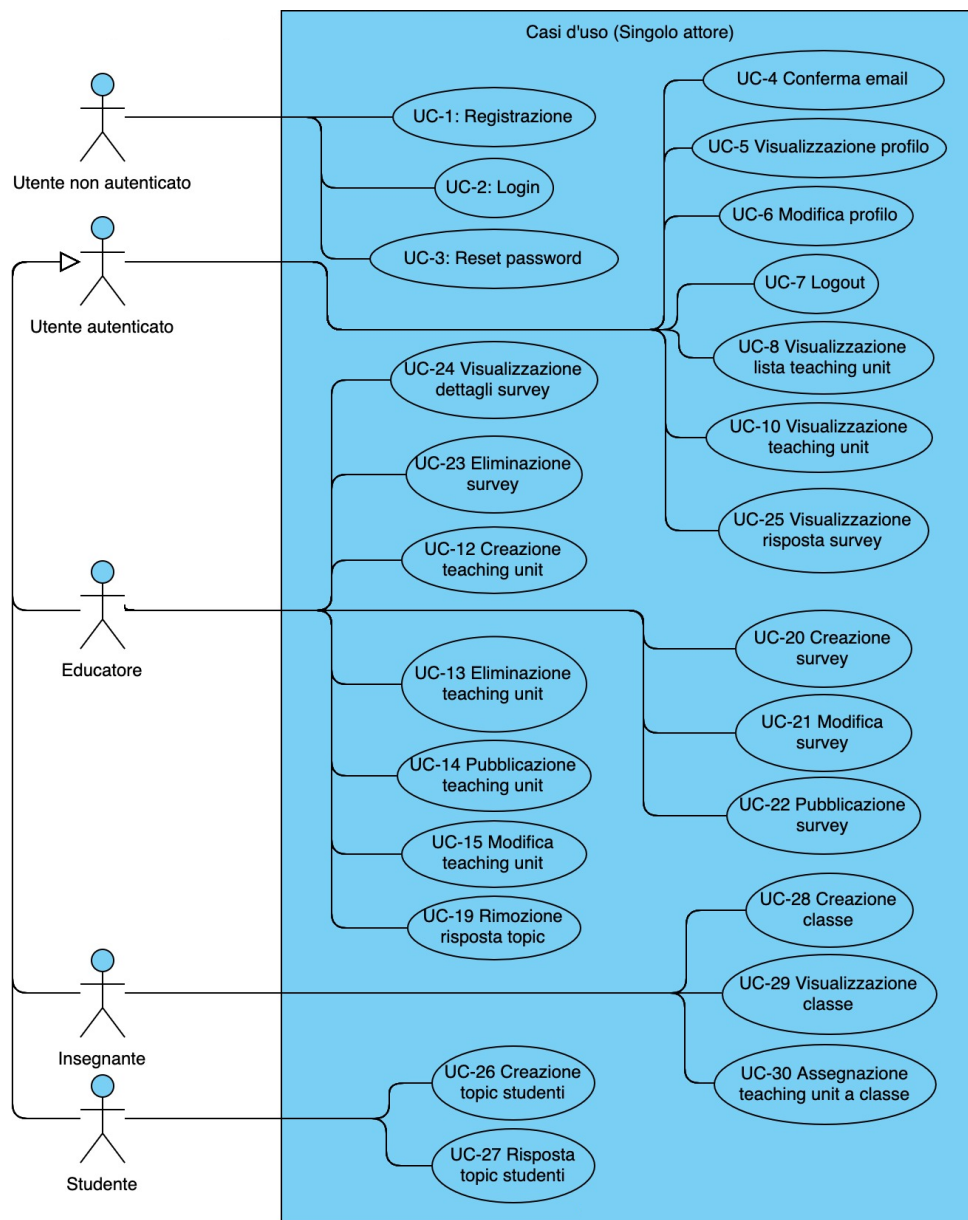
**Attori primari:** Utente non autenticato.

**Descrizione:** Un utente non autenticato inserisce i propri dati e viene registrato come studente.

**Precondizioni:** Un utente non autenticato si trova nella pagina di registrazione.

**Scenario principale:**

- \* L'utente inserisce la propria mail;
- \* L'utente inserisce il proprio nome;



**Figura 3.1:** Casi d'uso

- \* L'utente inserisce la propria data di nascita;
- \* L'utente inserisce la propria password;
- \* L'utente inserisce il codice della propria *classe*;
- \* L'utente conferma i dati inseriti.

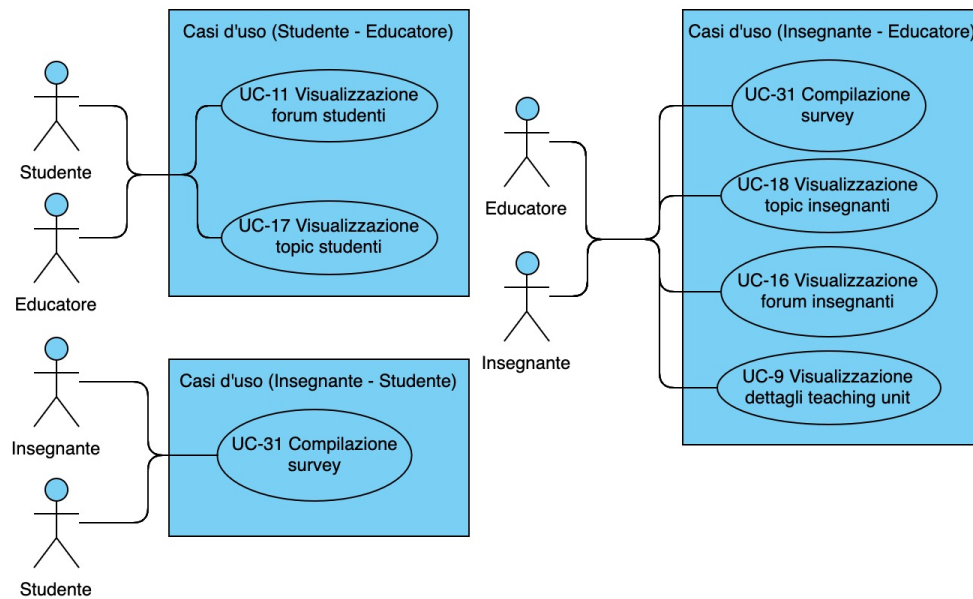


Figura 3.2: Casi d'uso

**Postcondizioni:** L'utente viene registrato, inserito nella *classe* relativa al codice inserito e viene automaticamente effettuato il primo log-in.

### 3.2.2 UC2: Log-in

**Attori primari:** Utente non autenticato.

**Descrizione:** Un utente non autenticato inserisce i propri dati per effettuare il login.

**Precondizioni:** Un utente non autenticato si trova nella pagina di login.

**Scenario principale:**

- \* L'utente inserisce la propria mail;
- \* L'utente inserisce la propria password;
- \* L'utente conferma i dati inseriti.

**Scenario alternativo:** L'utente inserisce dei dati errati (UC2.1: Visualizzazione messaggio dati errati).

**Postcondizioni:** Se i dati sono corretti l'utente effettua il login.

### 3.2.3 UC2.1: Visualizzazione messaggio dati errati

**Attori primari:** Utente non autenticato.

**Descrizione:** Un utente non autenticato ha provato ad effettuare il login con dei dati errati.

**Precondizioni:** Un utente non autenticato ha provato ad effettuare il login con dei dati errati.

**Scenario principale:** Viene visualizzato un messaggio d'errore che informa l'utente

dell'errore commesso.

**Postcondizioni:** L'utente è a conoscenza di aver inserito dati errati.

### 3.2.4 UC3: Reset password

**Attori primari:** Utente non autenticato.

**Descrizione:** Un utente non autenticato effettua il reset della password.

**Precondizioni:** Un utente non autenticato si trova nella pagina di login.

**Scenario principale:**

- \* L'utente richiede il reset della password;
- \* L'utente inserisce la propria mail;
- \* L'utente torna nell'applicazione utilizzando il link trovato nella mail ricevuta;
- \* L'utente inserisce la nuova password;
- \* L'utente conferma i dati inseriti.

**Postcondizioni:** La password dell'utente associato alla mail inserita viene aggiornata.

### 3.2.5 UC4: Conferma email

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato conferma la propria mail.

**Precondizioni:** Un utente autenticato effettua il login senza prima aver confermato la propria mail.

**Scenario principale:**

- \* L'utente richiede il reinvio della mail di conferma;
- \* L'utente accede all'applicazione dal link presente nella mail di conferma.

**Postcondizioni:** La mail dell'utente viene confermata.

### 3.2.6 UC5: Visualizzazione profilo

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato visualizza i dati associati all'account.

**Precondizioni:** Un utente autenticato entra nella pagina del profilo.

**Scenario principale:**

- \* L'utente visualizza la propria mail;
- \* L'utente visualizza il proprio nome.

**Postcondizioni:** L'utente ha visualizzato i propri dati.

### 3.2.7 UC6: Modifica profilo

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato modifica i dati del proprio account.

**Precondizioni:** Un utente autenticato entra nella pagina del profilo.

**Scenario principale:**

- \* L'utente sostituisce la propria mail con un'altra mail;
- \* L'utente sostituisce il proprio nome con un'altro nome;
- \* L'utente inserisce la propria password ed una nuova password;
- \* L'utente conferma i dati inseriti.

**Postcondizioni:** I dati dell'account dell'utente vengono aggiornati.

### 3.2.8 UC7: Logout

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato effettua il logout.

**Precondizioni:** Un utente autenticato si trova all'interno dell'applicazione.

**Scenario principale:** L'utente richiede di effettuare l'operazione di logout.

**Postcondizioni:** L'utente ha effettuato il logout e non è più riconosciuto dalla piattaforma.

### 3.2.9 UC8: Visualizzazione lista *teaching unit*

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato visualizza la lista di *teaching unit* ad esso associate (tutte quelle pubblicate nel caso l'utente sia un professore, solo quelle da lui create nel caso sia un educatore e solo quelle a cui partecipa la propria *classe* nel caso sia uno studente).

**Precondizioni:** Un utente si trova nella pagina delle *teaching unit*.

**Scenario principale:**

- \* L'utente visualizza l'immagine di preview delle *teaching unit* associate;
- \* L'utente visualizza il nome delle *teaching unit* associate;
- \* L'utente visualizza il *pilastro* di cui fanno parte le *teaching unit* associate;
- \* L'utente (se un professore) visualizza sue *classi* che partecipano alle *teaching unit* associate;
- \* L'utente (se un educatore) visualizza se una *teaching unit* è già stata pubblicata o meno.

**Postcondizioni:** L'utente ha visualizzato la lista delle *teaching unit* ad esso associate.



### 3.2.10 UC9: Visualizzazione dettagli *teaching unit*

**Attori primari:** Insegnante, educatore.

**Descrizione:** Un insegnante o un educatore visualizzano i dettagli associati ad una *teaching unit*.

**Precondizioni:** Un insegnante o un professore hanno selezionato una *teaching unit*.

**Scenario principale:**

- \* L'utente visualizza il nome della *teaching unit*;
- \* L'utente visualizza l'immagine di preview della *teaching unit*;
- \* L'utente visualizza la descrizione della *teaching unit*;
- \* L'utente visualizza il *pilastro* di cui fa parte la *teaching unit*;
- \* L'utente visualizza la data di creazione della *teaching unit*;
- \* L'utente visualizza la data di pubblicazione della *teaching unit*;
- \* L'utente visualizza quali *classi* partecipano a tale unità;
- \* L'utente visualizza quanti studenti partecipano a tale unità.

**Postcondizioni:** L'utente ha visualizzato i dettagli della *teaching unit*.

### 3.2.11 UC10: Visualizzazione *teaching unit*

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato visualizza i contenuti di una *teaching unit* selezionata.

**Precondizioni:** Un utente autenticato ha scelto una *teaching unit* della quale visualizzare i contenuti.

**Scenario principale:**

- \* L'utente visualizza il nome della *teaching unit*;
- \* L'utente visualizza eventuali paragrafi presenti nella *teaching unit*;
- \* L'utente visualizza eventuali caroselli presenti nella *teaching unit* (UC10.1: Visualizzazione carosello).

