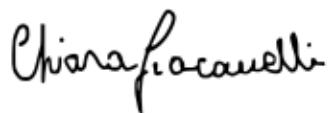


**“Latino Facile: supporti per DSA”:
editor e nuovi tipi di esercizi**

**Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di laurea in Informatica**

Candidato
Chiara Giacanelli
Matricola 1801145

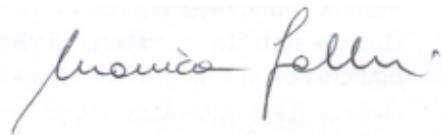


Responsabile
Prof. Andrea Sterbini



A/A 2020/2021

Corresponsabile
Prof.ssa Monica Fabbri



"Latino Facile: supporti per DSA": editor e nuovi tipi di esercizi
Tesi di Laurea. Sapienza – Università di Roma

© 2021 Chiara Giacanelli. Tutti i diritti riservati

Questa tesi è stata composta con L^AT_EX e la classe Sapthesis.

Email dell'autore: giacanelli.1801145@studenti.uniroma1.it

Sommario

Il seguente elaborato di tesi tratta dell'implementazione dell'editor di esercizi per l'applicazione web "Latino facile: supporti per DSA", in risposta alla necessità di rendere il sistema facilmente estendibile da nuove e diverse tipologie di esercizi, secondo una strategia didattica lineare.

Il lavoro è stato suddiviso in tre fasi: la modularizzazione del codice della versione precedente, la costruzione dell'editor e l'ampliamento del sistema con una nuova tipologia più complessa per testare la correttezza della ristrutturazione. Tutto ciò è stato realizzato su due livelli:

- Su un livello tecnico, ragionando sulla tripla *struttura dati-editor-visualizzatore*.
- Su un livello concettuale, attraverso una continua sensibilizzazione nei confronti del caso alla base di questo progetto: i disturbi specifici dell'apprendimento nella scuola.

Indice

Introduzione	1
1 Gli obiettivi della tesi	3
1.1 Punto di partenza: la versione precedente	3
1.2 Prerequisiti e argomenti trattati	4
1.3 Obiettivi didattici	5
1.4 Struttura dati	6
2 Tecnologie utilizzate	7
2.1 React	7
2.2 React-Redux	9
2.3 Il framework Express	10
2.4 Librerie grafiche	12
3 Modularizzazione	13
3.1 Dizionario degli esercizi	13
3.2 Importanza degli Slice	15
3.3 Components dell'Esercizio	17
3.4 Cum narrativo	19
3.4.1 Interfaccia utente	20
3.4.2 Nuovi campi nella struttura dati	33
3.5 Diagramma di flusso dell'API per la lettura dei file	35
4 Editor di esercizi	36
4.1 Components	36
4.2 Fase finale	60
4.3 Strutture dati	62
4.4 Diagramma di flusso dell'API finale	65

5 Test di usabilità e sperimentazione	66
5.1 Testing del cum narrativo	66
5.2 Testing dell'Editor	68
5.3 Considerazioni finali	70
Conclusioni	72
Bibliografia	73
Ringraziamenti	74

Introduzione

"Latino facile: supporti per DSA" è un'applicazione web-based che ha l'obiettivo di dare supporto allo studio del Latino per ragazzi liceali con disturbi specifici dell'apprendimento, in particolare con dislessia.

Il termine "disturbi specifici dell'apprendimento" (DSA) è un termine di carattere generale che si riferisce ad un gruppo eterogeneo di disordini che si manifestano con la presenza di significative difficoltà dell'acquisizione e uso di abilità di comprensione del linguaggio orale, espressione linguistica, lettura, scrittura, ragionamento o matematica [1]. Un DSA rispecchia uno sviluppo neurologico atipico (non è quindi una malattia o una disabilità), denotando necessità diverse a livello scolastico rispetto a quelle più "comuni". Per capire bene questo concetto, aiuta pensare ad un esempio forse troppo semplicistico ma d'impatto: una persona miope senza occhiali non riuscirebbe a vedere bene ciò che la circonda, a delineare i contorni e a vivere come tutti gli altri, ma tutto questo non vuol dire che, con gli occhiali, non abbia le stesse potenzialità di una persona normotipo. Provare in continuazione a resistere senza uno strumento di supporto risulterebbe una perdita di forze e di tempo per lei. La stessa cosa vale per il sostegno a ragazzi e adulti con dislessia: con la Legge 170/10 "Nuove norme in materia di disturbi specifici di apprendimento in ambito scolastico" viene indicato che "gli studenti con diagnosi di DSA hanno diritto a fruire di appositi provvedimenti dispensativi e compensativi [...] nell'ambito di una didattica individualizzata e personalizzata, che tenga conto delle caratteristiche e dei punti di forza dello studente" [2]. L'applicazione, quindi, è stata sviluppata e pensata con l'intento di essere uno **strumento compensativo** in grado di alleggerire alcune delle difficoltà che un ragazzo con questi disturbi incontra durante la sua vita a scuola, nell'ambito di una didattica inclusiva che tiene conto del bisogno educativo del singolo e allo stesso tempo cerca di non fare differenza tra "ragazzi con dislessia" e "ragazzi senza dislessia", poiché l'approccio dell'applicazione per entrambi i gruppi è analogo.

"Latino facile: supporti per DSA" nasce anche con l'idea di aiutare i docenti a co-

struire gli esercizi nel modo più opportuno affinchè vengano coniugate l’acquisizione di nuove nozioni e la minimizzazione di difficoltà nell’apprendimento per gli studenti. Quest’ultimo, in particolare, è stato oggetto del mio tirocinio ed importante valenza ha avuto la collaborazione con una rappresentante del corpo docenti impegnata in prima linea nell’insegnamento della materia, per capire al meglio sia come rispondere alle necessità che un insegnante ha nei confronti dell’insegnamento della materia, ma anche e soprattutto come maneggiare un argomento così delicato come la vita scolastica dei ragazzi con dislessia.

Nel primo capitolo della tesi vengono illustrati gli obiettivi del mio lavoro, con particolare riferimento alle versioni precedenti come suo punto di partenza.

Nel secondo capitolo, sono state descritte le tecnologie utilizzate per lo sviluppo dell’interfaccia utente e per la costruzione delle REST API rese necessarie allo scambio e l’aggiornamento di informazioni nell’applicazione (quali ad esempio i file json per la struttura dati della frase).

Nel terzo capitolo viene illustrata la prima fase del lavoro di tirocinio, ovvero la modularizzazione del codice per il player della versione precedente, avendo come obiettivo quello di permettere l’inserimento di qualsiasi tipo di subordinata all’interno dell’Eserciziario. Questa fase è stata testata e arricchita dall’inserimento di una struttura latina di subordinate più complessa, il *cum narrativo*, inserimento che ha permesso l’ampliamento dell’Eserciziario con nuove regole e visualizzazioni.

Nel quarto capitolo ho descritto la seconda e centrale fase del lavoro di tesi: la realizzazione dell’editor di esercizi per l’applicazione. In particolare, vengono descritte le componenti che lo strutturano fase per fase, simulando la costruzione di un esercizio. Nel capitolo che lo segue, viene illustrata la fase di testing eseguita, sia su studenti DSA che con normodotati e (per l’Editor) con docenti di latino, e i risultati ottenuti.

Capitolo 1

Gli obiettivi della tesi

1.1 Punto di partenza: la versione precedente

La tesi nasce con il primo obiettivo di rendere l'architettura creata precedentemente facilmente estendibile per permettere l'aggiunta di nuove tipologie di esercizi. La versione precedente, realizzata da Simona Lazzarini per la sua tesi, prevedeva la visualizzazione di tre tipologie di subordinate distinte: l'infinitiva, la completiva e l'infinitiva combinata alla finale, ognuna delle quali con una struttura di frontend specifica e differenziata dalle altre. Questo è risultato limitante nel momento in cui si è pensato di andare a costruire un Eserciziario in grado di far visualizzare al sistema qualsiasi tipo di proposizione. Infatti, su un piano didattico lo scopo è stato quello di fornire una continuità in relazione agli argomenti presentati dall'applicazione, cercando quindi di aggiungere nuove categorie di argomenti, sempre sullo scheletro dell'architettura già presente.

Lo scopo della web-app è quindi diventato duplice: è rimasto centrale il pensiero dedicato ai ragazzi ai quali è rivolta, e quindi anche l'obiettivo di rendere le task facilmente eseguibili e comprensibili, con particolari accorgimenti nei confronti della disambiguazione (ruolo fondamentale hanno avuto gli *helper*, strutture implementate per la visualizzazione di aiuti personalizzati rispetto al tipo di esercizio e regole che il docente vuole che vengano assimilate); un secondo obiettivo che ci si è posti è stato quello di creare un Editor in grado di guidare passo passo il docente nella costruzione di una nuova tipologia di esercizio, presentando anche la possibilità di comprendere maggiormente le motivazioni dietro a determinate scelte di visualizzazione (un esempio è il link al Panel delle FAQ che spiega l'importanza dell'utilizzo dei colori nella differenziazione delle tipologie delle subordinate). Difatti, non tutti i docenti che si interfacceranno a questa applicazione potrebbero aver avuto una sensibilizzazione

nei confronti dei disturbi specifici dell'apprendimento, e si è ritenuto fondamentale far capire che ogni dettaglio è stato il risultato di un ragionamento significativo.

1.2 Prerequisiti e argomenti trattati

L'applicazione si rivolge a due target di persone: in primo luogo ai ragazzi frequentanti il primo anno del liceo classico; in secondo luogo, ai docenti che si impegneranno ad utilizzare l'Editor realizzato per ampliare le tipologie di subordinate da proporre, oppure all'editor di una casa editrice che vuole realizzare un manuale di latino con supporti per DSA.

All'avvio della homepage, è stata realizzata una vista modale che chiederà all'utente di identificarsi come studente o come insegnante: questo è risultato necessario per differenziare i servizi che vengono offerti (Figura 1.1). Infatti, per uno studente, non sarà prevista la sezione dell'Editor, in quanto esso sarà accessibile soltanto dai docenti. Tutto ciò è stato realizzato memorizzando in *LocalStorage* le informazioni, con l'intenzione di affidare ad una tesi futura la gestione delle utenze e quindi il relativo servizio di autenticazione con un server di backend.

Ovviamente il prerequisito primario sia per gli insegnanti sia per gli studenti è una conoscenza base della struttura delle piattaforme web-based e dell'user interface design, cosa che, in questi ultimi anni, sta diventando sempre più frequente negli ambiti di lavoro e vita quotidiana; perciò si è pensato che la conoscenza base maturata recentemente nella maggior parte dei futuri utenti possa essere sufficiente all'utilizzo della piattaforma.

Con l'introduzione del *cum narrativo* gli elementi trattati sono stati ampliati. I nuovi argomenti presenti nell'Eserciziario riguardano:

- Struttura del *cum narrativo*
- Traduzione del *cum narrativo* con relativa disambiguazione a livello implicito e esplicito
- Concetti di anteriorità e contemporaneità (e quindi distinzione tra tempi principali e tempi storici)
- Descrizione semantica della frase

Oltre a ciò, l'Editor è stato realizzato tenendo conto di tutte le proprietà che caratterizzano una frase latina, ovvero il numero di subordinate presenti, la tipologia delle subordinate, le coniugazioni e i tempi verbali, la declinazione e la funzione

delle parole, la traduzione (semplice o multipla, a seconda se la frase risulti ambigua come in alcuni casi del cum narrativo).

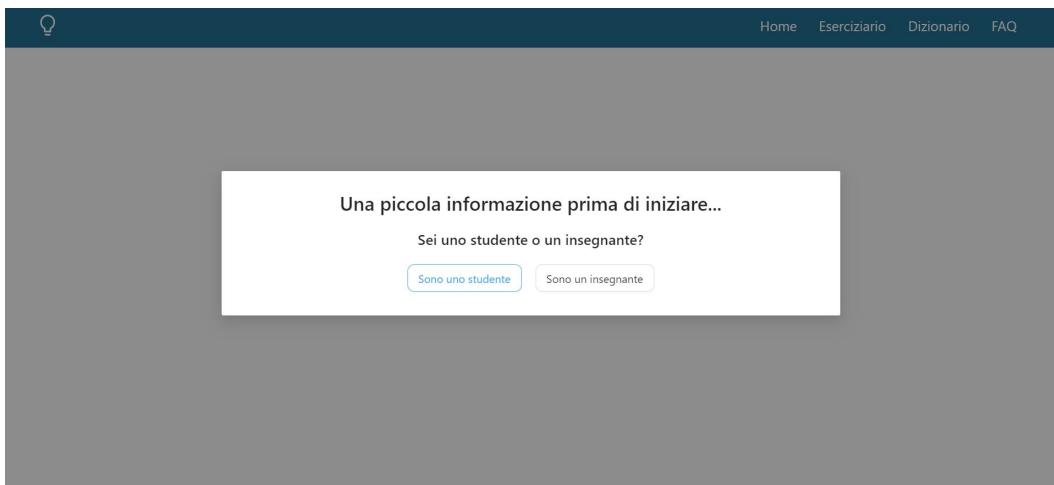


Figura 1.1. Modal iniziale per la differenziazione degli utenti

1.3 Obiettivi didattici

Nella costruzione della struttura del Cum Narrativo, si è lavorato affinchè potesse essere gestita la disambiguazione a livello semantico nella traduzione delle subordinate. Infatti, è stato molto importante tenere a mente durante tutto lo sviluppo che il latino è una lingua con strutture grammaticali interpretabili in più di un modo. Ci si è posti l'obiettivo, quindi, di fornire ai ragazzi una chiave di lettura e di lavoro in più, cioè la possibilità di ragionare sulla traduzione della frase in base al contesto. I punti di partenza da cui si è voluto iniziare a impostare il lavoro sono state le difficoltà che riscontrano gli studenti nello studio di questa nuova regola. In particolare, si è notato che i ragazzi con dislessia riscontrano problematicità nella distinzione della parola *cum*, sia su un livello di lunghezza della parola che di accostamento di lettere. Un primo obiettivo, quindi, è stato l'evidenziazione della parola, tramite l'utilizzo di proprietà CSS quali ad esempio *color* (si è deciso di assegnare alla parola il colore arancio, per la sua brillantezza e contrapposizione con il colore assegnato alla subordinata del cum narrativo, l'azzurro) e *letter-spacing*, che ha permesso di distinguere e mettere in evidenza le lettere una per una. Un altro aspetto su cui ci si è voluto soffermare è la posizione delle parole della frase in latino, spesso in un ordine diverso nella traduzione italiana. Infatti, la considerazione iniziale è stata relativa alla difficoltà che incontrano i ragazzi (normotipo e DSA) di

effettuare una traduzione meno letterale e di riconoscere quindi il contesto semantico in cui inserirla. Il secondo obiettivo didattico che ci si è posti riguarda i docenti che si interesseranno all'applicazione: nonostante le tipologie di esercizi inserite siano state pensate per fornire una linearità didattica al percorso che effettuano i ragazzi della prima classe del liceo classico, l'Editor è stato realizzato affinchè possa essere costruita una qualsiasi frase di qualsiasi tipologia (ovviamente mantenendo sempre la struttura offerta dal sistema).

Ogni obiettivo è stato posto tenendo in mente che il sistema è pensato non per sostituire il tradizionale studio del latino, ma piuttosto per arricchirlo e supportarlo.

1.4 Struttura dati

Per la visualizzazione degli esercizi, la struttura dati utilizzata è il dizionario JSON¹. Questa scelta è stata motivata da due considerazioni:

1. La struttura doveva essere universale per la visualizzazione di qualsiasi tipologia di esercizio, i quali sarebbero stati distinti dai valori assegnati ai campi, gestiti quest'ultimi in maniera analoga;
2. Allo stesso modo, la struttura doveva essere facilmente estendibile, per aggiungere campi o flags al fine di integrare facilmente nuove proprietà;

L'utilizzo di essa è risultato molto utile per la costruzione dell'Editor: infatti, come spiegato nei capitoli successivi, ogni esercizio è caratterizzato dal proprio file json e in tutte le fasi verranno man mano compilati tutti i campi chiave-valore relativi (quali ad esempio: la tipologia di frase, la struttura ad albero, l'array relativo alle parole della frase, etc.). Al submit dell'esercizio, il dizionario costruito verrà scritto nel file corrispondente. Inoltre, i file json necessari alla visualizzazione degli aiuti ovvero dizionario, declinazioni, coniugazioni, congiunzioni, corrispondenze per la traduzione e via dicendo, vengono tutti aggiornati nello stesso modo.

¹JavaScript Object Notation

Capitolo 2

Tecnologie utilizzate

2.1 React

React è una libreria Javascript di natura dichiarativa, open-source e component-based, per lo sviluppo di applicazioni web-based ed in particolare per la visualizzazione di esse a lato client. Gli elementi di React sono prodotti, nella maggior parte dei casi, da quella che viene chiamata JSX (Javascript Syntax Extension), un'estensione del linguaggio Javascript puro che permette l'assemblamento delle componenti di logica e di markup, senza la necessità di doverle obbligatoriamente separare. Un'altra fondamentale caratteristica di React è l'utilizzo del Virtual DOM¹. Questo permette un aggiornamento molto più veloce degli state degli oggetti nell'applicazione, in quanto non viene manipolato direttamente il DOM, ma viene prima confrontato il suo state precedente con quello del VDOM e vengono modificati soltanto gli oggetti che ne differiscono. Grazie a questo e all'utilizzo dei Components, React è una libreria che riesce a velocizzare di molto il rendering degli elementi (ovvero di ciò che verrà visualizzato su schermo), rispetto ad altre sviluppate da frameworks e librerie alternativi.

I mattoni costituenti di React sono:

- I Components e Props
- Lo State e Lifecycle
- Gli Hooks

¹Document Object Model: standard HTML che visualizza e processa l'intera pagina web come una struttura ad albero

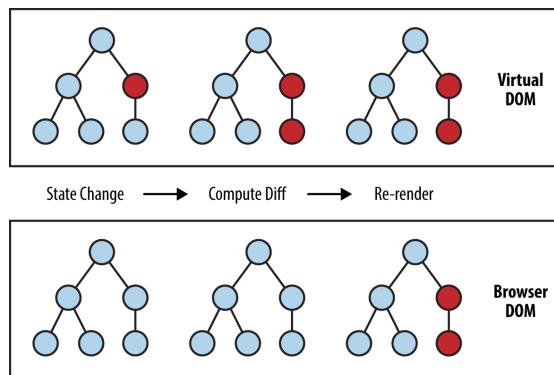


Figura 2.1. Funzionamento del Virtual DOM

Components I Components rappresentano parti autonome e riutilizzabili della User Interface che prendono in input dei valori arbitrari (i props, diminutivo di *properties*) e ritornano elementi React[3]. L'utilizzo dei Components ha particolarmente semplificato la costruzione della web-application, in quanto è stato possibile passare ed estrarre da essi oggetti e valori e utilizzarli per il funzionamento di altri Components. Bisogna specificare che i Components si comportano esattamente come delle funzioni nei confronti dei loro props, di conseguenza non possono in alcun modo modificare i propri inputs direttamente. Il compito di apportare dinamicità ai parametri è affidato agli States, i quali sono risultati fondamentali per la strutturazione dello scheletro del codice in esame.

State e Lifecycle Come dice il nome stesso, gli *states* sono degli oggetti che si occupano di fornire alle variabili e ai Components uno stato, in forma di variabili (di qualsiasi tipo) il quale può essere aggiornato, rendendo la visualizzazione dinamica. I metodi del lifecycle servono per gestire la visualizzazione e il processing dei dati in relazione al momento in cui i Components vengono renderizzati nel DOM. Un esempio nel codice è la chiamata all'API che legge le directory per definire le tipologie di esercizio nella classe Eserciziario.