**Postcondizioni:** L'utente ha visualizzato i contenuti della *teaching unit*.

### 3.2.12 UC10.1: Visualizzazione carosello

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato visualizza il contenuto di un carosello presente in una *teaching unit*.

**Precondizioni:** Un utente autenticato sta visualizzando i contenuti di una *teaching unit* che include un carosello.

**Scenario principale:**

- \* L'utente visualizza il testo della slide, se si tratta di una slide di testo;

- \* L'utente visualizza l'immagine della slide, se si tratta di una slide immagine;
- \* L'utente visualizza il video della slide, se si tratta di una slide video;
- \* L'utente visualizza la didascalia della slide, se si tratta di una slide immagine o video;
- \* L'utente passa alla slide successiva (se presente);
- \* L'utente passa alla slide precedente (se presente).

**Postcondizioni:** L'utente ha visualizzato i contenuti del carosello in questione.

### 3.2.13 UC11: Visualizzazione forum studenti

**Attori primari:** Studente, educatore.

**Descrizione:** Uno studente o un educatore visualizzano la lista dei topic presenti nel forum studenti.

**Precondizioni:** Uno studente o un educatore si trovano all'interno della piattaforma.

**Scenario principale:**

- \* L'utente visualizza il titolo di ogni topic studenti;
- \* L'utente visualizza il numero di risposte di ogni topic studenti;
- \* L'utente visualizza la data dell'ultima risposta di ogni topic studenti.

**Postcondizioni:** L'utente ha visualizzato la lista dei topic presenti nel forum studenti.

### 3.2.14 UC12: Creazione *teaching unit*

**Attori primari:** Educatore.

**Descrizione:** Un utente educatore crea una nuova *teaching unit*.

**Precondizioni:** L'utente si trova all'interno della piattaforma.

**Scenario principale:**

- \* L'utente inserisce il nome della *teaching unit*;
- \* L'utente seleziona di quale *pilastro* fa parte la *teaching unit* (physical activity and health, fairplay e employability skills);
- \* L'utente conferma l'inserimento dei dati.

**Scenario alternativo:** L'utente visualizza un messaggio d'errore nel caso non abbia inserito un nome per la *teaching unit* (UC12.1: Visualizzazione messaggio d'errore).

**Postcondizioni:** L'utente ha creato una nuova *teaching unit* in bozza.

### 3.2.15 UC12.1: Visualizzazione messaggio d'errore

**Attori primari:** Educatore.

**Descrizione:** L'utente visualizza un messaggio d'errore che lo informa del fatto che l'inserimento del nome della *teaching unit* è necessario per la creazione di quest'ultima.

**Precondizioni:** L'utente ha provato a creare una nuova *teaching unit* senza inserirne il nome.

**Scenario principale:** L'utente visualizza il messaggio d'errore.

**Postcondizioni:** L'utente è ora a conoscenza del fatto che il nome di una *teaching unit* è obbligatorio alla sua creazione.

### 3.2.16 UC13: Eliminazione *teaching unit*

**Attori primari:** Educatore.

**Descrizione:** L'utente elimina una sua *teaching unit* in bozza.

**Precondizioni:** L'utente ha selezionato la *teaching unit* da eliminare.

**Scenario principale:** L'utente elimina la *teaching unit*.

**Postcondizioni:** L'utente ha eliminato la sua *teaching unit*.

### 3.2.17 UC14: Pubblicazione *teaching unit*

**Attori primari:** Educatore.

**Descrizione:** L'utente pubblica una sua *teaching unit* in bozza.

**Precondizioni:** L'utente ha selezionato una sua *teaching unit* da pubblicare.

**Scenario principale:** L'utente pubblica la *teaching unit* in questione.

**Scenario secondario:** L'utente, se la *teaching unit* non è terminata, visualizza un messaggio d'errore informandolo che la *teaching unit* deve essere completata (UC14.1: Visualizzazione messaggio d'errore).

**Postcondizioni:** L'utente ha pubblicato la *teaching unit*.

### 3.2.18 UC14.1: Visualizzazione messaggio d'errore

**Attori primari:** Educatore. **Descrizione:** L'educatore visualizza un messaggio d'errore che lo informa che la *teaching unit* che sta provando a pubblicare non è terminata.

**Precondizioni:** L'educatore ha provato a pubblicare una *teaching unit* in bozza senza aver inserito tutti i campi necessari.

**Scenario principale:** L'educatore visualizza un messaggio d'errore che lo informa che la *teaching unit* che sta provando a pubblicare non è terminata.

**Postcondizioni:** L'utente è stato informato dell'assenza di informazioni necessarie al fine di pubblicare la *teaching unit*.

### 3.2.19 UC15: Modifica *teaching unit*

**Attori primari:** Educatore.

**Descrizione:** Un educatore modifica una sua *teaching unit* in bozza.

**Precondizioni:** L'utente ha selezionato una *teaching unit* da modificare.

**Scenario principale:**

- \* L'utente inserisce un nuovo nome per la *teaching unit*;
- \* L'utente seleziona un nuovo *pilaastro* all'interno del quale si trova la *teaching unit*;
- \* L'utente inserisce una nuova descrizione per la *teaching unit*;
- \* L'utente modifica i contenuti della *teaching unit* (UC15.1: Modifica contenuti *teaching unit*);
- \* L'utente conferma i nuovi dati inseriti.

**Postcondizioni:** L'utente ha modificato la *teaching unit*.

### 3.2.20 UC15.1: Modifica contenuti *teaching unit*

**Attori primari:** Educatore.

**Descrizione:** L'utente modifica i contenuti di una *teaching unit*.

**Precondizioni:** L'utente ha selezionato una *teaching unit* da modificare.

**Scenario principale:**

- \* L'utente inserisce una nuova sezione paragrafo;
- \* L'utente inserisce una nuova sezione carosello;
- \* L'utente cambia la posizione di una sezione;
- \* L'utente elimina una sezione;
- \* L'utente cambia il testo di una sezione paragrafo;
- \* L'utente modifica una sezione carosello (UC15.1: Modifica sezione carosello);
- \* L'utente salva le modifiche effettuate.

**Postcondizioni:** L'utente ha modificato i contenuti di una *teaching unit*.

### 3.2.21 UC15.1: Modifica sezione carosello

**Attori primari:** Educatore.

**Descrizione:** L'utente modifica una sezione carosello all'interno di una *teaching unit*.

**Precondizioni:** L'utente ha selezionato un carosello da modificare.

**Scenario principale:**

- \* L'utente inserisce una nuova slide testo;
- \* L'utente inserisce una nuova slide immagine;
- \* L'utente inserisce una nuova slide video;
- \* L'utente cambia l'ordine delle slide;
- \* L'utente elimina una slide;
- \* L'utente cambia il testo di una slide testo;

- \* L'utente cambia l'immagine di una slide immagine;
- \* L'utente cambia il video di una slide video;
- \* L'utente cambia la didascalia di una slide immagine o video;
- \* L'utente cambia il tipo di una slide.

**Postcondizioni:** L'utente ha modificato i contenuti del carosello in questione.

### 3.2.22 UC16: Visualizzazione forum insegnanti

**Attori primari:** Educatore, insegnante.

**Descrizione:** Un educatore o insegnante visualizzano la lista di topic presenti nel forum insegnanti, separati in base alla *teaching unit* a cui sono assegnati.

**Precondizioni:** L'utente ha selezionato una *teaching unit* di cui vedere i topic creati.

**Scenario principale:**

- \* L'utente visualizza il titolo di ogni topic relativo alla *teaching unit*;
- \* L'utente visualizza il numero di risposte di ogni topic relativo alla *teaching unit*;
- \* L'utente visualizza la data dell'ultima risposta di ogni topic relativo alla *teaching unit*.

**Postcondizioni:** L'utente ha visualizzato la lista dei topic associati alla *teaching unit* in questione.

### 3.2.23 UC17: Visualizzazione topic studenti

**Attori primari:** Studente, educatore.

**Descrizione:** Uno studente o un educatore ha selezionato un topic studenti e ne visualizza il contenuto e le risposte.

**Precondizioni:** L'utente in questione ha selezionato un topic studenti.

**Scenario principale:**

- \* L'utente visualizza il nome dello studente che ha pubblicato il topic;
- \* L'utente visualizza la nazionalità dello studente che ha pubblicato il topic;
- \* L'utente visualizza la data di pubblicazione del topic;
- \* L'utente visualizza il contenuto del topic;
- \* L'utente visualizza il nome di ogni studente che ha risposto al topic;
- \* L'utente visualizza la nazionalità di ogni studente che ha risposto al topic;
- \* L'utente visualizza la data di pubblicazione di ogni risposta al topic;
- \* L'utente visualizza il contenuto di ogni risposta al topic.

**Postcondizioni:** L'utente ha visualizzato i contenuti e risposte del topic studente.

### 3.2.24 UC18: Visualizzazione topic insegnanti

**Attori primari:** Insegnante, educatore.

**Descrizione:** Un insegnante o un educatore ha selezionato un topic insegnanti e ne visualizza il contenuto e le risposte.

**Precondizioni:** L'utente in questione ha selezionato un topic insegnanti.

**Scenario principale:**

- \* L'utente visualizza il nome dello utente che ha pubblicato il topic;
- \* L'utente visualizza la nazionalità dello utente che ha pubblicato il topic;
- \* L'utente visualizza la data di pubblicazione del topic;
- \* L'utente visualizza il contenuto del topic;
- \* L'utente visualizza il nome di ogni utente che ha risposto al topic;
- \* L'utente visualizza la nazionalità di ogni utente che ha risposto al topic;
- \* L'utente visualizza la data di pubblicazione di ogni risposta al topic;
- \* L'utente visualizza il contenuto di ogni risposta al topic.

**Postcondizioni:** L'utente ha visualizzato i contenuti e risposte del topic insegnanti.

### 3.2.25 UC19: Rimozione risposta topic

**Attori primari:** Educatore.

**Descrizione:** Un educatore rimuove una risposta all'interno di un topic studente o insegnante.

**Precondizioni:** Un educatore sta visualizzando un topic.

**Scenario principale:**

- \* L'utente seleziona una risposta da eliminare dal topic;
- \* L'utente elimina tale risposta.

**Postcondizioni:** L'utente ha rimosso la risposta selezionata del topic in questione.

### 3.2.26 UC20: Creazione questionario

**Attori primari:** Educatore.

**Descrizione:** Un educatore crea un questionario che dovrà essere compilata da insegnanti o studenti.

**Precondizioni:** L'utente si trova all'interno dell'applicazione.

**Scenario principale:**

- \* L'utente seleziona la tipologia di questionario da creare (insegnanti o studenti);
- \* L'utente inserisce il titolo del questionario da creare.

**Postcondizioni:** L'utente ha creato un nuovo questionario in bozza.

### 3.2.27 UC21: Modifica questionario

**Attori primari:** Educatore.

**Descrizione:** Un educatore modifica un questionario in bozza.

**Precondizioni:** L'utente deve aver selezionato un questionario in bozza da modificare.

**Scenario principale:**

- \* L'utente modifica il titolo del questionario;
- \* L'utente inserisce una nuova domanda con risposta aperta;
- \* L'utente inserisce una nuova domanda chiusa con risposta multipla;
- \* L'utente inserisce una nuova domanda chiusa con risposta singola;
- \* L'utente cambia la tipologia di una delle domande;
- \* L'utente cambia l'ordine delle domande;
- \* L'utente cambia il testo di una delle domande;
- \* L'utente inserisce una nuova risposta ad una domanda chiusa;
- \* L'utente cambia il testo di una delle risposte alle domande chiuse;
- \* L'utente rimuove una risposta ad una delle domande chiuse;
- \* L'utente elimina una delle domande;
- \* L'utente salve le modifiche effettuate.

**Postcondizioni:** L'utente ha modificato il questionario in bozza selezionato.

### 3.2.28 UC22: Pubblicazione questionario

**Attori primari:** Educatore.

**Descrizione:** Un educatore pubblica un questionario in bozza.

**Precondizioni:** L'utente ha selezionato un questionario in bozza da pubblicare.

**Scenario principale:** L'utente pubblica il questionario in questione.

**Postcondizioni:** L'utente ha pubblicato il questionario in questione.

### 3.2.29 UC23: Eliminazione questionario

**Attori primari:** Educatore.

**Descrizione:** Un educatore elimina un questionario in bozza.

**Precondizioni:** L'utente ha selezionato un questionario in bozza da eliminare.

**Scenario principale:** L'utente elimina il questionario in bozza selezionato.

**Postcondizioni:** L'utente ha eliminato il questionario in questione.

### 3.2.30 UC24: Visualizzazione dettagli questionario

**Attori primari:** Educatore.

**Descrizione:** Un educatore visualizza i dettagli di un questionario pubblicato.

**Precondizioni:** L'utente ha selezionato un questionario pubblicato di cui vedere i dettagli.

**Scenario principale:**

- \* L'utente visualizza il titolo del questionario;
- \* L'utente visualizza la data di pubblicazione del questionario;
- \* L'utente visualizza il testo di ogni domanda del questionario;
- \* L'utente visualizza le possibili risposte di ogni domanda chiusa del questionario.

**Postcondizioni:** L'utente ha visualizzato i dettagli del questionario.

### 3.2.31 UC25: Visualizzazione risposte questionario

**Attori primari:** Utente autenticato.

**Descrizione:** Un utente autenticato visualizza le risposte date ad un questionario, le proprie nel caso l'utente sia un insegnante o uno studente quelle degli altri utenti nel caso si tratti di un educatore.

**Precondizioni:** L'utente seleziona la quale risposta (compilazione) di un questionario vuole visualizzare.

**Scenario principale:**

- \* L'utente visualizza il titolo del questionario;
- \* L'utente visualizza il testo delle domande del questionario;
- \* L'utente visualizza il testo delle risposte delle domande chiuse;
- \* L'utente visualizza quali risposte delle domande chiuse sono state selezionate;
- \* L'utente visualizza le risposte alle domande aperte.

**Postcondizioni:** L'utente ha visualizzato le risposte al questionario.

### 3.2.32 UC26: Creazione topic studente

**Attori primari:** Studente.

**Descrizione:** Uno studente crea un nuovo topic studente.

**Precondizioni:** L'utente si trova all'interno dell'applicazione.

**Scenario principale:**

- \* L'utente inserisce il testo del topic da creare;
- \* L'utente conferma i dati inseriti.

**Postcondizioni:** L'utente ha creato un nuovo topic studente.



**3.2.33 UC27: Risposta a topic studente.**

**Attori primari:** Studente.

**Descrizione:** Uno studente risponde ad un topic studente.

**Precondizioni:** L'utente ha selezionato il topic studente a cui rispondere.

**Scenario principale:**

- \* L'utente inserisce il testo della risposta da creare;
- \* L'utente conferma i dati inseriti.

**Postcondizioni:** L'utente ha risposto al topic in questione.

**3.2.34 UC28: Creazione *classe***

**Attori primari:** Insegnante.

**Descrizione:** Un insegnante crea una nuova *classe*.

**Precondizioni:** L'utente si trova all'interno della piattaforma.

**Scenario principale:**

- \* L'utente inserisce il nome della nuova *classe*;
- \* L'utente conferma i dati inseriti;
- \* L'utente visualizza il join code della *classe* appena creata.

**Postcondizioni:** L'utente ha creato una nuova *classe*.

**3.2.35 UC29: Visualizzazione *classe***

**Attori primari:** Insegnante.

**Descrizione:** Un insegnante visualizza i dettagli di una propria *classe*.

**Precondizioni:** L'utente ha selezionato la *classe* da visualizzare.

**Scenario principale:**

- \* L'utente visualizza il nome della *classe*;
- \* L'utente visualizza il nome degli studenti iscritti alla *classe*;
- \* L'utente visualizza il numero degli studenti iscritti alla *classe*;
- \* L'utente visualizza quanti studenti iscritti alla *classe* hanno completato l'ultimo questionario studenti pubblicata;
- \* L'utente visualizza quante *teaching unit* sono state assegnate alla *classe*;
- \* L'utente visualizza il join code della *classe*;
- \* L'utente visualizza quali *teaching unit* sono state assegnate alla *classe*.

**Postcondizioni:** L'utente ha visualizzato i dettagli della *classe* selezionata.

### 3.2.36 UC30: Assegnazione *teaching unit* ad una *classe*

**Attori primari:** Insegnante.

**Descrizione:** Un insegnante assegna una nuova *teaching unit* ad una delle proprie *classi*.

**Precondizioni:** L'insegnante ha selezionato la *classe* alla quale aggiungere la *teaching unit*.

**Scenario principale:**

- \* L'utente seleziona una *teaching unit* non assegnata alla *classe* in questione;
- \* L'utente conferma la *teaching unit* selezionata.

**Postcondizioni:** L'utente ha assegnato la *teaching unit* selezionata alla *classe* in questione.

### 3.2.37 UC31: Compilazione questionario

**Attori primari:** Insegnante, studente.

**Descrizione:** Un insegnante o uno studente compilano uno dei rispettivi questionari.

**Precondizioni:** L'utente ha selezionato il questionario da compilare.

**Scenario principale:**

- \* L'utente visualizza il titolo del questionario;
- \* L'utente visualizza il testo delle domande del questionario;
- \* L'utente visualizza il testo delle risposte delle domande chiuse del questionario;
- \* Per ogni domanda aperta l'utente scrive una risposta;
- \* Per ogni domanda chiusa a risposta singola l'utente seleziona una risposta;
- \* Per ogni domanda chiusa a risposta multipla l'utente seleziona una o più risposte;
- \* L'utente conferma le risposte che ha dato.

**Postcondizioni:** L'utente ha compilato il questionario selezionato.

## 3.3 Tracciamento requisiti

Successivamente all'analisi dei requisiti ed alla stesura dei casi d'uso effettuate nel progetto sono stati tracciati nelle successive tabelle i requisiti identificati.

I requisiti possono essere delle seguenti tipologie:

- \* **Funzionali:** legati alle funzionalità offerte dal sistema;
- \* **Prestazionali:** legati alle prestazioni del sistema;
- \* **Qualitativi:** rappresentanti standard e metriche da seguire per garantire la qualità del sistema;
- \* **di Vincolo:** i requisiti dipendenti da fattori esterni legali o di dominio (in questo progetto assenti).

In molti casi è indicata anche un'importanza dei requisiti identificata dalle parole:

- \* obbligatorio,
- \* desiderabile,
- \* facoltativo.

Nel nostro caso, però, ogni requisito risulta fondamentale per l'effettiva completezza del progetto; quindi, per quanto i requisiti possano avere priorità diverse, ai fini dello sviluppo, essi risultano tutti necessari.

Di conseguenza l'importanza dei requisiti è omessa.

### 3.3.1 Requisiti funzionali

Nella Tabella 3.1 vengono riassunti i requisiti funzionali dell'applicazione.

ID Requisito	Descrizione	Fonti
<b>RF1</b>	Gli utenti non autenticati devono potersi registrare come studenti	UC1
<b>RF2</b>	Gli utenti non autenticati che hanno eseguito la registrazione in precedenza devono poter effettuare il log-in	UC2
<b>RF3</b>	Gli utenti non autenticati devono poter richiedere il cambio della password	UC3
<b>RF4</b>	Gli utenti autenticati devono confermare la propria mail successivamente al log-in se non l'hanno già fatto in precedenza	UC4
<b>RF5</b>	Gli utenti autenticati devono poter visualizzare i dati associati al proprio profilo	UC5
<b>RF6</b>	Gli utenti autenticati devono poter modificare mail nome e password	UC6
<b>RF7</b>	Gli utenti autenticati devono poter effettuare il log-out	UC7
<b>RF8</b>	Gli utenti autenticati devono poter visualizzare la lista di teaching unit ad essi associate	UC8
<b>RF9</b>	Gli insegnanti e gli educatore devono poter visualizzare i dettagli di una specifica teaching unit	UC9
<b>RF10</b>	Gli utenti autenticati devono poter visualizzare i contenuti di specifiche teaching unit	UC10
<b>RF11</b>	Gli studenti e gli educatori devono poter visualizzare il forum studenti	UC11
<b>RF12</b>	Gli educatori devono poter creare nuove teaching unit (in bozza)	UC12
<b>RF13</b>	Gli educatori devono poter eliminare teaching unit non ancora pubblicate	UC13
<b>RF14</b>	Gli educatori devono poter pubblicare teaching unit in bozza	UC14
<b>RF15</b>	Gli educatori devono poter modificare teaching unit non ancora pubblicate	UC15

<b>RF16</b>	Gli educatori e gli insegnanti devono poter visualizzare il forum degli insegnanti	UC16
<b>RF17</b>	Gli educatori e gli studenti devono poter visualizzare i topic del forum studenti	UC17
<b>RF18</b>	Gli educatori e gli insegnanti devono poter visualizzare i topic del forum dei professori	UC18
<b>RF19</b>	Gli educatori devono poter eliminare risposte all'interno di topic sia insegnanti che studenti	UC19
<b>RF20</b>	Gli educatori devono poter creare nuovi questionari (in bozza)	UC20
<b>RF21</b>	Gli educatori devono poter modificare i questionari non ancora pubblicati	UC21
<b>RF22</b>	Gli educatori devono poter pubblicare i questionari in bozza	UC22
<b>RF23</b>	Gli educatori devono poter eliminare i questionari in bozza	UC23
<b>RF24</b>	Gli educatori devono poter visualizzare i dettagli i questionari pubblicati	UC24
<b>RF25</b>	Gli educatori devono poter visualizzare le risposte date ai questionari	UC25
<b>RF26</b>	Gli studenti devono poter creare nuovi topic nel forum studenti	UC26
<b>RF27</b>	Gli studenti devono poter rispondere a topic studenti esistenti	UC27
<b>RF28</b>	Gli insegnanti devono poter creare nuove <i>classi</i>	UC28
<b>RF29</b>	Gli insegnanti devono poter visualizzare i dettagli di una propria <i>classe</i>	UC29
<b>RF30</b>	Gli insegnanti devono poter assegnare teaching unit pubblicate alle proprie <i>classi</i>	UC30
<b>RF31</b>	Gli insegnanti e gli studenti devono poter compilare i rispettivi questionari	UC31

**Tabella 3.1:** Tabella requisiti funzionali

### 3.3.2 Requisiti qualitativi

Nella Tabella 3.2 vengono riassunti i requisiti qualitativi dell'applicazione.

ID Requisito	Descrizione	Fonti
<b>RQ1</b>	Il codice scritto dovrà essere versionabile tramite Git	Azienda
<b>RQ2</b>	Il repository remoto utilizzato dovrà essere su BitBucket	Azienda
<b>RQ3</b>	Per l'implementazione delle funzionalità dovrà essere utilizzato il framework Angular	Azienda
<b>RQ4</b>	Per la creazione di componenti Angular ci si dovrà basare da quelli forniti da Angular Material	Azienda
<b>RQ5</b>	Per le comunicazioni con il back-end si richiede di utilizzare il framework Apollo Angular	Azienda

<b>RQ7</b>	Il back-end dell'applicazione dovrà utilizzare GraphQL	Azienda
------------	--	---------

**Tabella 3.2:** Tabella requisiti qualitativi

### 3.3.3 Requisiti di vincolo

Nella Tabella 3.3 vengono riassunti i requisiti di vincolo dell'applicazione.

<b>ID Requisito</b>	<b>Descrizione</b>	<b>Fonti</b>
<b>RV1</b>	I dati relativi agli utenti dovranno essere memorizzati nel back-end	Azienda
<b>RV2</b>	I dati relativi alle <i>teaching unit</i> dovranno essere memorizzati nel back-end	Azienda
<b>RV3</b>	I dati relativi ai questionari dovranno essere memorizzati nel back-end	Azienda
<b>RV4</b>	I dati relativi ai forum dovranno essere memorizzati nel back-end	Azienda

**Tabella 3.3:** Tabella requisiti di vincolo



## Capitolo 4

# Sviluppo

Questo capitolo tratterà della progettazione e dell'implementazione dell'applicazione. Saranno evidenziati i casi in cui gli standard aziendali non sono stati seguiti, spiegando la situazione e le motivazioni che hanno portato a tale decisione.