In questo caso, viene utilizzato il metodo *componentWillMount()*, il quale viene eseguito prima del rendering della classe stessa. Questo permette al sistema per prima cosa di leggere i nomi delle directory tramite il fetch dell'API e impostare il valore delle tipologie al risultato della chiamata (*apiResponse*), e successivamente di caricare la pagina che mostrerà all'utente tutte le tipologie prese.

```

class Esercizio extends React.Component {
  constructor(props) {
    super(props);
    this.state = { apiResponse: "" };
  }

  callAPI() {
    fetch("http://localhost:9000/testAPI")
      .then((res) => res.text())
      .then((res) => this.setState({ apiResponse: JSON.parse(res) }));
  }

  //evento triggerato quando il component si carica
  UNSAFE_componentWillMount() {
    this.callAPI();
  }
}

```

Figura 2.2. Metodi di lifecycle per la richiesta HTTP in Esercizio

Hooks Gli Hooks sono una nuova feature di React 16.8, che permettono di utilizzare elementi di React senza definire una classe. Un esempio sono gli State Hooks, che sono risultati molto utili nello sviluppo di tutto il codice. Tramite assegnamenti di destrutturazione² in una coppia di valori, permettono di dichiarare un componente, un suo state e di aggiornare quest'ultimo. Un esempio di questo è la dichiarazione della variabile di stato del component Modal per la visualizzazione della guida nell'Editor. Inizialmente, la vista modale non è resa visibile, ed è per questo che

```
| const [showModal, setModal] = useState(false);
```

lo stato viene inizializzato a false e showModal viene assegnato al prop *visible* del Component. Appena viene innescato l'evento di click del bottone, viene aggiornato lo state del Modal tramite la chiamata alla funzione *setState(true)*, che lo renderà visibile.

2.2 React-Redux

React-Redux è una libreria molto versatile e performante, la quale può essere utilizzata non solo da React stesso ma anche da qualsiasi framework dedicato allo sviluppo di user-interfaces. Viene definita dalla pagina ufficiale in questo modo: "*Redux è un contenitore di stato prevedibile per le applicazioni JavaScript*" [4]. Essa viene utilizzata per gestire le variazioni degli state degli oggetti e fornisce uno store per l'intero albero degli state della piattaforma. Questo infatti, viene reso disponibile ai Components dell'applicazione dal root di essa ed aggiornato (non direttamente, ma ne viene creata una nuova versione) ogni volta che lo stato si modifica.

²Espressioni Javascript che permettono di estrarre singoli valori da array o Objects.

In particolare, lo store è servito per gestire i reducers della funzione `createSlice`, che ritorna oggetti con un'initialState, un nome e delle funzioni pure (chiamate appunto reducers, o riduttori). Nell'applicazione, gli oggetti ritornati sono relativi agli Slice costruiti per la gestione degli stati delle frasi subordinate, dell'Editor e di tutte le strutture afferenti al dizionario e agli helper. I reducers vengono richiamati nell'applicazione attraverso la funzione `dispatch()`, la quale innesca un cambiamento dello stato in risposta ad un'azione o un creatore di azione (ovvero una funzione che restituisce azioni). Le azioni sono responsabili della comunicazione tra lo store e i componenti dell'applicazione. Come esempio, di seguito è riportato l'utilizzo di `createSlice` per la costruzione dell'oggetto per l'Editor.

```
import { createSlice } from "@reduxjs/toolkit";

const fasiProgresso = Array.from({ length: 8 }, (_, i) => 0);

export const editorslice = createSlice({
  name: "editorslice",
  initialState: {
    progressoCreazione: 1,
    frase: '',
    declinazioni: {},
    distrattori: {},
    fasiProgresso: fasiProgresso,
    json: {},
    jsonVerbi: {},
    indexAnalisi: 0,
    indexStep: 1,
    colors: {},
    indexVerbiCompatti: 0,
    approfondimento: {},
    analisiParole: {},
    paroleDizionario: {},
    corrispondenze: {}
  },
  reducers: {
    getFrase(state, action) {
      state.frase= action.payload[0]
      state.json = action.payload[1]
    },
    setProgress(state, action) {
      state.progressoCreazione = action.payload;
    },
  }
});
```

Figura 2.3. Porzione dello Slice per l'Editor

2.3 Il framework Express

Express è un framework minimale e flessibile di Node.js ed è stato utilizzato per creare le REST API necessarie alla lettura delle directory e dei file json. Infatti, non essendo previsto per le informazioni un metodo di storage in un database per questo

lavoro di tesi, è stato deciso di organizzare il tutto intorno alla lettura e scrittura di file e directories in modo tale da permettere a docenti esterni di effettuare le modifiche opportune direttamente sui file che l'applicazione leggerà.

Uno dei concetti base di express (e quello che poi ha permesso il funzionamento dello scambio dei dati) è il **routing**: esso consiste nella modalità con cui un'applicazione risponde alle richieste da parte del client ad un URI³, tramite metodi HTTP (quali ad esempio get e post). In particolare, Express gestisce queste richieste tramite un'istanza *app* della sua classe. Nell'applicazione, l'oggetto è una funzione Javascript passato al server HTTP per gestire e rispondere alle richieste e il server è in continuo ascolto sulla porta 9000. Nella figura 2.4, si può vedere come è stata implementata la REST API, che comunica con il client leggendo la directory "exercise" e tutte le cartelle al suo interno per prenderne i nomi. La lettura delle directory e dei file è stata realizzata tramite il modulo *fs* di Node.js, il quale fornisce molte funzionalità per accedere e interagire con il filesystem [5].

```
var express = require("express");
var router = express.Router();
var fs = require("fs");

router.get("/", function (req, res, next) {
  var files = Object();
  var dir = "";
  var directories = Object();
  files = fs.readdirSync("../exercise", function (err, data) {
    if (err) {
      res.writeHead(404);
      res.end(JSON.stringify(err));
      return;
    }
  });
  files.forEach((file) => {
    dir = `../exercise/${file}`;
    if (fs.lstatSync(dir).isDirectory()) {
      directories[file] = fs.readdirSync(dir, function (err, data) {
        if (err) {
          return;
        }
      });
    }
  });
  res.send(directories);
});

module.exports = router;
```

Figura 2.4. REST API per la lettura delle tipologie nei nomi delle directory

³Uniform Resource Identifier

2.4 Librerie grafiche

Ai fini della realizzazione e visualizzazione dei Components della piattaforma, sono state utilizzate altre librerie: Material-UI, antd (libreria UI per React di Ant Design) e PrimeReact. L'utilizzo di antd è stato fondamentale per la gestione di componenti quali in particolare: Modal, Button, Tooltip, message e Drawer.

Per la realizzazione delle mappe concettuali pensate per il cum narrativo, è stata utilizzata la libreria React Flow, volta a costruire applicazioni node-based e per creare grafi personalizzabili sia nei nodi che negli archi [6].

Un'ulteriore libreria utilizzata è react-color-list, un *color palette manager* per React [7], necessaria per la realizzazione e visualizzazione del component ColorList nella sezione **Ulteriori Informazioni** nell'Editor.

Capitolo 3

Modularizzazione

Il primo passo effettuato al fine di rendere estendibile l'architettura è stato quello di modularizzare il codice già presente e le strutture dati per la visualizzazione delle frasi. In particolare, il lavoro di tirocinio è partito con l'analisi delle affinità e differenze che avevano le strutture che descrivevano le tipologie e dei Components che le rappresentavano, con l'obiettivo di costruire strutture completamente autodescrittive che potessero essere spostate nel sistema senza che il loro funzionamento dovesse cambiare. Gli obiettivi quindi erano i seguenti:

- Implementare delle strutture universali e facilmente ampliabili per visualizzare il tipo di esercizio
- Costruire un sistema unico volto alla rappresentazione di qualsiasi tipologia di esercizio
- Estendere le strutture base degli esercizi per poter visualizzare nuovi tipi di aiuti e/o suggerimenti

3.1 Dizionario degli esercizi

Per rappresentare un esercizio la scelta è ricaduta sulla struttura dati del **dizionario nidificato JSON**: questa scelta era già stata adottata nel lavoro precedente, in vista di estensioni future. Il lavoro è iniziato generalizzando la struttura del dizionario in modo tale da poter utilizzare la stessa per diverse tipologie di frasi. Dall'analisi delle strutture di partenza, si è notato che i campi che differivano nella descrizione delle diverse tipologie di frasi erano proprio quelli relativi alla descrizione delle subordinate stesse (quindi, ad esempio, una frase completa aveva l'analogia struttura di una frase infinitiva se non per i campi *"infinitiva"* e *"completiva"*, i quali

contenenti le informazioni necessarie per quella specifica subordinata nella frase). Il ragionamento è quindi partito da questi, al fine di poter rappresentare frasi diverse con coppie chiave-valore delle quali solo il secondo potesse differenziarsi dagli altri. I campi descriventi l'esercizio, in primo luogo, si occupano di definire la tipologia della frase in questione. Questo campo è risultato cardine nell'Editor per la creazione delle directory che descrivono l'esercizio. Infatti, il sistema leggerà la tipologia inserita e collocherà la frase all'interno di una directory, se ne esiste già una con questo nome, oppure creerà una nuova cartella avente come nome la tipologia stessa. Si è deciso, inoltre, di non fare più distinzione tra frase principale e frase subordinata, in quanto (come verrà spiegato più precisamente in seguito) il sistema analizzerà le proposizioni che compongono la frase in maniera analoga. Nel dizionario che descrive l'esercizio è quindi presente il campo "*proposizioni*", costituito da dizionari più piccoli che descrivono la struttura di ciascuna di esse, indicizzate e aventi nella chiave "nome" l'esatta tipologia. In questo modo è stato possibile effettuare i controlli adeguati e linkare gli attributi corretti della tipologia ai file json che ne descrivono gli aiuti e la struttura (come ad esempio il colore e le regole relative al verbo e soggetto, Figura 3.2).

```

"Bona domina domino dicit ancillam Romam ire": {
  "tipologia": "infinitiva",
  "numero subordinate": 1,
  "parole": ["Bona", "domina", "domino", "dicit", "ancillam", "Romam", "ire"],
  "proposizioni": {
    "1": {
      "nome": "principale",
      "Bona": ["attributo", "nominativo"],
      "domina": ["soggetto", "nominativo"],
      "domino": ["terminale", "dativo"],
      "dicit": "verbo"
    },
    "2": {
      "nome": "infinitiva",
      "ancillam": ["soggetto", "accusativo"],
      "Romam": ["moto a luogo", "accusativo"],
      "ire": "verbo"
    }
  }
},
  "principale": {
    "soggetto": "nominativo",
    "verbo": "coniugato",
    "colore": "#a635aa"
  },
  "infinitiva": {
    "soggetto": "accusativo",
    "verbo": "verbo infinito",
    "colore": "#72c407"
  }
}

```

Figura 3.1. Struttura iniziale della frase infinitiva

Figura 3.2. Strutture per i suggerimenti divisi per tipologia

Gli indici assegnati alle proposizioni sono serviti per descrivere l'ordine in cui dovranno essere tradotte e sono risultati necessari e indispensabili per aggiungere facilmente nuovi campi con cui gestirne le proprietà. Un esempio di questo è il campo *visibile*, creato per gestire la visibilità della frase da parte dell'utente negli esercizi sulla nuova struttura del Cum Narrativo. Grazie ad esso, è bastato assegnare a ciascun indice un valore tra true e "none" e settare la proprietà display di CSS allo stesso valore.

```

    "ordineTraduzione": ["1", "2"],
    "traduzione": {
      "1": ["la signora", "buona", "dice", "al signore"],
      "2": ["che", "l'ancella", "va", "a Roma"]
    },
    "traduzioniParole": {
      "bona": "buona",
      "domina": "la signora",
      "domino": "al signore",
      "dicit": "dice",
      "ancillam": "l'ancella",
      "Romam": "a Roma",
      "ire": "va"
    },
    "visible": {
      "1": true,
      "2": true
    }
  }
}

```

Figura 3.3. Campi per la traduzione e visibilità delle proposizioni in base agli indici

Gli indici hanno anche permesso di gestire frasi con subordinate multiple (l’obiettivo, infatti, era quello di poter permettere ad un docente di far eseguire un esercizio più complesso, con 3 o più tipologie di subordinate distinte), in quanto vengono rappresentate allo stesso modo all’interno della struttura e l’Esercizio è stato strutturato in modo tale da scorrere tutte le chiavi del dizionario delle proposizioni e visualizzare, quindi, tutte le frasi. Ogni frase è stata inserita all’interno di un proprio file json personale, collocato nella directory che prende il nome della tipologia che ne descrive la struttura grammaticale latina.