### 4.1 Tecnologie utilizzate

Segue la lista di tecnologie utilizzate per lo sviluppo dell'applicazione:

- \* **ECMAScript 6**: è l'insieme delle specifiche standardizzate da ECMA International; È stato creato per standardizzare JavaScript in modo da facilitare più implementazioni indipendenti. JavaScript è rimasto l'implementazione più conosciuta dalla pubblicazione dello standard [8, 9].
- \* **TypeScript**: è un'implementazione di ECMAScript sviluppata da Microsoft; l'obiettivo di TypeScript è trovare gli errori nel codice il prima possibile tramite un sistema tipizzato e di rendere quindi lo sviluppo in JavaScript più efficiente. TypeScript inoltre è un superset di JavaScript, quindi, ogni blocco di codice funzionante in JavaScript funzionerà anche su TypeScript [22, 16].
- \* **HTML5**: è un linguaggio di markup che definisce le proprietà e i contenuti di una pagina web. Le novità introdotte dall'HTML5 sono finalizzate soprattutto a migliorare il disaccoppiamento fra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, ecc.), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio. Inoltre HTML5 prevede il supporto per la memorizzazione locale di grandi quantità di dati scaricati dal web browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta di Google o altri servizi analoghi) anche in assenza di collegamento a Internet [10, 11].
- \* **CSS (Cascading Style Sheets)** è un linguaggio utilizzato per definire lo stile ed il layout di componenti web, separando quindi la struttura ed il loro comportamento dal modo in cui vengono visualizzati [20].
- \* **Sass (Syntactically Awesome StyleSheets)** è un linguaggio che estende CSS sfruttando un preprocessore per aggiungere funzionalità come: l'utilizzo di variabili, la divisione lo stile in più file e la creazione di funzioni [27].

- \* **Web browsers:** Per lo sviluppo è stato utilizzato il browser Mozilla Firefox sviluppato da Mozilla [23]. Per il testing invece, oltre a Firefox, sono stati utilizzati anche Safari, sviluppato da Apple [15], e Chrome, sviluppato da Google [19].
  - \* **WebStorm:** è un IDE (Integrated Development Environment) che è stato utilizzato per lo sviluppo dell'applicazione. È stato scelto WebStorm in quanto è appositamente creato per lo sviluppo di webapp, è altamente modulare grazie al supporto di numerosi plugin esterni e presenta un'interfaccia intuitiva ma potente [29].
  - \* **Git:** è il sistema di versionamento più utilizzato ed è ormai uno standard dell'industria [24]; durante lo sviluppo è stato utilizzato mediante SourceTree.
  - \* **Sourcetree:** è un client Git che permette di utilizzare Git in modo più semplice ed intuitivo grazie ad un'interfaccia grafica, supporta inoltre GitFlow in modo nativo [28].
  - \* **BitBucket:** è un servizio di hosting online per repository che supporta sia Git che Mercurial, un'altro sistema di versionamento. È stato scelto BitBucket al posto di GitHub, quello più utilizzato, perché a differenza di GitHub permette la creazione di repository private gratuitamente [14].
  - \* **Angular:** Angular (da non confondere con AngularJS) è un framework JavaScript che semplifica lo sviluppo di interfacce web tramite una sintassi dichiarativa. I principali vantaggi di angular sono i seguenti:
    - Data-binding bidirezionale: permette di gestire semplicemente la sincronizzazione tra DOM e modello e viceversa;
    - MVVM: implementa nativamente un modello MVVM, questo permette di rispettarlo con molta facilità;
    - Dependency injection: permette di dichiarare quali sono le dipendenze dei diversi componenti dell'applicazione e si occupa in modo automatico di renderle disponibili dove necessario. Questo favorisce la separazione delle funzionalità nei singoli componenti e servizi che se ne devono occupare.
- [1, 21]
- \* **Angular CLI:** è uno strumento a linea di comando utilizzato per inizializzare le applicazioni Angular e per creare moduli, servizi, componenti e direttive [5].
  - \* **Angular Material:** è un toolkit che fornisce un insieme di componenti già pronti per l'utilizzo in un'applicazione che rispetta i principi del material design [4].
  - \* **Angular Cdk:** è un toolkit che fornisce un insieme di direttive e componenti al fine di implementare semplicemente comportamenti comuni, ma non banali, delle applicazioni web o mobile. Un esempio di questi comportamenti è il drag and drop [3].
  - \* **GraphQL:** è un linguaggio di query utilizzato per accedere a dati presenti nel back-end dell'applicazione, i principali vantaggi sono che permette di richiedere informazioni parziali, ottenendo in risposta solo quelle e che permette di richiedere più di una risorsa in un'unica richiesta [25].



- \* **Apollo GraphQL**: è un framework che permette di gestire entrambi i lati (front-end e back-end) di un API GraphQL [18].
- \* **Angular Apollo**: è un client che permette di integrare facilmente il framework Apollo con Angular, sviluppato per l'utilizzo in componenti della UI (User Interface) che hanno la necessità di recuperare dati un API GraphQL [2].

## 4.2 Progettazione

### 4.2.1 L'architettura di angular

I blocchi che compongono un'applicazione angular sono moduli, che forniscono un contesto di compilazione per i componenti.

I componenti sono classi che definiscono viste, che sono un insieme di elementi dello schermo tra cui Angular può scegliere e modificare sulla base della logica e dei dati del programma. Questi componenti utilizzano i servizi, classi che forniscono funzionalità non collegate direttamente alle viste. I servizi possono essere iniettati dentro i componenti che li utilizzano come dipendenze, questo rende il codice modulare, riutilizzabile ed efficiente.

I moduli racchiudono tutto il relativo codice in insiemi funzionali; un'applicazione Angular è definita da un insieme di questi moduli. Un'applicazione angular ha sempre almeno un modulo radice che permette il bootstrap, e tipicamente ha molti più moduli funzionali.

Sia i componenti che i servizi forniscono metadati che indicano ad Angular come utilizzarli; nel caso dei componenti i metadati li associano a dei template, che definiscono delle viste, mentre nel caso dei servizi i metadati forniscono ad Angular le informazioni che ha bisogno di avere per mettere a disposizione i servizi ai componenti tramite dependency injection.

Un template combina HTML standard con le direttive di Angular e del markup di legamento che permettono ad Angular di modificare l'HTML prima di renderizzarlo per visualizzarlo.

I componenti di un'applicazione tipicamente definiscono molte viste, disposte in modo gerarchico. L'aspetto dei componenti viene definito da codice CSS, ogni file è associato ad un componente e Angular fa in modo che il CSS di un componente non influenzi quello di altri componenti.

Angular mette infine a disposizione il servizio Router che permette di definire i percorsi di navigazione attraverso le views [1].

La Figura 4.1 riassume l'architettura di Angular.

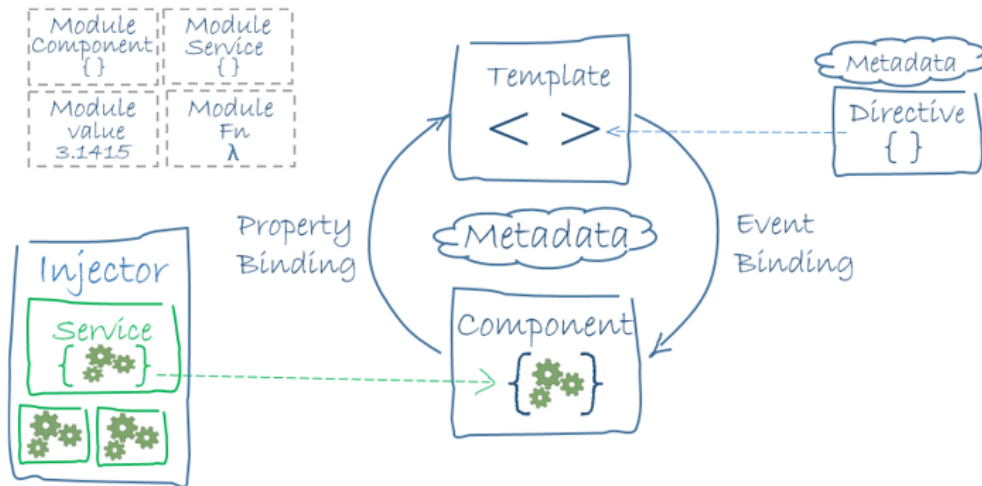
### 4.2.2 Comunicazione con il back-end

Il back-end dell'applicazione fornisce un'API che implementa GraphQL per accedere ai dati.

Accedere ai dati tramite GraphQL è possibile tramite le query e le mutation, le prime quando si vuole leggere i dati o comunque elaborarli senza provocare effetti collaterali, le seconde invece quando si vogliono modificare i dati memorizzati.

Oltre a query e mutation GraphQL permette di definire i fragment, che consistono nella definizione di strutture di dati utilizzate in una o più query/mutation.

Per integrare GraphQL con l'applicazione Angular è stato utilizzato AngularApollo, questo infatti fornisce un servizio che permette di gestire la connessione al back-end,



**Figura 4.1:** L'architettura di Angular [1]

includendo anche la gestione dell'autenticazione.

I componenti Angular possono accedere ai dati del back-end chiamando delle apposite funzioni del servizio a cui vengono passata come parametri le definizioni delle query o delle mutation da eseguire.

### 4.2.3 L'architettura dell'applicazione

Come standard per l'azienda, l'architettura dell'applicazione segue un insieme di regole basilari:

- \* ogni pagina dell'applicazione è definita come un modulo separato, la sua vista viene definita in un unico componente;
- \* se un componente viene utilizzato solamente in una pagina o solamente in singolo altro componente allora sarà definito all'interno del modulo della pagina o del componente che lo contiene;
- \* se ha senso che una stessa istanza di un componente venga utilizzata all'interno di tutta l'applicazione questo verrà definito all'interno di un suo modulo specifico, il quale però, verrà prima importato in un modulo chiamato Core, il cui scopo è appunto raccogliere tutti i componenti di questo tipo. Il modulo Core verrà infine importato nel modulo radice;
- \* i componenti riutilizzati da componenti separati verranno definiti separatamente ai componenti che li contengono;
- \* se non si riesce ad influenzare l'aspetto di un componente tramite il file SCSS associato (sezioni di html caricate dinamicamente o componenti di librerie esterne) viene creato un nuovo file SCSS identificato da ".deep" nel nome;
- \* tutti i file scss ".deep" vengono importati in un file deep.scss apposito che viene importato a sua volta nel file scss di radice dell'applicazione; questo permette al codice dei file ".deep" di modificare altri componenti oltre a quello a loro associati;

- \* query e mutation GraphQL vengono definite assieme ai componenti che le utilizzano;
- \* in modo analogo ai componenti shared anche le query, le mutation e i fragment graphql che sono utilizzati da più componenti vengono definiti separatamente dagli altri.

Nello standard aziendale sarebbe consigliato l'utilizzo di un modulo shared, che definisce tutti i componenti utilizzati in più parti dell'applicazione; questo poi viene importato quando è necessario utilizzare uno dei componenti. Si è deciso insieme all'azienda che, partendo con questo progetto, il modulo shared non sarebbe stato utilizzato, ma semplicemente ogni componente che ne avrebbe fatto parte sarebbe stato definito nel proprio modulo, separatamente dagli altri, in un una cartella shared, e successivamente importato all'interno dei moduli che ne hanno bisogno.

Questo permette di evitare l'import automatico di tutti i componenti shared ogni qualvolta se ne debba utilizzare solo uno.

Un'altra variazione rispetto allo standard aziendale è nell'header; per quanto strutturalmente questo permanga in tutta l'applicazione esso ha dei contenuti che variano sulla base del percorso in cui ci si trova e che possono successivamente essere cambiati dinamicamente dal componente della pagina in cui ci si trova.

Di conseguenza, mentre esso sarebbe un componente "core", esso in questo progetto è situato insieme ai componenti shared in quanto si è deciso di avere un'istanza separata del componente per ogni pagina che ne fa utilizzo; in questo modo è possibile caricare le variazioni direttamente dal componente della pagina senza dover passare per un complesso servizio aggiuntivo.