3.2 Importanza degli Slice

La gestione delle variabili utilizzate per la visualizzazione degli esercizi è affidato agli Slice, strutture di React-Redux descritte nel paragrafo omonimo del secondo capitolo. Successivamente alla strutturazione dei campi della frase, infatti, il lavoro è stato in gran parte incentrato sulla costituzione di un unico slice per il mantenimento delle informazioni delle frasi, indipendentemente dalla loro tipologia, per poi costruire i Components dell’Esercizio intorno ad esso. La strutturazione di tale slice è risultata l’aspetto più importante ai fini della modularizzazione.

Lo slice creato, che prende il nome di "frasiSubordinate", ha un *initialState* che prende come parametri nove oggetti (come si può vedere nella Figura 3.4). Gli oggetti sono volti alla memorizzazione nello store di informazioni quali: le parole analizzate per ogni tipologia e frase in analisi, lo stato del progresso per l’esercizio che si sta svolgendo, il numero della frase che si sta svolgendo (basato sull’ordine in cui la frase

compare all'interno della directory), il numero delle frasi analizzate per ogni tipologia, l'indice di ogni tipologia, l'array di tutte le tipologie e la visibilità delle proposizioni della frase. Lo slice si occupa di tutte le tipologie di subordinate in questo modo: inizialmente, tutte le strutture (tranne l'array delle tipologie, inizializzato di default con la principale) sono vuote. Ad un esercizio si accede tramite l'Eserciziario. Al load di quest'ultimo, viene effettuata la chiamata alla REST API la quale ritorna un dizionario (con il nome di *apiResponse*), avente come chiavi i nomi delle directory (che ricordiamo vengono assegnati ai nomi delle tipologie che il sistema propone), e come valori un array contenente i nomi di tutti i file presenti all'interno di quella determinata directory (che corrispondono ai file json che rappresentano la frase di quella tipologia). Nell'Eserciziario, vengono visualizzati tanti Link al Component di Esercizio quante sono le chiavi di *apiResponse*. In ogni Link vengono passati tre parametri: la tipologia della frase, il nome dello slice (ovvero *frasiSubordinate*, analogo per tutti) e files, che come valore prende *apiResponse*.

```
import { createSlice } from "@reduxjs/toolkit";

/*Porzione codice per lo slice delle proposizioni*/
var tipologie = ["principale"];

//Ho un dizionario che prende le parole analizzate per ogni tipologia
const paroleAnalizzate = {}

//creo un dizionario che contiene un oggetto di frasi per ogni tipologia
const statoProgresso = {}

//contatori differenziati per tipologia
const countFraseCorrente = {};

const countFraseCompletata = {};

//dizionario che tiene in mente l'indice per ogni tipologia
const index_tipologie = {};

export const fraseSlice = createSlice({
  name: "frasiSubordinate",
  initialState: {
    tipologia: "",
    index: index_tipologie,
    numeroFraseCorrente: countFraseCorrente,
    numeroFrasiCompletate: countFraseCompletata,
    statoProgresso: statoProgresso,
    paroleAnalizzate: paroleAnalizzate,
    tipologie: tipologie,
    subtaskProgresso: 1,
    visibility: {}
  },
  
```

Figura 3.4. Porzione di Slice per le informazioni nell'Esercizio

Quando si accede ad un Esercizio, vengono costruite tutte le strutture dati presi i valori dell’API e inserite poi nello store tramite l’invocazione al reducer (*checkStrutture*) che se ne occupa nello slice. In particolare, vengono costruiti i dizionari nidificati aventi come chiavi principali le tipologie presenti nella directory e come valori di essi N oggetti Javascript per le N frasi contenute nella directory. Questi sono in particolare:

- L’array per lo **stato del progresso** della frase, costituito dai primi due elementi che descrivono il tipo di esercizio scelto (di apprendimento o autoverifica), il nome della subordinata che si sta analizzando, e tanti valori quante sono le tipologie per indicare il numero di passaggi completati dallo studente;
- Il dizionario per il conteggio del numero della frase in analisi e delle frasi completate per ogni tipologia;
- Il dizionario che assegna ad ogni tipologia presente un indice, il quale serve per estrarre il valore della tipologia nell’array per lo stato del progresso e quindi per la visualizzazione di tutti gli steps dell’Esercizio nell’ordine giusto;
- Il dizionario nidificato che, per ogni tipologia, restituisce delle coppie chiave-dizionario in cui le chiavi corrispondono alle frasi presenti nella directory e i valori le parole analizzate fino a quel momento nella frase;

Grazie a questi valori dello slice, si riesce ad accedere facilmente alla frase dell’esercizio che si sta svolgendo, e quindi alla sua struttura JSON descritta in precedenza.

3.3 Components dell’Esercizio

La struttura iniziale prevedeva una distinzione tra EsercizioSubordinata e EsercizioSubordinate, due Components volti uno alla visualizzazione di un esercizio con una subordinata singola, e l’altro con più subordinate (a loro volta con componenti differenti a seconda della tipologia di subordinata). Con l’utilizzo degli indici e delle strutture sopracitate, si è riusciti a creare un unico Component Esercizio, il quale opera attraverso l’estrazione di informazioni da esse e cicla tutte le tipologie delle proposizioni presenti nell’ordine descritto nella struttura JSON. L’Esercizio è ora composto da sette Components, che rappresentano le sette fasi di un esercizio di traduzione, senza distinzioni di tipologia o numero di subordinate. Le informazioni necessarie per il corretto funzionamento della pagina, vengono passate come props e sono state generalizzate attraverso la creazione di ulteriori strutture JSON inserite

all'interno della directory chiamata *helper*. Un esempio di questo è il file JSON per le domande di approfondimento, che contiene al suo interno le domande che vengono richieste allo studente nell'esercizio di approfondimento, suddivise per frase e indice di proposizione. In questo caso, il Component passerà i suoi props in questo modo:

```
: statoProgresso[0] === true ? (
  statoProgresso[index_subordinata + 1] === 4 ? (
    <RipassoProposizioni
      ordineProposizioni={ordineProposizioni}
      slice={slice}
      tipoSubordinata={tipo}
      objDizionario={objDizionario}
      fraseCompleta={frase}
      parole={parole}
      suggerimento={true}
      domande={[
        <div className="domanda-approfondimento">
          {domandeApprofondimento[frase][statoProgresso[1]].domande["1"]}
        </div>,
        <div className="domanda-approfondimento">
          {domandeApprofondimento[frase][statoProgresso[1]].domande["2"]}
        </div>,
      ]}
      risposte={domandeApprofondimento[frase][statoProgresso[1]].risposte}
      intercambiabili={false}
    />
```

Il procedimento è analogo per tutti i Components: l'indice di subordinata corrisponde all'indice assegnato alla sua tipologia e lo stato del progresso è relativo alla frase che si sta analizzando. Si è anche valutato il caso in cui in una frase ci siano più proposizioni della stessa tipologia: essendo l'indice di esse analogo, dopo l'analisi della prima si presentava il problema di aggiornare il suo valore nello stato del progresso in modo tale da poter analizzare anche le altre. È stato quindi inserito un controllo nel momento dell'estrazione dell'indice della tipologia stessa tale per cui il Component Esercizio invocherà il reducer che ne resetterà il valore nel momento in cui il valore risulta al massimo ma l'analisi della frase intera non è ancora terminata.

Per concludere, con l'ipotesi che i docenti abbiano a disposizione le directory dei file per estenderle anche manualmente, si è pensata la situazione in cui un docente crei una directory con il nome di una tipologia ma non vi inserisca alcuna frase dentro (ed in questo caso il valore della tipologia in questione nel dizionario *apiResponse* sarà una lista vuota): in questo caso il sistema non risponderà portando l'utente alla pagina dell'Esercizio, ma ad una pagina di errore, che gli notifica che non esiste ancora alcuna frase pronta per quella tipologia (Figura 3.5).

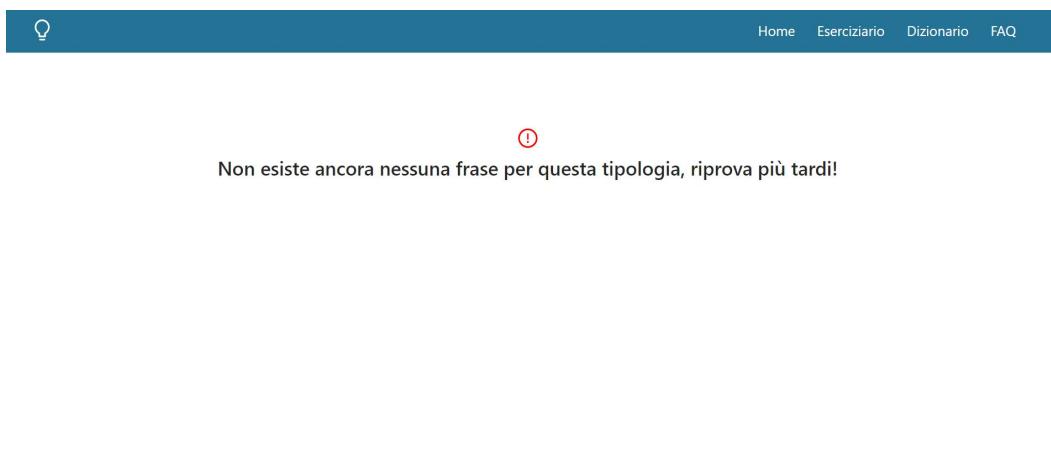


Figura 3.5. Pagina di errore mostrata per una directory vuota

3.4 Cum narrativo

Dopo la modularizzazione, si è ritenuto opportuno testarla inserendo nel sistema un esercizio di una tipologia più complessa che permettesse la costruzione di nuove strutture di help e visualizzazione. La scelta è ricaduta sulla struttura del **cum narrativo**, anche in un'ottica di coerenza didattica nei confronti del piano che i docenti di latino seguono durante il primo anno di liceo. Il lavoro è iniziato approfondendo gli aspetti importanti di tale struttura, per poi capire quali di essi potessero essere evidenziati da parte della piattaforma e su quali ci si dovesse concentrare, soprattutto da un punto di vista della dislessia. I nuovi argomenti inseriti in Latino facile riguardano, quindi:

- Disambiguazione nella funzione del cum, che può essere preposizione o congiunzione subordinante (e quindi necessita di diverse strutture)
- Consecutio temporum, e quindi la relazione tra i tempi verbali
- Traduzione implicita e esplicita della frase, e quindi aggiunta di un ragionamento concettuale sul suo significato

Il tema *disambiguazione* è centrale sia nel latino in generale, sia nello studio che i ragazzi con DSA ne fanno. Infatti, distinguere la funzione delle parole in base al contesto della frase risulta essere un grande scoglio per gli studenti di questa materia. Questo è l'aspetto su cui si è incentrato maggiormente il mio lavoro: guidare il ragazzo passo passo nell'identificazione delle strutture importanti (in questo caso

il cum e i verbi d'appoggio) e nella comprensione di quale funzione svolgono nella frase.

3.4.1 Interfaccia utente

La prima pagina si presenta come un esercizio già presente nella struttura della precedente tesi: infatti, le task sono rimaste invariate. Sono state, però, effettuate delle aggiunte negli aiuti, nell'analisi del verbo e nella traduzione.

Partendo dagli aiuti, nel momento in cui il ragazzo vuole analizzare la funzione del **cum**, può cliccare sul ToggleButton *Aiuto nascosto*, che presenta, per ogni parola nella frase, (come già strutturato per tutte le frasi) un riepilogo del corrispondente contenuto del dizionario ed eventuali tabelle. In particolare, per la parola cum, si è deciso di presentare una serie di domande semplici a doppia opzione per stimolare il ragionamento del ragazzo nei confronti della struttura del cum narrativo. La prima domanda chiede al ragazzo se abbia compreso la funzione della parola.

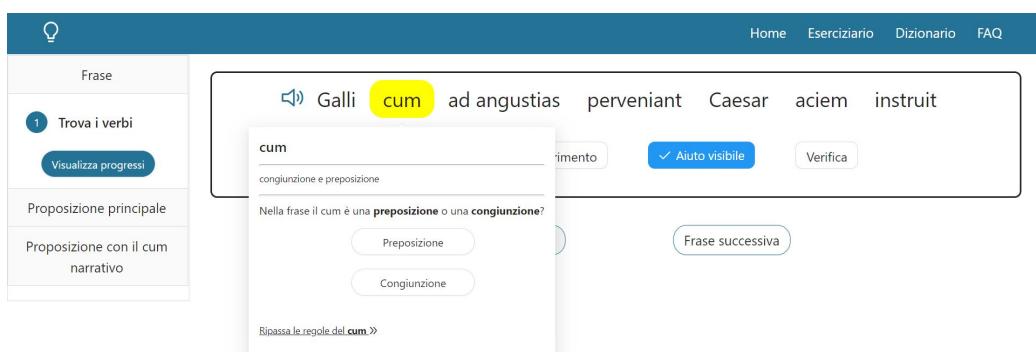


Figura 3.6. Prima domanda nel Tooltip del cum

In qualsiasi momento, si potrà visualizzare il Drawer (finestra al lato) che spiega le funzioni della parola cum, cioè quando può essere considerata una proposizione (con degli esempi in merito) e quando invece è una congiunzione subordinante, cliccando su *Ripassa le regole del cum*. È importante specificare che si è cercato di non dare al ragazzo la "risposta pronta", ma piuttosto di farlo ragionare su eventuali errori; infatti (come si può vedere nella Figura 3.9) la domanda che viene presentata nel momento in cui egli seleziona la risposta errata (che in questo caso è *Proposizione*), chiede se la struttura del cum nella frase è coerente con la funzione scelta.

Figura 3.7. Drawer consultabile per le funzioni del cum: sezione Proposizione

Figura 3.8. Drawer consultabile per le funzioni del cum: sezione Congiunzione

Figura 3.9. Seconda domanda nel caso della prima risposta errata

Se il ragazzo risponderà in modo errato anche a questa domanda, gli si presenterà il Drawer nella figura 3.7, perché possa ragionare sulla struttura corretta del cum nella frase. Successivamente, gli sarà data la possibilità di rispondere nuovamente alla seconda domanda per vedere se ha capito. Una volta aver dato la risposta corretta, il sistema ritornerà alla prima domanda (Figura 3.6).

La seconda domanda presentata dal sistema fa ragionare il ragazzo sulla struttura del cum nella frase, e lo fa entrare nell'argomento cardine dell'esercizio: il cum narrativo. In particolare, gli viene data la possibilità di ragionare a primo impatto sul tempo del verbo che affianca la congiunzione.

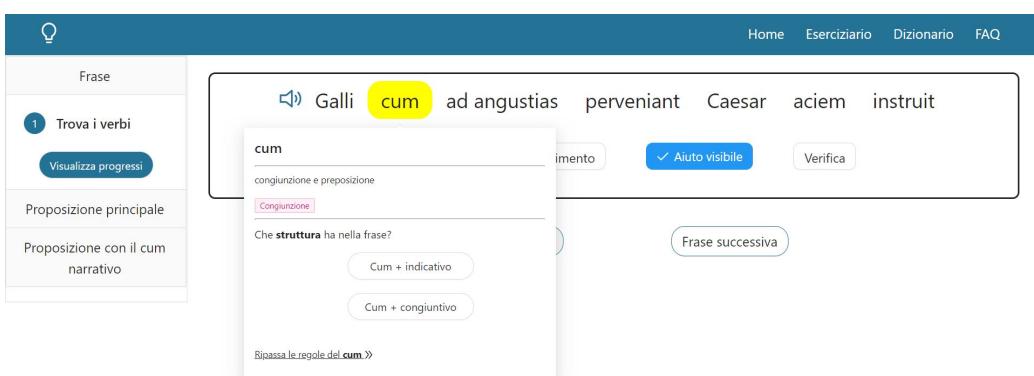


Figura 3.10. Seconda domanda nel Tooltip del cum

Dalla seconda domanda in poi, ogni volta che si procede negli step, vengono presentati dei Tag di colori diversi sopra alla domanda, per ricordare al ragazzo le risposte corrette che ha selezionato in precedenza e per guiderlo nella risposta della domanda corrente.

In più, d'ora in avanti, ogni risposta errata riporterà al Drawer delle regole del cum, nella sezione dedicata al cum come congiunzione.

La terza domanda è volta alla comprensione della struttura del cum narrativo nella frase in analisi: viene quindi chiesto allo studente per quale parola il cum fa da congiunzione subordinante. Come opzioni vengono date il verbo nella frase del cum narrativo (ovvero, la risposta esatta), e un valore preso in modo randomico tra le parole della frase intera, tolta ovviamente la parola cum e il verbo stesso. Quindi, ad esempio, la parola *angustias*, prima opzione nella figura 3.12, ad un load successivo dell'esercizio potrebbe diventare Galli, aciem e via dicendo.



Figura 3.11. Terza domanda nel Tooltip del cum

La quarta e ultima domanda è dedicata all'apprendimento del tempo verbale del cum narrativo. Dopo aver capito che il cum affianca il verbo (*perveniant*, nell'esempio), lo studente quindi ragionerà su quest'ultimo, sempre potendo consultare il Drawer in cui è specificato in quali casi esso è indicativo e in quali invece è congiuntivo.

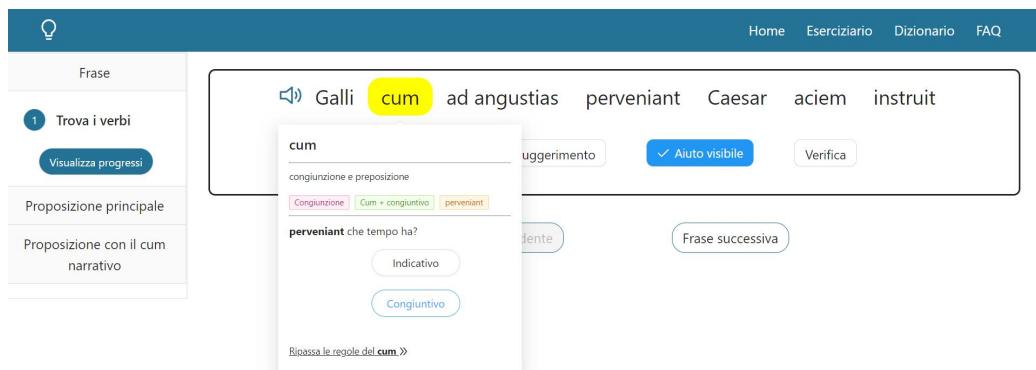


Figura 3.12. Quarta domanda nel Tooltip del cum

Ricapitolando, il ragazzo è stato guidato passo passo in questi step:

1. Identificazione della funzione corretta del cum
2. Comprensione della struttura del cum nella frase
3. Individuazione del verbo che il cum affianca e il suo tempo

Con questi presupposti, viene quindi presentata al ragazzo una **mappa concettuale** consultabile in qualsiasi momento che riassume e condensa queste nuove conoscenze legate al cum. Si è scelto di utilizzare la struttura della concept-map per svariati motivi, partendo dal fatto che esse risultano molto utili per sostenere la comprensione delle spiegazioni orali e scritte nei ragazzi con difficoltà nel processare informazioni uditive e del linguaggio scritto (quali quindi i ragazzi con dislessia) [8]. Nel lavoro con bambini e ragazzi con DSA si è largamente osservato quanto le mappe concettuali siano uno strumento prezioso poiché permettono l'organizzazione delle informazioni attraverso l'uso di poco testo (parole chiave), l'uso di strategie associative (immagini, colori, forme), una chiara evidenziazione delle connessioni logiche tra i concetti chiave (frecce) e l'utilizzo dello spazio di un unico foglio per riportare contenuti anche complessi e articolati [9]. Si è pensato anche che possano rappresentare il lavoro che ha fatto il ragazzo nei passi che l'hanno portato all'analisi del cum precedentemente.