## 4.3 Implementazione

Descriviamo nel seguito componenti, servizi e moduli Angular, creati seguendo le regole poste dalla progettazione, utilizzati per implementare l'applicazione.

### 4.3.1 Shared

Questi sono gli elementi utilizzati da più componenti, che non rientrano nel modulo Core.

All'interno di shared sono inoltre presenti mutazioni e query GraphQL utilizzate da più elementi e tutti i fragment GraphQL.

#### **BaseComponent**

Questo componente non è mai utilizzato in modo diretto, ma, serve come componente di base per i componenti che devono gestire più subscriptions, in modo tale che vengano memorizzate e chiuse quando il componente viene eliminato.

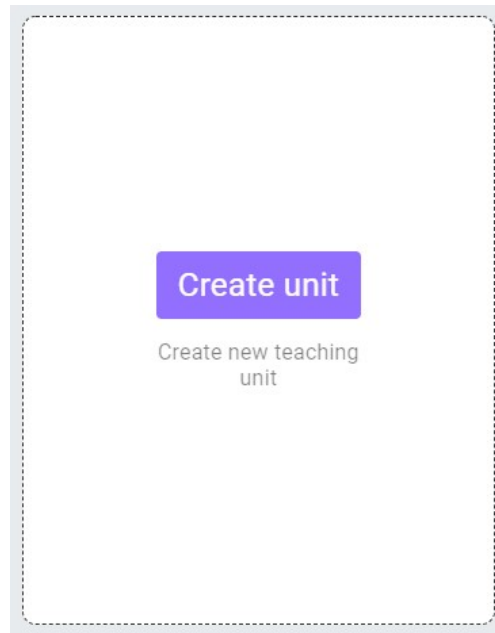
Viene utilizzato tramite ereditarietà.

#### **CardPlaceholderComponent**

Questo componente (visualizzato in Figura 4.2) consiste in una card bianca con il bordo tratteggiato, questa può includere testo o pulsanti.

Viene utilizzata quando è necessario un placeholder tra altre card nel caso del drag

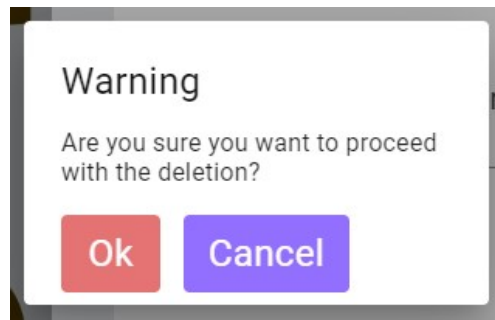
and drop, oppure, quando in una lista di card è necessaria la possibilità di avere azioni di aggiunta.



**Figura 4.2:** CardPlaceholderComponent appearance

### DeleteWarningComponent

Questo componente (visualizzato in Figura 4.3) consiste in una finestra modale utilizzata quando l'utente prova ad eliminare qualcosa, questa modale non fa altro che chiedere conferma per l'eliminazione.



**Figura 4.3:** DeleteWarningComponent appearance

### DragHandleComponent

Questo componente consiste in sei pallini organizzati su tre righe. Viene utilizzata come elemento per ancorare il drag and drop.

**HamburgerButtonComponent**

Questo componente consiste in un bottone composto da tre segmenti paralleli orizzontali, al click di questo bottone i tre segmenti vengono animati in modo che vadano a formare una x.

Questo bottone viene utilizzato per aprire e chiudere il menu laterale da mobile.

**HeaderComponent**

Questo componente consiste nell'header dell'applicazione.

L'header è composto da:

- \* un pulsante per tornare alla pagina precedente: `BackArrowComponent`;
- \* le breadcrumbs (dinamiche) formate da link che permettono di accedere alle pagine precedenti del percorso fatto. Queste sono ricavate dal `BreadcrumbsService`;
- \* del contenuto generico dichiarato dalla pagina che include l'header;
- \* eventuali pulsanti dipendenti dalla pagina che include il componente.

**BackArrowComponent**

Questo componente è dichiarato all'interno di `HeaderModule` in quanto utilizzato solo al suo interno.

Rappresenta un pulsante per tornare alla pagina precedente.

**LoadingComponent**

Questo componente viene utilizzata ogni qual volta dei dati vengono caricati dal back-end, rappresenta uno spinner da visualizzare durante il caricamento.

**PreviewTeachingUnitComponent**

Questo componente (visualizzato in Figura 4.4) consiste nella card di anteprima delle *teaching unit*, con parametri che determinano quali dati visualizzare.

**StatsComponent**

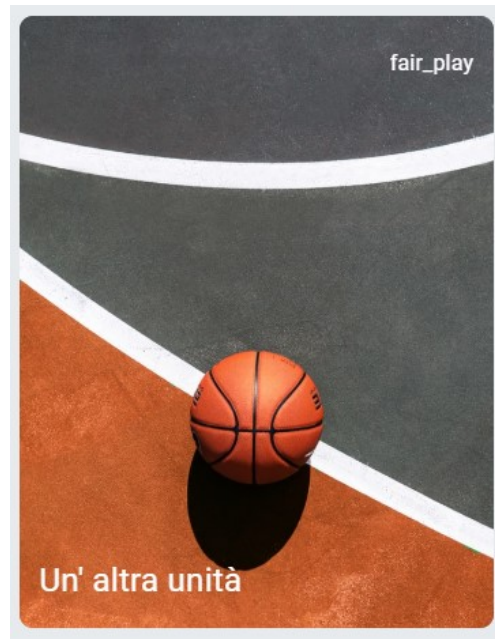
Questo componente è utilizzata per visualizzare un insieme di statistiche passate in input.

Si tratta di un componente shared in quanto questo particolare layout viene utilizzato più volte all'interno dell'applicazione.

**TopicsComponent**

Questo componente è utilizzata per gestire e visualizzare una lista di topic.

Questi possono essere sia relativi al forum studenti, sia al forum insegnanti e quindi relativi ad un unica *teaching unit*.



**Figura 4.4:** PreviewTeachingUnitComponent appearance

### YoutubeIFrameComponent

Questo componente serve per gestire la visualizzazione di video recuperati da Youtube tramite l'API fornita da Youtube.

La creazione del componente è stata necessaria per risolvere un problema di flickering (4.4.2).

### 4.3.2 Core

Gli elementi core sono elementi la cui permanenza deve essere slegata da componenti specifici, o per lo meno legata a AppComponent, la radice dell'applicazione.

Questi elementi includono i servizi, utilizzati per passare informazioni tra componenti non direttamente collegati o per gestire particolari eventi.

### BreadcrumbsService

In un'applicazione web si parla di *breadcrumb* quando ci si riferisce ad una lista di link o testi che rappresentano il percorso che l'utente ha effettuato per raggiungere la pagina in cui si trova attualmente. Il servizio Breadcrumbs intercetta l'evento di termine della navigazione per poi identificare il percorso fatto per raggiungere la pagina corrente e costruire le *breadcrumb* dell'applicazione, per maggiori dettagli il suo utilizzo viene spiegato nella Sezione 4.4.3. Viene utilizzato dall'header.

### GraphQLModule

Questo modulo inizializza e configura il servizio di Apollo-GraphQL in modo da collegarlo, tramite action-cable, al corretto database.

Gestisce inoltre il passaggio dei token per gestire l'autenticazione nell'header delle richieste graphQL.

#### **HamburgerService**

Il servizio Hamburger gestisce il menu in base della dimensione della finestra e al suo ridimensionamento.

Il menu è infatti permanentemente aperto a sinistra se l'ampiezza della finestra supera i 1280px; se invece questo non si verifica il menu viene posizionato a destra e da la possibilità di essere aperto e chiuso.

#### **MetadataService**

Il servizio Metadata serve per gestire i metadati dell'applicazione; esso viene utilizzato principalmente all'avvio, in modo da impostare i metadati dato che questi non subiscono modifiche durante il normale utilizzo dell'applicazione.

#### **ScrollService**

Il servizio Scroll serve per gestire gli eventi relativi allo scroll della pagina tramite gli *Observable* della libreria *rxjs*, questi sono collezioni di dati ottenuti asincronamente, funzionano in modo analogo alle *Promise* [7] definite in ECMAScript, solo che, anziché ritornare un singolo valore e poi essere "consumati" gli *Observable* possono ritornare da 0 a (potenzialmente) infiniti valori [6].

#### **SideNavbarComponent**

Questo componente determina l'aspetto del menu.

Definiti all'interno del modulo sono inoltre *LogoComponent* e *JoinClassDialog*.

#### **LogoComponent**

Questo componente determina l'aspetto del logo del progetto SOS visualizzato nel menu. È stato creato un componente apposito anche se non viene riutilizzato in modo da separare i tag HTML utilizzati per visualizzare in modo reattivo il logo dai tag strutturali degli altri componenti.

#### **JoinClassDialogComponent**

Questo componente viene utilizzato per visualizzare una finestra modale che permette ad uno studente, a cui non è associata nessuna *classe*, di entrare a far parte di una *classe*.

Durante la normale operazione dell'applicazione questa finestra non dovrebbe mai apparire; è stata comunque implementata per prevenire situazioni in cui gli studenti non possono utilizzare l'applicazione in seguito ad interventi anomali.

#### **UserService**

Il servizio User serve per gestire i permessi ed i dati dell'utente autenticato. Lavora a stretto contatto con il modulo dedicato al routing dell'applicazione fornito da Angular per determinare se una pagina può essere visualizzata, o meno, dall'utente che sta utilizzando l'applicazione.

## CoreModule

Questo modulo racchiude tutti gli elementi precedentemente definiti. Ciò permette di racchiuderli tutti in un unico import effettuato all'avvio dell'applicazione.

### 4.3.3 ClassPage

Questa pagina (visualizzata in Figura 4.5) è utilizzata per visualizzare i dettagli di una *classe*; ci può accedere quindi solo il professore associato a tale *classe*.

La pagina è composta da una prima sezione che mostra i dettagli della *classe*, includendo alcune statistiche (tramite StatsComponent) e i nomi degli studenti che ne fanno parte. La seconda sezione invece mostra la preview (tramite PreviewTeachingUnitComponent) delle unità associate alla *classe* e da la possibilità di associarne di nuove (tramite CardPlaceholderComponent e AddUnitDialogComponent).

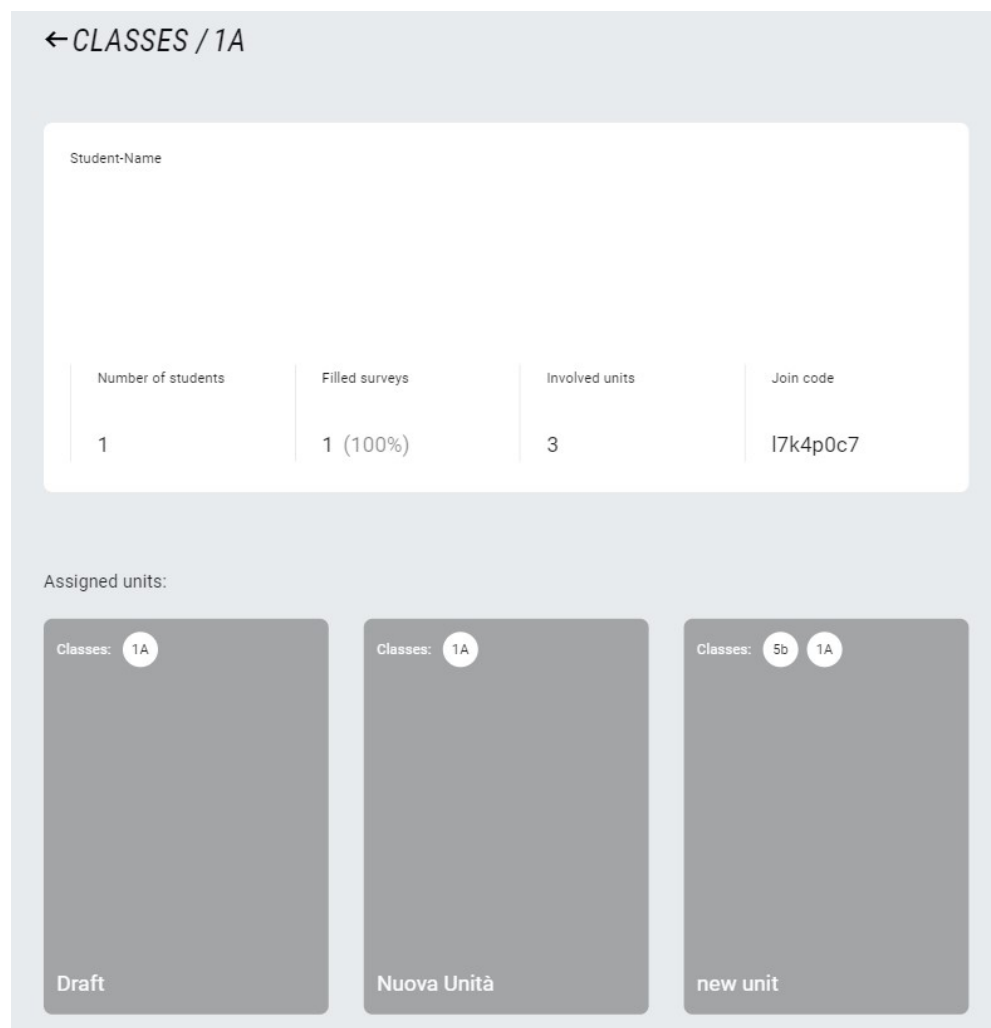


Figura 4.5: ClassPage appearance



### AddUnitDialogComponent

All'interno di ClassPageModule viene definito il componente AddUnitDialog, questo rappresenta una finestra modale che permette di selezionare quale unità associare alla *classe*.