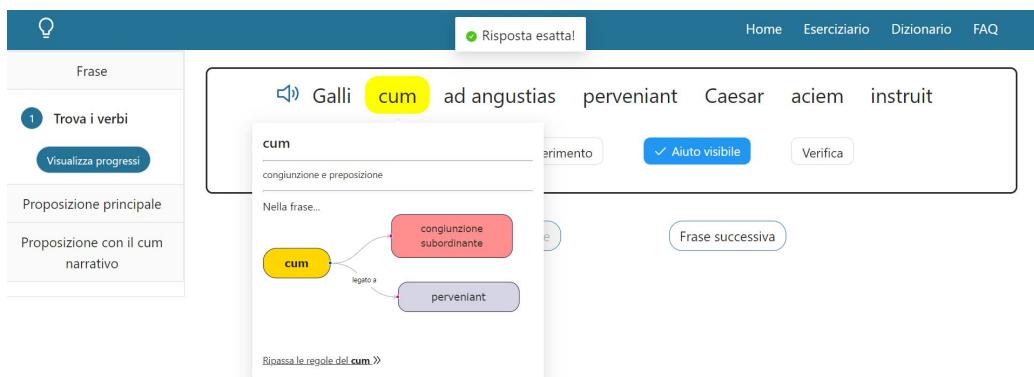


Figura 3.13. Mappa concettuale del cum

La mappa mostra le due informazioni principali che il ragazzo deve ricordare mentre sta svolgendo un esercizio sul cum narrativo: la funzione del cum e il verbo a cui è legato. Si è cercato di distinguere il più possibile le informazioni in modo tale che un ragazzo con DSA possa assumerle singolarmente in concetti separati; è per questo che il nodo della funzione del cum è in rosso e quello del verbo è in grigio chiaro, mentre la radice (la parola principale, in questo caso *cum*) è in oro. Come detto in precedenza, la creazione della mappa è stata possibile grazie all'utilizzo della libreria React-Flow. Di seguito, è mostrata una porzione di codice relativa alla

visualizzazione dei nodi della mappa, in particolare del nodo del verbo e del cum, in cui *verbo* è passato come props al Component in modo tale da poter visualizzare per qualsiasi frase il verbo corretto.

Cum è un input node, quindi è costituito da un unico punto di origine da cui partono gli archi che lo collegano agli output nodes (di struttura opposta), che sono appunto il nodo relativo al verbo e alla funzione.

```
{  
  id: "node-2",  
  type: "input",  
  sourcePosition: "right",  
  data: { label: <div>cum</div> },  
  position: { x: 0, y: 50 },  
  style: {  
    width: 100,  
    borderRadius: 20,  
    border: "1px solid",  
    backgroundColor: "gold",  
    fontSize: 15,  
    fontWeight: "bold",  
    fontFamily: "Verdana",  
  },  
},  
{  
  id: "node-3",  
  type: "output",  
  targetPosition: "left",  
  position: { x: 200, y: 100 },  
  data: { label: <div>{verbo}</div> },  
  style: {  
    background: "#D6D5E6",  
    color: "#333",  
    border: "1px solid #222138",  
    borderRadius: 15,  
    width: 180,  
    fontSize: 15,  
    fontFamily: "Verdana",  
  },  
},
```

Figura 3.14. Porzione di codice della mappa concettuale del cum

La stessa impostazione è stata organizzata per l'analisi del verbo della principale, per far riprendere la struttura della consecutio temporum e quindi per stimolare il ragionamento sulla funzione del verbo. Quindi, se i ragazzi cliccano sull'aiuto del verbo della principale, si ritroveranno un'interfaccia analoga a quella del cum che gli chiede se esso sia un tempo principale o tempo storico. In qualsiasi momento, inoltre, possono consultare il Drawer della consecutio temporum cliccando su "Visualizza le regole della consecutio temporum" (Figura 3.16).

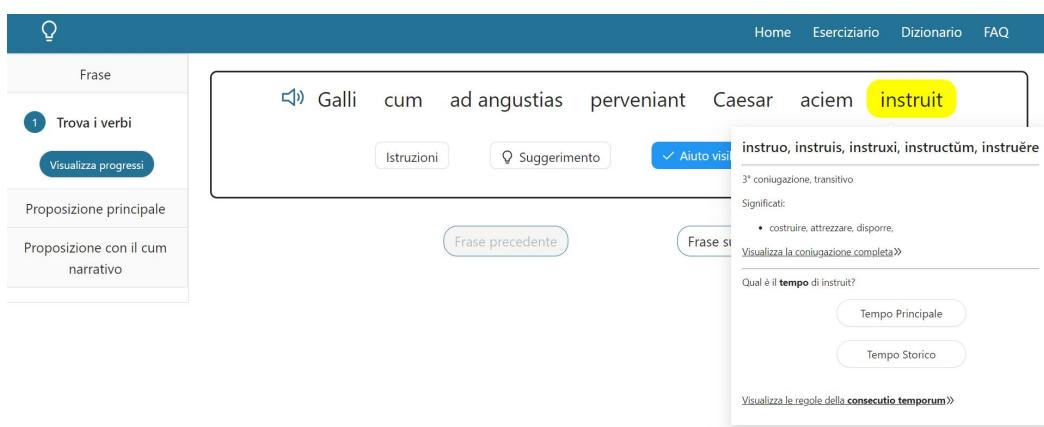


Figura 3.15. Prima domanda nel Tooltip del verbo della principale

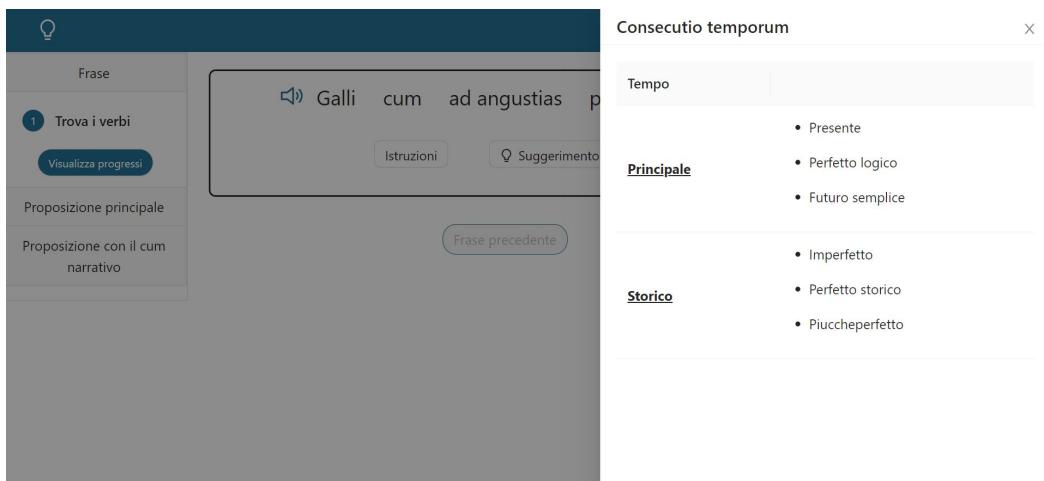


Figura 3.16. Drawer per la consecutio temporum

Se lo studente risponde correttamente, visualizzerà la mappa concettuale che lega il verbo alla sua funzione nella frase (Figura 3.17).

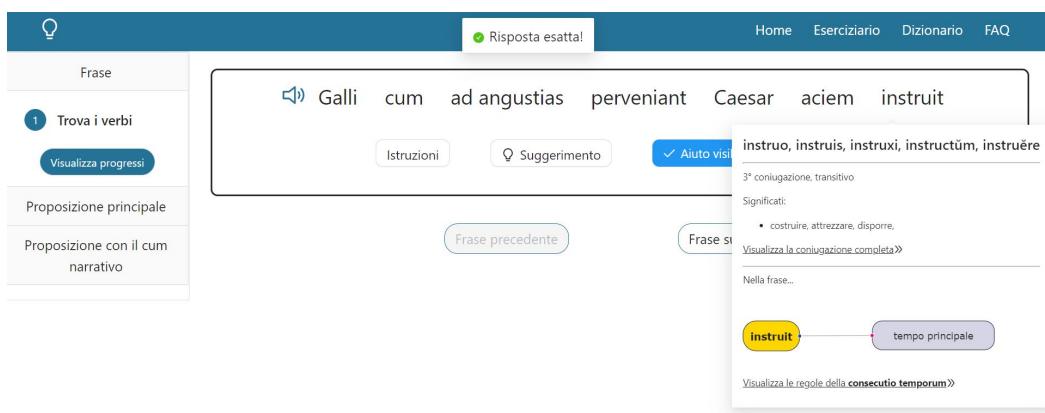


Figura 3.17. Mappa concettuale per il verbo della principale

È importante ricordare che le domande e le mappe saranno disponibili all’uso e visualizzazione da parte dello studente in tutto lo svolgimento dell’esercizio, e quindi egli potrà consultarli e/o rispondere alle domande in qualsiasi momento vorrà.

Lo svolgimento dell’esercizio è analogo alla struttura che il sistema presenta di default per le altre tipologie. Quando, però, lo studente dovrà analizzare il verbo della proposizione con il cum narrativo, dovrà ragionare su un altro campo: *Esprime*. Potrà consultare gli aiuti nella sezione *Aiuto nascosto* e, nel caso ne avesse bisogno, potrà cliccare sull’icona QuestionCircleOutline, che aprirà un Modal per il ripasso del rapporto temporale del verbo con il cum narrativo (Figura 3.19).

The analysis page for the verb 'cum' shows the sentence: 'Galli cum ad angustias perveniant Caesar aciem instruit'. Below the sentence are five boxes for analysis: 'MODO' (Indicativo, Infinito, Congiuntivo), 'TEMPO' (Presente, Perfetto, Imperfetto, Pluperfetto), 'FORMA' (Attiva, Passiva), 'PERSONA' (1° persona, 2° persona, 3° persona), and 'NUMERO' (Singolare, Plurale). A sixth box, 'ESPRIME', contains the question mark icon. Buttons at the bottom include 'Istruzioni', 'Aiuto visibile', and 'Verifica'. Navigation buttons 'Frase precedente' and 'Frase successiva' are at the bottom.

Figura 3.18. Pagina per l’analisi del verbo del cum narrativo

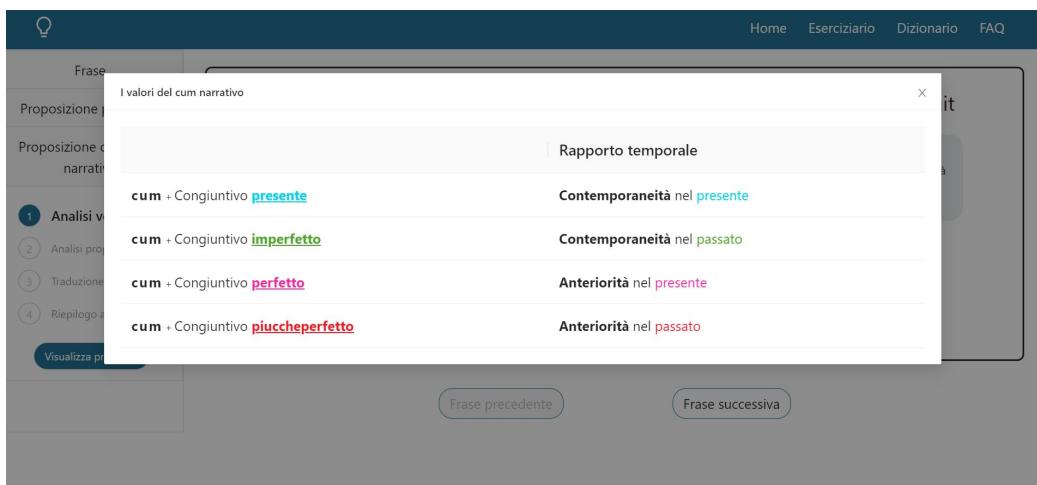


Figura 3.19. Modal per il ripasso dei tempi verbali nel cum narrativo

I tempi sono stati evidenziati con colori diversi per permettere la distinzione tra una definizione e l'altra.

Nella fase della traduzione, è stato aggiunto un ulteriore aiuto per l'identificazione e l'ordine corretto del soggetto nella frase. Infatti, ci si è concentrati sulla traduzione implicita della frase (quindi con il gerundio) e si è cercato di aiutare i ragazzi che fanno difficoltà ad individuare la posizione del verbo e del soggetto.

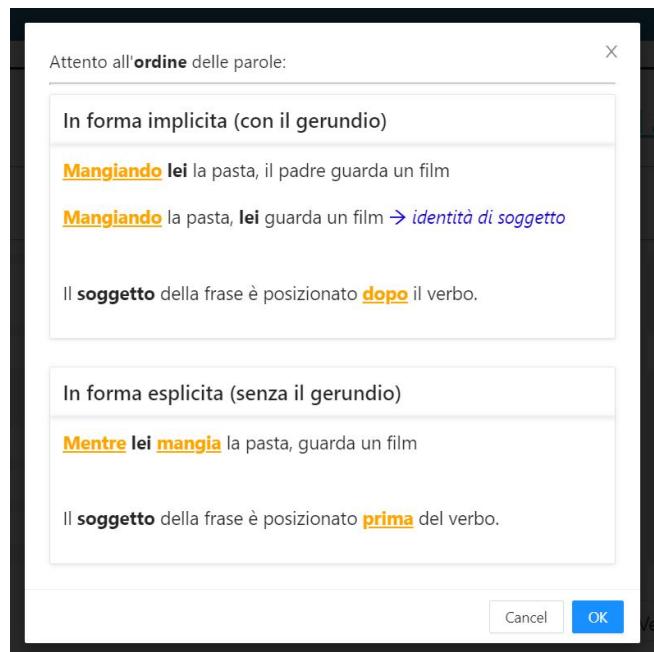


Figura 3.20. Modal per il soggetto del cum narrativo

L'obiettivo della fase di traduzione è stato l'introduzione del concetto di **ambiguità semantica** nella traduzione con il gerundio.

Figura 3.21. Traduzione corretta con ambiguità di significato

Infatti, dopo aver tradotto correttamente la frase, spunterà un Modal non chiudibile in cui verrà fornito un input di ragionamento riguardo a questo concetto. Il ragazzo potrà quindi "sbloccare" una proposizione ulteriore, facente parte sempre della stessa frase, che analizzerà e alla fine lo potrà aiutare a capire qual è la traduzione esplicita migliore da affidare alla frase del cum narrativo.

Figura 3.22. Modal non chiudibile per l'introduzione alla nuova frase



Figura 3.23. Proposizione finale sbloccata, con i Tooltip che ne indicano il tipo

Quando il ragazzo cliccherà su *Continua l'analisi*, l'esercizio riprenderà dalla task successiva, ovvero il Riepilogo, come da struttura. Noterà, però, che nel Sider (il Menu al lato sinistro) è stata aggiunta una proposizione nuova da analizzare, ovvero quella che lo aiuterà a capire il contesto della frase (nel caso in esempio è la proposizione finale "*ut exercitus castra defendat*"), e, a fianco ai bottoni degli aiuti, è comparso il bottone "*Mappa concettuale*".

Cliccando su di esso, il ragazzo potrà visualizzare la mappa concettuale generale della frase che ha appena analizzato, con tutte le informazioni collezionate sul cum narrativo. Questa è stata realizzata distinguendo la subordinata con il cum dalla proposizione principale, ed i nodi corrispondono a quelli risultanti dall'analisi del verbo della principale e della funzione del cum nei Tooltip degli Aiuti.

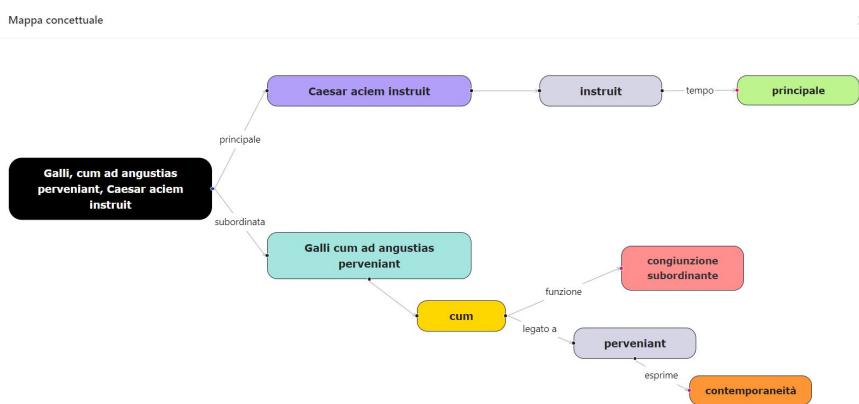


Figura 3.24. Mappa concettuale generale

L'esercizio prosegue con l'analisi della nuova subordinata, fino a quando non si arriva al Riepilogo, ovvero la pagina che presenta le traduzioni della frase. In questa pagina lo studente visualizzerà un nuovo bottone in basso a destra, che lo invita a trovare la seconda traduzione del cum narrativo.

The screenshot shows a user interface for analyzing Latin phrases. On the left, there's a sidebar with navigation options: 'Frase', 'Proposizione principale', 'Proposizione con il cum narrativo', and 'Proposizione finale'. The main content area displays the Latin phrase 'Caesar aciem instruit Galli cum ad angustias perveniant ut exercitus castra defendat' with its English translation: 'Cesare dispone l'esercito giungendo i Galli nelle strettoie affinchè (l'esercito) difenda gli accampamenti'. Below the text is a yellow thumbs-up emoji. At the bottom right of the main area is a button labeled 'Trova l'altra traduzione' (Find another translation). Navigation buttons 'Frase precedente' and 'Frase successiva' are at the bottom center. The top right of the screen has links to 'Home', 'Eserciziario', 'Dizionario', and 'FAQ'.

Figura 3.25. Pagina del riepilogo con il bottone per la traduzione del cum narrativo

Cliccando sul bottone, si aprirà una vista modale chiudibile nella quale lo studente potrà ragionare sulla disambiguazione della frase. Il ragazzo potrà rileggere la traduzione della frase, ma sarà assente quella del cum narrativo in quanto dovrà trovare quella esplicita tra le opzioni date. Il modal è chiudibile in quanto così lo studente potrà in ogni momento consultare la frase latina, se ne ha necessità.

The screenshot shows a modal window titled 'Ora sei pronto a trovare un'altra traduzione per il cum narrativo, che dipende dal contesto della frase'. Inside the modal, there is a text box containing the Latin phrase 'Cesare dispone l'esercito ? affinchè (l'esercito) difenda gli accampamenti'. Below this, three options are listed in separate bubbles: 'Mentre i Galli giungono nelle strettoie', 'Poichè i Galli giungono nelle strettoie', and 'Nonostante i Galli giungano nelle strettoie'. Each option has a question mark icon next to it. At the bottom of the modal is a blue 'Verifica' button. The background of the modal is semi-transparent, showing parts of the previous interface.

Figura 3.26. Modal per la seconda traduzione del cum narrativo

Di fianco ai bottoni, è possibile consultare un Tooltip che spiega la funzione della congiunzione, e in quali tipi di subordinate è presente. In basso, lo studente potrà leggere anche degli esempi di frasi italiane con tale congiunzione.