#### 4.3.4 ClassesListPage

Questa pagina (visualizzata in Figura 4.6), alla quale può accedere solo un insegnante, mostra una lista di tutte le *classi* create dall'insegnante che sta visualizzando la pagina. Le *classi* sono rappresentate da una card che mostra alcune informazioni riguardo alla *classe* (anche tramite StatsComponent).

Tramite l'header (HeaderComponent) è possibile creare una nuova *classe*.

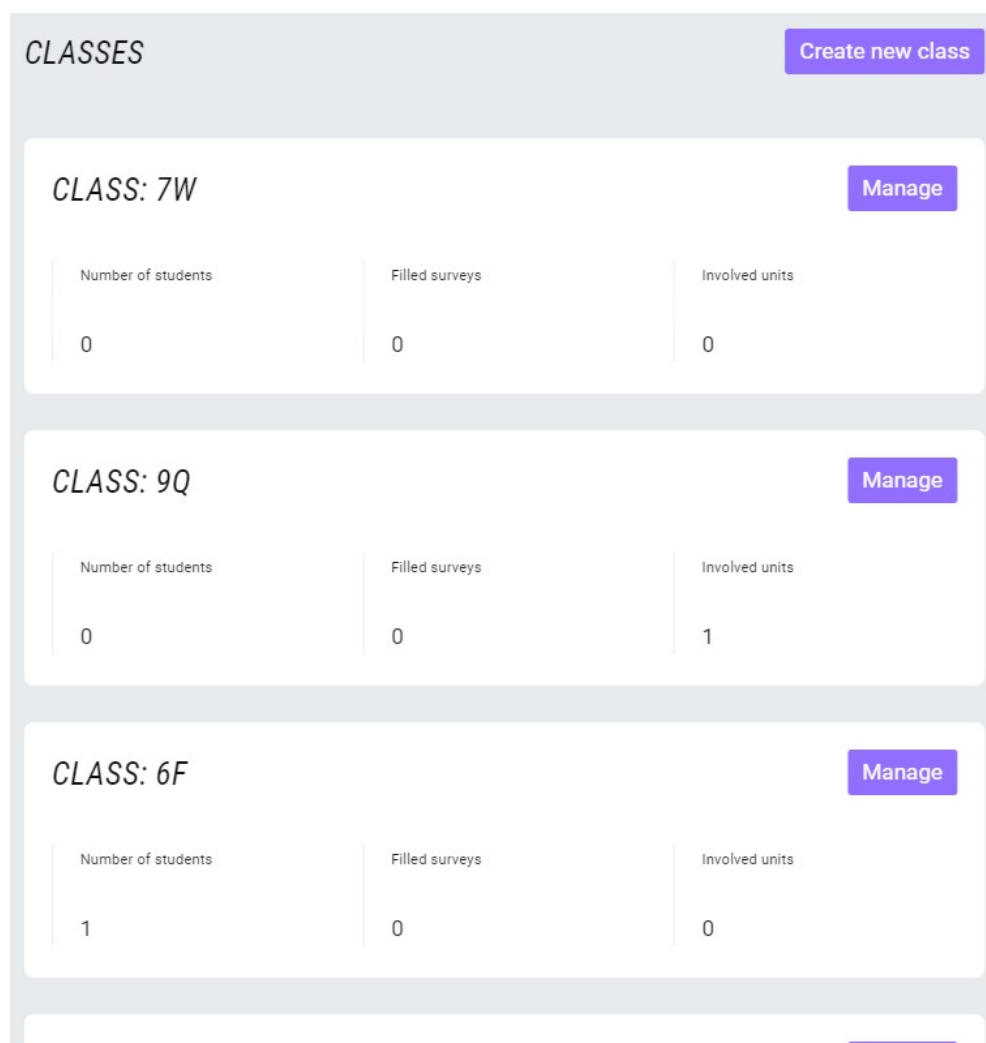


Figura 4.6: ClassesListPage appearance

### CreateClassDialogComponent

Questo componente (visualizzata in Figura 4.7), definito all'interno di `ClassesListPageModule`, rappresenta una finestra modale che permette di creare una nuova *classe* inserendone il nome e visualizzando poi il suo join code.

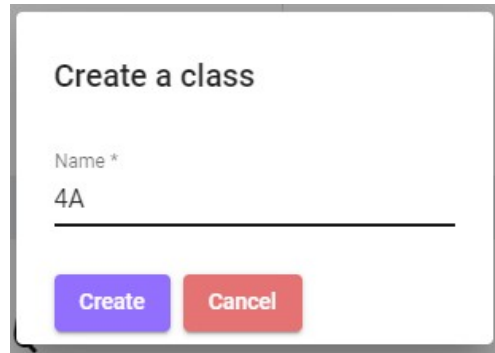


Figura 4.7: CreateClassDialogComponent appearance

### 4.3.5 EditSurvey

Questa pagina permette la modifica di un questionario specifico; da notare la possibilità di espandere e comprimere le domande per ottimizzare lo spazio utilizzato e la possibilità di riordinare le domande tramite le funzionalità di drag and drop fornite da Angular Cdk.

### 4.3.6 EditTeachingUnit

Questa pagina è la più complessa dell'applicazione, questo perché permette di modificare le *teaching unit*, l'elemento alla base dell'intera applicazione e una struttura di dati non banale.

La struttura della pagina è invece relativamente semplice, vi è una lista delle sezioni della pagina, siano esse paragrafi o sequenze (caroselli) e al termine di questa lista c'è un `CardPlaceholderComponent` che permette di aggiungere altre sezioni.

La gestione e la validazione delle sequenze è gestita da un'altro componente (`CarouselEditComponent`) mentre la gestione dei paragrafi e dell'ordinamento (tramite `CdkDragAndDrop`) è invece gestita dalla pagina.

### CarouselEditComponent

Questo componente, definito in `EditTeachingUnitModule`, gestisce la modifica e la validazione di una sequenza di una *teaching unit*.

La sezione principale del componente è dedicata alla gestione delle slide della sequenza, che possono essere di tre tipi: testo, immagine e video.

L'ultima slide presente nella sequenza è una slide vuota fittizia; quando viene modificata, infatti, ne apparirà una nuova in ultima posizione.

Se in una slide non vengono caricati video o immagini, allora questa sarà una slide di testo, altrimenti il testo inserito nella sezione inferiore della slide sarà la didascalia del

video o dell'immagine.

L'inserimento di una nuova immagine viene effettuato tramite un file picker presente nella sezione superiore della slide, l'inserimento invece di un video avviene tramite il menu di gestione della slide.

Lo stesso menu (identificato dal pulsante con i tre pallini) permette la rimozione dell'immagine/video caricati e la rimozione della slide.

I video sono visualizzati tramite `YoutubeIFrameComponent`.

Le slide possono essere riordinate sempre tramite `CdkDragAndDrop`, cliccando e trascinando sul pulsante con le quattro frecce.

L'implementazione del drag and drop in questo elemento è stata abbastanza complessa in quanto, se le slide non possono stare su un'unica riga devono andare a capo; questo non è supportato di default da `CdkDragAndDrop` e ha quindi causato dei problemi (4.4.1).

#### **LoadVideoDialogComponent**

Questo componente è dichiarato all'interno di `EditTeachingUnitModule` e rappresenta una finestra modale finalizzata al caricamento di un video da Youtube.

La finestra è costituita semplicemente da un campo di input dove inserire il link del video; in automatico, prima di caricare il link a back-end, il link viene convertito tramite espressioni regolari in un formato adatto all'utilizzo in un iframe dell'API di Youtube.

#### **4.3.7 ForgotPassword**

Questa pagina ha una duplice funzionalità: inizialmente, quando l'utente ci entra, la pagina contiene solamente un form che chiede all'utente di inserire la propria mail per effettuare il recupero password.

Al submit del form l'utente riceverà una mail contenente l'indirizzo di questa stessa pagina, con un token passato tramite richiesta URL.

Quando infine l'utente torna su questa pagina dalla mail al posto del form di inserimento della mail si presenta il form per effettuare il cambio della password.

#### **4.3.8 Profile**

Questa pagina contiene i form necessari per la modifica dei dati dell'utente. In particolare, il form dedicato al cambio della password è separato rispetto a quello per il cambio degli altri dati, in questo modo l'utente non è forzato a modificare la password ogni volta che vuole modificare i propri dati.

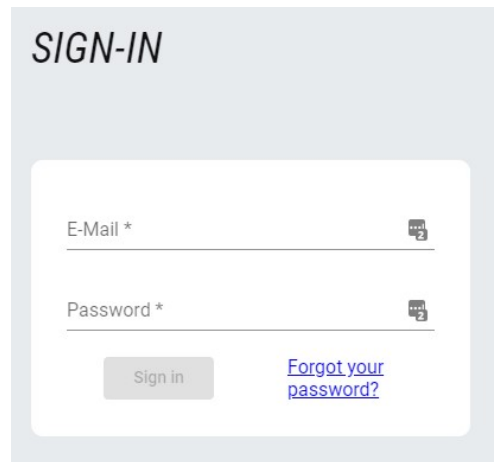
#### **4.3.9 ProfileEmailConfirm**

Questa pagina richiede all'utente autenticato, qual'ora non sia ancora stato fatto, di confermare la propria mail. Per effettuare questa operazione l'utente dovrà accedere all'indirizzo e-mail inserito alla registrazione e seguire le istruzioni che ha ricevuto.

#### **4.3.10 SignIn**

Questa pagina (visualizzata in Figura 4.8) contiene il form per permettere ad un utente di effettuare il login. In particolare, come nel sign-up, viene utilizzata una funzione

che, se vengono ricevuti messaggi di errore dal back-end dovuti all'inserimento erraneo dei dati, li visualizza come messaggi d'errore sotto i singoli campi del form.



**Figura 4.8:** SignIn appearance

#### 4.3.11 SignUp

Questa pagina (visualizzata in Figura 4.9) contiene il form necessario all'iscrizione di uno studente. Dovendo permettere l'inserimento di tipologie di dati diversi, come la data di nascita, sono stati utilizzati gli appositi componenti di Angular Material per effettuare l'input dei dati. Questo ha permesso di evitare l'implementazione di componenti relativamente complessi che non ricadono nello scope dell'applicazione.

#### 4.3.12 StudentForum

Questa pagina rappresenta il punto di accesso ai topic del forum studenti, non essendo questi legati alle *teaching unit*.

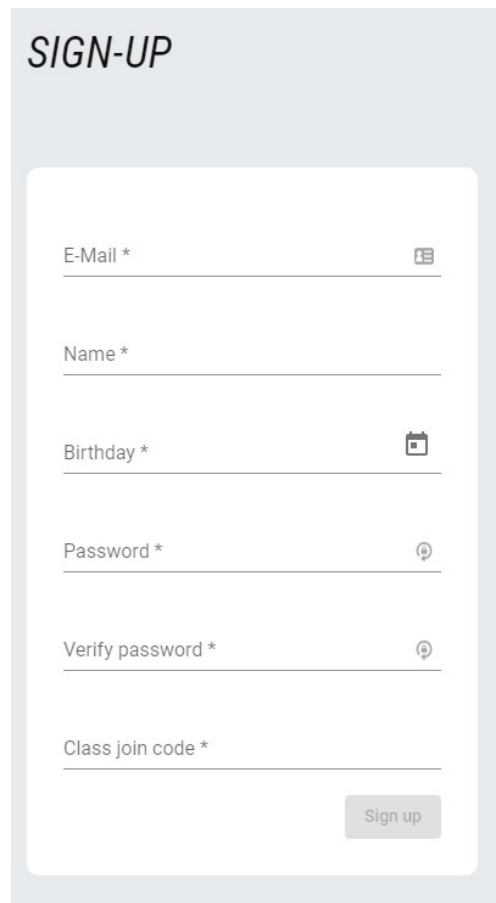
La pagina contiene semplicemente l'header (HeaderComponent) e un'istanza di TopicListComponent contenente tutti i topic studenti.

#### 4.3.13 Survey

Questa pagina permette di compilare un questionario scelto dall'utente tramite la pagina ViewSurveys; per ogni domanda ne viene identificata la tipologia e sulla base di questa viene utilizzato un template diverso per permettere all'utente di inserire un testo, selezionare una risposta o selezionarne più di una.

#### 4.3.14 TeachingUnitPageTeacher

Questa pagina (visualizzata in Figura 4.10) permette ad un insegnante di visualizzare i dettagli di una *teaching unit* già pubblicata, ne mostra la preview tramite PreviewTeachingUnitComponent e ne mostra alcune statistiche tramite StatsComponent.

A mobile app sign-up form titled "SIGN-UP" in a bold, italicized font. The form is contained within a white rounded rectangle on a light gray background. It features six input fields, each with a label and an asterisk indicating it is required. The fields are: "E-Mail" with an envelope icon, "Name", "Birthday" with a calendar icon, "Password" with a key icon, "Verify password" with a key icon, and "Class join code". A "Sign up" button is located at the bottom right of the form.

**Figura 4.9:** SignUp appearance

#### 4.3.15 TeachingUnitPageTrainer

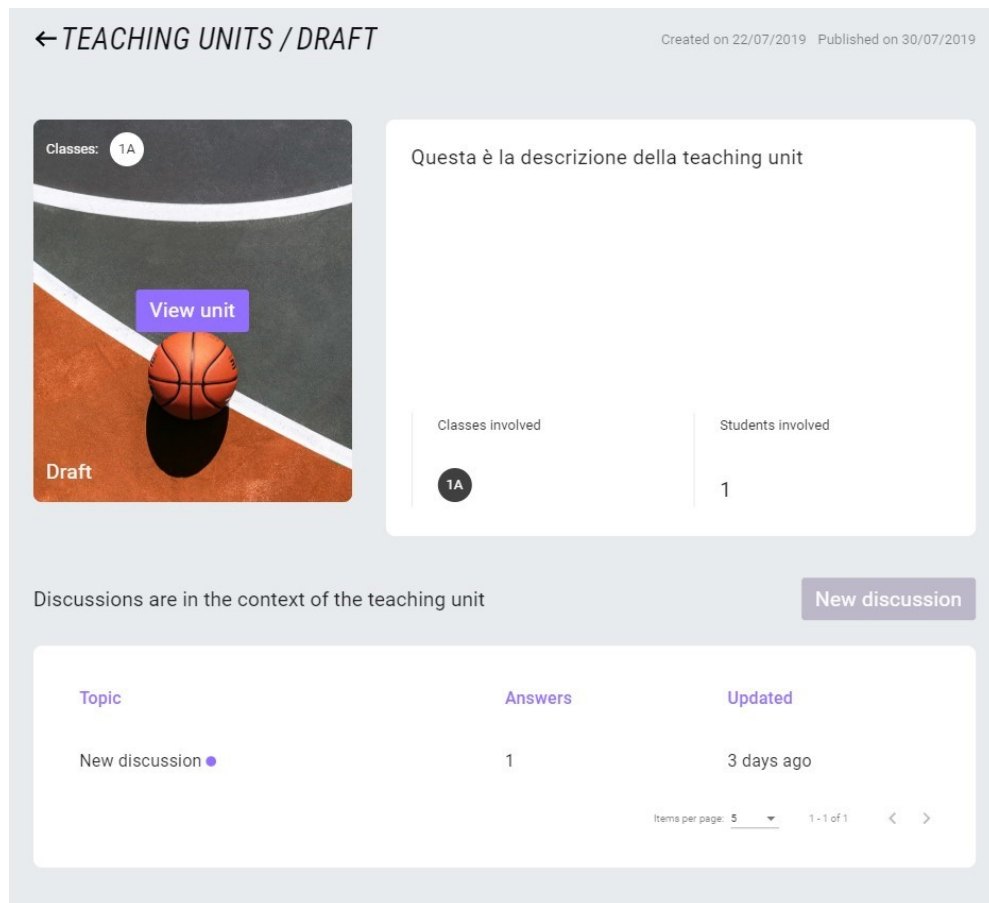
Questa pagina (visualizzata in Figura 4.11) ha una duplice funzione: permette ad un educatore di visualizzare i dettagli di una *teaching unit* già pubblicata in modo analogo a TeachingUnitPageTeacher mentre se la *teaching unit* non è ancora pubblicata, ma è in bozza, permette di modificarne i dettagli del tipo:

- \* immagine di anteprima,
- \* *pilaastro* associato,
- \* descrizione,
- \* titolo.

Nell'header (HeaderComponent) della pagina infine c'è la possibilità di eliminare o pubblicare la *teaching unit*.

#### 4.3.16 TeachingUnitsList

Questa pagina (visualizzata in Figura 4.12) ha la funzionalità di mostrare una lista di *teaching unit* per permettere all'utente di selezionarne una; se l'utente è un educatore



**Figura 4.10:** TeachingUnitPageTeacher appearance

ne permetta anche la creazione.

La pagina è composta da una lista di `PreviewTeachingUnitComponent` che mostrano l'anteprima e qualche dettaglio delle *teaching unit*, se l'utente è un educatore a questa lista viene aggiunto un `CardPlaceholderComponent` per permettere la creazione di nuove *teaching unit*.

### CreateTeachingUnitComponent

Questo componente è dichiarato all'interno di `TeachingUnitListModule` e rappresenta una finestra modale il cui scopo è la creazione di una nuova *teaching unit*.

A questo fine, in questa finestra, è possibile inserire il titolo della nuova *teaching unit* e selezionare a quale *pilaastro* appartiene.

#### 4.3.17 Topic

Questa pagina ha lo scopo di visualizzare o creare un topic studente o insegnante.

Nell'*url* della pagina viene specificato l'*url* del topic da visualizzare, se questo non è presente la pagina va in modalità creazione; la differenza sta nel fatto che in modalità

The screenshot shows a web interface for managing teaching units. At the top, there's a header with a back arrow, the text 'TEACHING UNITS / VIEW', and buttons for 'Draft', 'Delete', 'Save', and 'Publish'. Below the header, the 'Title \*' field is set to 'View'. The main content area is divided into two sections. On the left, there's a large dashed box with the text 'Click to upload a preview image'. On the right, there's a form titled 'Exercise sheet'. This form has a 'Type \*' dropdown menu currently showing 'Fairplay', and a 'Description' text area containing the text 'Teaching unit di prova per testare il funzionamento della visualizzazione delle teaching unit'. At the bottom of the right section, there are two buttons: 'Fill in' and 'Preview', followed by a small instruction: 'Complete the exercise sheet to make it available to teachers.'

**Figura 4.11:** TeachingUnitPageTrainer appearance

creazione anziché visualizzare titolo e contenuto del topic come plain-text vengono visualizzati due campi di input. Se la pagina sta visualizzando un topic esistente da anche la possibilità di rispondere al topic.

#### 4.3.18 ViewFilledSurvey

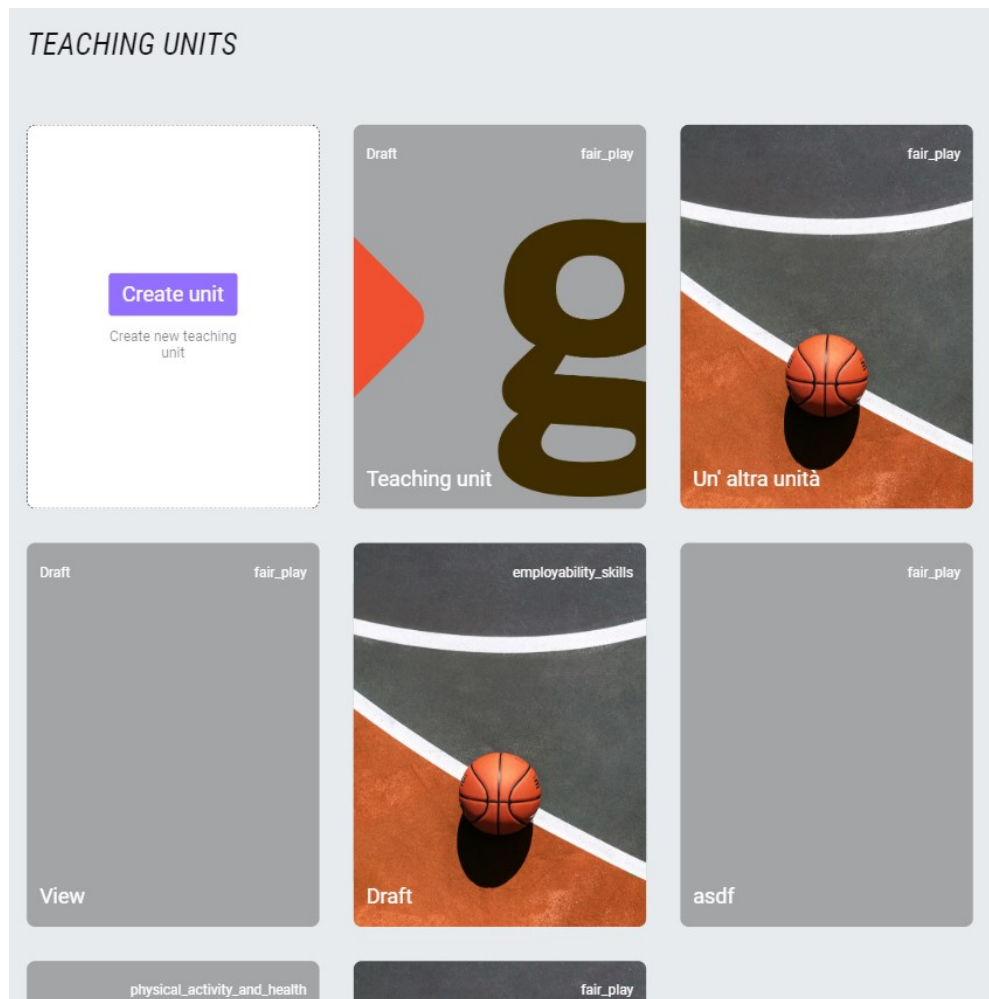
Questa pagina (visualizzata in Figura 4.13) è analoga a Survey la differenza principale sta nel fatto che i campi di input di questa pagina sono disabilitati e precompilati.

#### 4.3.19 ViewSurveys

Questa pagina ha lo scopo di visualizzare uno o più questionari con qualche dettaglio, permettendone la selezione. Nel caso l'utente sia un educatore esso visualizzerà tutti i questionari, siano essi pubblicati o in bozza; in questo modo l'utente potrà consultarne le statistiche, modificarli o crearne di nuovi. In caso contrario invece l'utente vedrà solamente i questionari pubblicati associati alla propria categoria.

#### 4.3.20 ViewUnit

Questa pagina permette di visualizzare i contenuti di una *teaching unit*. Sia i paragrafi che le sequenze vengono visualizzate in card apposite; i paragrafi sono semplicemente costituiti da del testo, mentre le sequenze sono formate da un carosello le cui diverse sezioni sono un'immagine con sotto la didascalia associata, un video (tramite YoutubeIFrameComponent) sempre con la didascalia associata o del testo



**Figura 4.12:** TeachingUnitsList appearance

analogo a quello dei paragrafi, questa sezione presenta inoltre i controlli per avanzare alla slide successiva o ritornare alla slide precedente.

## 4.4 Problemi di implementazione riscontrati

### 4.4.1 Drag and Drop multidirezionale

L'implementazione del drag and drop per l'ordinamento delle slide all'interno delle sequenze delle *teaching unit* si è rivelato più problematico del previsto; questo perché il drag and drop implementato in AngularCdk non supporta di default la possibilità di effettuare il riordinamento su più direzioni, ma solo orizzontalmente o verticalmente. Per riuscire ad implementare lo stesso la funzionalità è stato necessario tracciare quando il componente che viene trascinato passa sopra al contenitore di un altro elemento. Quando questo avviene, il contenitore viene considerato attivo e aggiorna la visualizzazione degli elementi in modo da rispecchiare il cambio di ordine.



The screenshot shows a web interface for a survey. At the top, there is a navigation bar with a back arrow and the text "← SURVEYS / NEW SURVEY". Below this, the survey is divided into four sections, each with a title and a list of answers:

- Prima domanda (chiusa)**: A closed question with three radio button options: "Ans 1" (selected), "Ans 2", and "Ans 3".
- Terza domanda (risposta multipla)**: A multiple-choice question with four checkbox options: "Ans 1", "Ans 2" (checked), "Ans 3" (checked), and "Ans 4".
- Domanda random**: A question with a single text input field containing the letter "a".
- Seconda domanda (aperta)**: An open question with a single text input field containing the letter "a".

**Figura 4.13:** ViewFilledSurvey appearance

Quando il componente trascinato viene rilasciato, se è presente un container attivo, allora propaga il cambiamento dell'ordine degli elementi anche nella struttura dati del model, che tramite il doppio binding di angular, viene rispecchiato nella view.

#### 4.4.2 Youtube video flickering

Inizialmente gli iframe dei video caricati da Youtube erano inseriti direttamente all'interno del componente che li doveva contenere; questo però causava problemi ogni qual'volta il video doveva essere mosso "artificialmente" ad esempio dal drag and drop o dall'avanzamento della slide in cui si trova.

Il video infatti cambiava velocemente tra una schermata nera e la thumbnail del video, per poi sistemarsi quando il movimento si ferma.

La soluzione è stata quella di visualizzare effettivamente l'iframe di Youtube solamente quando deve essere riprodotto il video, quando invece l'utente non ha ancora interagito con il video, o il video viene spostato, allora viene visualizzata solamente la thumbnail del video con il pulsante play di Youtube.

#### 4.4.3 Breadcrumbs tracking

L'implementazione delle breadcrumbs dell'applicazione non è stato banale come si pensava all'inizio, questo perchè le breadcrumbs non sono identificate solamente dal percorso che ha fatto l'utente, ma possono anche dipendere da dati recuperati dal back-end (ad esempio, nella pagina relativa ad una *teaching unit*, l'ultima breadcrumb

deve consistere nel titolo della *teaching unit*); le breadcrumb così generate, poi, devono permanere anche se si cambia pagina, rimanendo visualizzate nel caso in cui si vada in percorsi figli di quello con la breadcrumb dinamica.

All'inizio si voleva utilizzare una libreria esterna per gestire la situazione, purtroppo però, quelle trovate o non fornivano tutte le funzionalità di cui necessitavamo o non funzionavano correttamente.

Il servizio creato per risolvere questo problema (BreadcrumbsService) intercetta l'evento di termine della navigazione in una nuova pagina, calcola ricorsivamente le breadcrumb relative ad ogni elemento del percorso effettuato, dopo di chè, controlla per ciascuna di queste se sono state passate altre breadcrumb come parametro url e, se sono presenti, le visualizza al posto di quelle relative al percorso (questo per la permanenza delle breadcrumb dinamiche); infine fornisce la possibilità di sostituire l'ultima breadcrumb dinamicamente.

## Capitolo 5

# Conclusioni

In questo capitolo verranno valutati gli obiettivi raggiunti e le competenze acquisite; vi sarà infine una valutazione personale dello stage svolto.

### 5.1 Resoconto degli obiettivi raggiunti

La Tabella 5.1 riassume gli obiettivi dello stage e ne presenta lo stato a stage terminato.

ID	IMPORTANZA	DESCRIZIONE	STATO
O01	Obbligatorio	Studio e formazione sulle tecnologie utilizzate (AngularJS, GraphQL, Apollo framework etc...)	Completato
O02	Obbligatorio	Redazione dell'analisi dei requisiti.	Completato
O03	Obbligatorio	Progettazione dell'applicazione frontend secondo i requisiti trovati durante la fase di analisi;	Completato
O04	Obbligatorio	Implementazione dell'applicazione frontend;	Completato
O05	Obbligatorio	Test del funzionamento dell'applicazione realizzata	Completato
D01	Desiderabile	Aggiunta di funzionalità di comunicazione in tempo reale;	Non completato
D02	Desiderabile	Creazione suite di testing frontend;	Non completato
D03	Desiderabile	Redazione documentazione completa sull'applicazione.	Non completato
F01	Facoltativo	Realizzazione di ulteriori funzionalità che potrebbero emergere nella fase di analisi.	Completato

**Tabella 5.1:** Tabella degli obiettivi raggiunti a fine stage

Tutti i requisiti dell'applicazione, al termine del periodo di stage, sono stati rispettati, lo stesso, come si può vedere in Tabella 5.1, vale per gli obiettivi obbligatori e facoltativi. Gli obiettivi desiderabili invece non sono stati soddisfatti; si è valutato, infatti, in collaborazione con il tutor interno all'azienda, che il valore apportato dal completamento degli obiettivi desiderabili e facoltativi sarebbe stato minore rispetto a quello portato

dal miglioramento della qualità del codice, al fine di poter aumentare la manutenibilità dell'applicazione, e alla pacchettizzazione di alcune soluzioni utilizzate in modo che possano successivamente essere riutilizzate in altri progetti aziendali.

Una di queste soluzioni è, per esempio, il meccanismo utilizzato per risolvere il problema del Drag and Drop multidirezionale; tale problema era già stato riscontrato infatti in un'altro progetto dell'azienda, quindi non solo è stato possibile riutilizzare la soluzione trovata per risolvere questo problema, ma sarà possibile riutilizzarla ogni qual volta questo problema verrà riscontrato in futuro (almeno finché AngularCdk non offrirà il supporto nativo per questa funzionalità). Segue la Tabella 5.2 che mostra i tempi pianificati ed effettivi delle diverse attività dello stage.

Attività	Durata pianificata	Durata effettiva
Formazione e apprendimento delle abilità necessarie allo svolgimento del progetto	20 ore	24 ore
Comprensione del sistema e degli obiettivi	20 ore	16 ore
Analisi dei requisiti	32 ore	32 ore
Progettazione	12 ore	16 ore
Studio e setup ambiente di sviluppo	20 ore	16 ore
Termine progettazione e analisi dei requisiti	16 ore	16 ore
Implementazione	150 ore	136 ore
Test e validazione	30 ore	32 ore
Documentazione	20 ore	24 ore
<b>Totale</b>	<b>320 ore</b>	<b>312 ore</b>

**Tabella 5.2:** Tabella delle variazioni della pianificazioni

## 5.2 Valutazione dell'esperienza svolta

Questa esperienza di stage è stata fondamentale per la mia formazione, sia perché mi ha permesso di mettere alla prova ciò che ho appreso nel corso degli studi, sia perché mi ha permesso di provare ciò che vuol dire entrare nel mondo del lavoro.

L'azienda mi ha lasciato una buona quantità di autonomia, questo mi ha permesso di sperimentare le conseguenze di tutte le scelte fatte durante l'implementazione.

L'autonomia lasciatami, inoltre, mi ha fatto capire l'importanza di mantenere un sistema di tracciamento dei requisiti ben aggiornato, in modo da poter sempre aver un'idea dei progressi svolti.

Le tecnologie utilizzate si sono rivelate molto interessanti e, per quanto non mi possa definire un esperto nell'ambito, sono felice del livello di familiarità che ho acquisito con queste..

Infine sono rimasto stupito dall'ambiente di lavoro che ho trovato una volta iniziato lo stage, anche solo questo avrebbe reso l'esperienza di stage positiva.





# Riferimenti bibliografici e sitografici

## Riferimenti bibliografici

- [1] *Architettura generale di angular*. URL: <https://angular.io/guide/architecture>.
- [2] *Documentazione di Angular Apollo*. URL: <https://www.apollographql.com/docs/angular/>.
- [3] *Documentazione di Angular Cdk*. URL: <https://material.angular.io/cdk/categories>.
- [4] *Documentazione di Angular Material*. URL: <https://material.angular.io/components/categories>.
- [5] *Documentazione ufficiale Angular CLI*. URL: <https://angular.io/cli>.
- [6] *Documentazione ufficiale della libreria Rxjs sugli Observable*. URL: <https://rxjs-dev.firebaseapp.com/guide/observable>.
- [7] *Documentazione ufficiale sulle Promise ECMAScript*. URL: [https://developer.mozilla.org/it/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/it/docs/Web/JavaScript/Reference/Global_Objects/Promise).
- [8] *ECMAScript*. URL: <https://en.wikipedia.org/wiki/ECMAScript>.
- [9] *ECMAScript: uno standard Javascript*. URL: <https://auth0.com/blog/a-brief-history-of-javascript/#ECMAScript--JavaScript-as-a-standard>.
- [10] *HTML5*. URL: <https://en.wikipedia.org/wiki/HTML5>.
- [11] *Introduction to HTML5*. URL: [https://www.w3schools.com/html/html5\\_intro.asp](https://www.w3schools.com/html/html5_intro.asp).
- [12] *Introduzione a Scrum*. URL: [https://it.wikipedia.org/wiki/%20Scrum\\_\(informatica\)](https://it.wikipedia.org/wiki/%20Scrum_(informatica)).
- [13] *Manifesto per lo Sviluppo Agile del Software*. URL: <http://agilemanifesto.org/iso/it/manifesto.html>.
- [14] *Pagina informativa su BitBucket ufficiale*. URL: <https://www.atlassian.com/it/software/bitbucket>.

- [15] *Pagina informativa su Safari ufficiale.* URL: <https://www.apple.com/it/safari/>.
- [16] *Pagina wikipedia di TypeScript.* URL: <https://en.wikipedia.org/wiki/TypeScript>.
- [17] *Presentazione del progetto SOS (Sports Open Schools).* URL: <https://www.padovasport.tv/altri-sport/altri/cus-padova-un-nuovo-modo-di-fare-sport-con-il-progetto-sos/>.
- [18] *Sito ufficiale Apollo GraphQL.* URL: <https://www.apollographql.com/>.
- [19] *Sito ufficiale Chrome.* URL: [https://www.google.com/intl/it\\_it/chrome/](https://www.google.com/intl/it_it/chrome/).
- [20] *Sito ufficiale CSS.* URL: <https://www.w3.org/Style/CSS/>.
- [21] *Sito ufficiale di Angular.* URL: <https://angular.io>.
- [22] *Sito ufficiale di TypeScript.* URL: <http://www.typescriptlang.org/index.html>.
- [23] *Sito ufficiale Firefox.* URL: <https://www.mozilla.org/it/firefox/>.
- [24] *Sito ufficiale Git.* URL: <https://git-scm.com/>.
- [25] *Sito ufficiale GraphQL.* URL: <https://graphql.org/>.
- [26] *Sito ufficiale Moku S.r.l.* URL: <http://moku.io>.
- [27] *Sito ufficiale Sass.* URL: <https://sass-lang.com/>.
- [28] *Sito ufficiale Sourcetree.* URL: <https://www.sourcetreeapp.com/>.
- [29] *Sito ufficiale WebStorm.* URL: <https://www.jetbrains.com/webstorm/>.