Una volta che lo studente ha cliccato sulla risposta corretta, la frase verrà mostrata

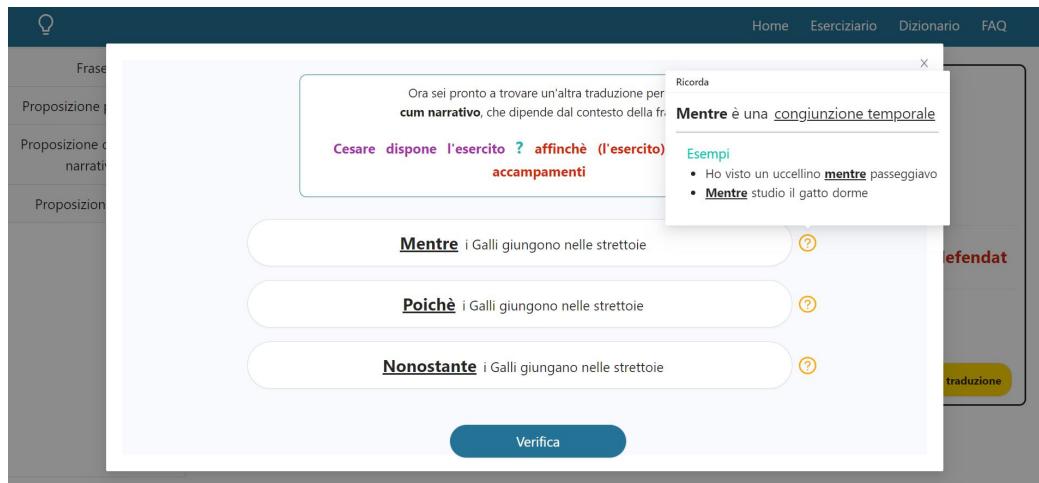


Figura 3.27. Modal per la seconda traduzione del cum narrativo

per intero insieme alla nuova traduzione. Il modal non si chiuderà in quanto si è pensato che potrebbe risultare didatticamente utile far rileggere allo studente la frase appena creata per stimolare il ragionamento concettuale dietro al contesto della stessa.



Figura 3.28. Versione finale della frase con la traduzione esplicita

3.4.2 Nuovi campi nella struttura dati

Per la realizzazione degli helper e delle nuove features aggiunte con il cum narrativo, è risultato fondamentale il lavoro svolto in precedenza sulla modularizzazione delle strutture di visualizzazione e descrizione degli esercizi. Infatti, l'estensione è stata effettuata agendo su dei nuovi campi inseriti all'interno del file json che descrive l'esercizio. I nuovi campi inseriti descrivono:

- La possibilità di una proposizione di avere più traduzioni
- Se la frase aderisca alle regole della consecutio temporum
- Se il cum narrativo abbia una traduzione implicita e quindi la nuova frase da tradurre (e quindi il cum e il verbo devono essere visti come un unico segmento)
- Se una delle proposizioni abbia traduzioni multiple (come nel caso del cum narrativo), e le opzioni fornite di cui almeno una giusta
- L'indice della frase che viene analizzata dopo aver trovato la traduzione implicita della proposizione con il cum narrativo
- La visibilità delle frasi (che cambierà nel corso dell'esercizio)

Prendendo ad esempio la frase "*Galli, cum ad angustias perveniant, Caesar aciem instruit*", illustrata nel paragrafo precedente, i campi sopra descritti sono i seguenti:

```

"precondizioni": {
  "consecutio temporum": true,
  "traduzioni multiple": { "1": true, "2": false },
  "da tradurre": { "1": ["Galli", "cum perveniant", "ad angustias"] }
},
"traduzioniMultiple": {
  "1": {
    "Mentre i Galli giungono nelle strettoie": false,
    "Poichè i Galli giungono nelle strettoie": true,
    "Nonostante i Galli giungano nelle strettoie": false
  }
},
"fraseSuccessiva": 3,
"visible": {
  "1": true,
  "2": true,
  "3": "none"
}

```

È da precisare che gli indici sono attribuibili in questo modo: 1, alla proposizione con il cum narrativo; 2 alla proposizione principale; 3 alla finale. Il cum narrativo ha

abilitate le traduzioni multiple e da qui si possono ricavare le chiavi del dizionario di esse, con associati i valori di true o false a seconda se la frase sia o meno una giusta traduzione. Inizialmente la visibilità della finale è impostata a "none", e soltanto quando la frase del cum viene analizzata verrà impostata a true attraverso l'invocazione al reducer `changeVisibility`, aggiunto anch'esso allo slice per le subordinate.

Per la visualizzazione del tempo del verbo nella principale (storico o principale) per strutturare la sua mappa concettuale, è stato aggiunto un flag al suo dizionario nel file json con i campi dei verbi.

Similmente, è stato aggiunto un campo per definire cosa esprime il verbo della proposizione del cum (contemporaneità o anteriorità).

Le frasi con il cum narrativo aggiunte sono state due: quella appena mostrata e "*Consul cum milites urbem oppugnarent hortatus est equites*", la quale è stata inserita per far riflettere i ragazzi anche sulla contemporaneità di un verbo storico e sull'essenzialità del contesto in questo caso.

Quest'ultima frase ha permesso anche l'aggiunta di un piccolo dettaglio nel dizionario dei verbi, che è la descrizione di un verbo **deponente** (Figura 3.29), la quale si apre in una vista modale se si clicca sull'omonima porzione di testo.



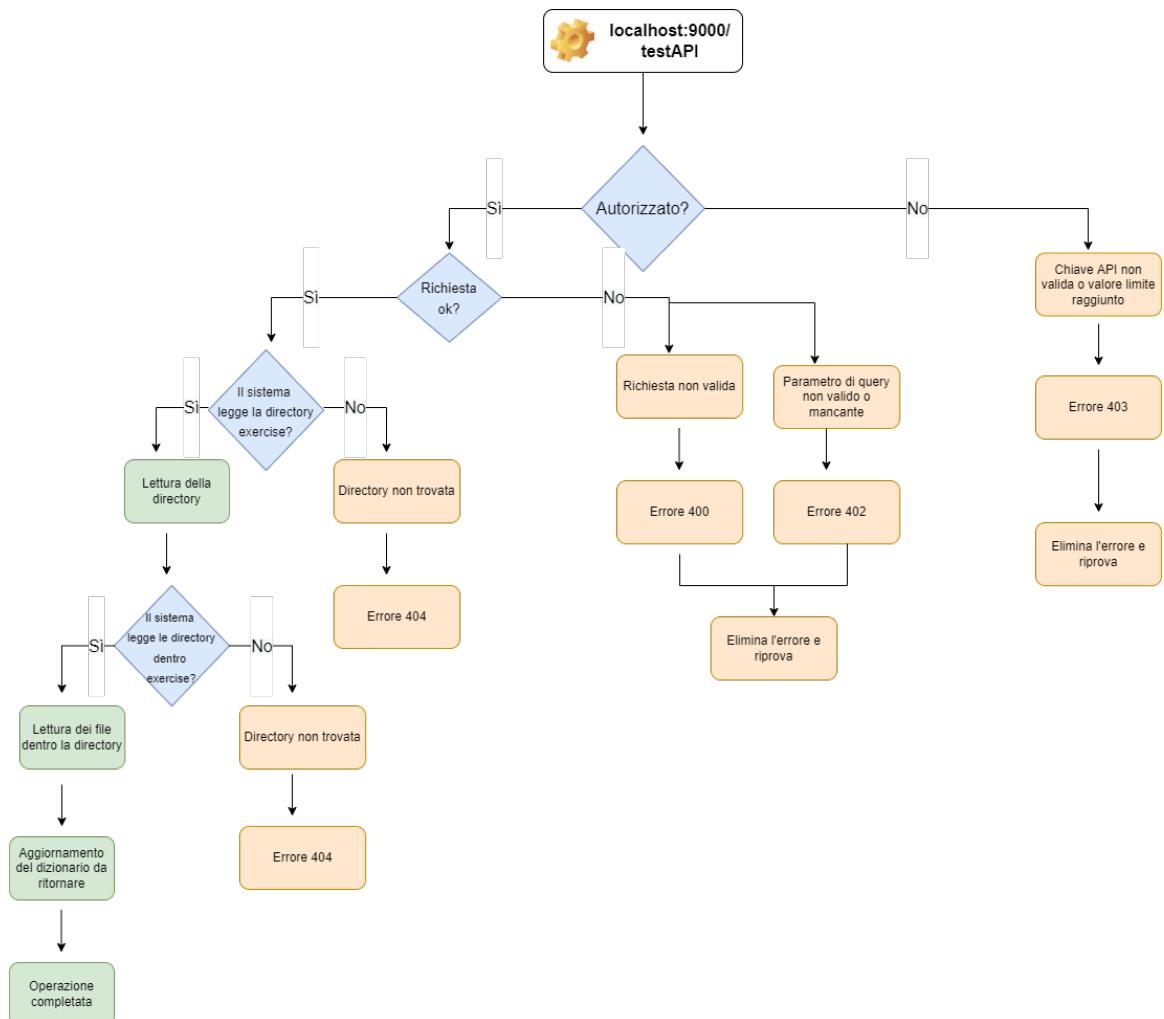
Figura 3.29. Modal per i verbi deponenti

L'introduzione di questa nuova tipologia è stato solo il primo piccolo passo verso un'estensione ulteriore del sistema, che (sperabilmente) potrà essere in grado di

contenere tanti altri nuovi aspetti che possano guidare in modo dinamico i ragazzi con DSA e normotipo nello studio del latino, sempre con la consapevolezza di avere in mano uno strumento compensativo e non sostitutivo del tradizionale insegnamento. Un'idea da implementare in futuro potrebbe essere quella di dare la possibilità al ragazzo di creare la propria mappa concettuale, in quanto risulta essere uno mezzo didattico estremamente personale e potrebbe aiutare molto gli studenti avere a disposizione qualcosa che rappresenta la propria organizzazione dei pensieri. React-Flow è strumento valido e personalizzabile, tramite il quale i grafici, impostando a *true* i props appositi, possono essere modificati dall'utente.

Questo, ovviamente, potrà essere realizzato soltanto dopo una gestione delle utenze e la costruzione di un backend.

3.5 Diagramma di flusso dell'API per la lettura dei file



Capitolo 4

Editor di esercizi

La seconda parte del lavoro di tirocinio è stata la realizzazione dell'Editor di esercizi per il sistema. L'obiettivo di quest'ultimo è stato il permettere ai futuri docenti che si interfacceranno con la piattaforma di inserirvi un nuovo esercizio, sulla base della struttura che i Components offrono e della modularizzazione del codice effettuata precedentemente. Come già detto, l'architettura è stata organizzata intorno alla scrittura di files e directories, con l'obiettivo di porre soltanto una base per un solido sistema di backend che possa in futuro contenere tutte le frasi inserite.

4.1 Components

Il software è stato realizzato tenendo in mente tutte le informazioni che devono essere necessariamente inserite all'interno del sistema per fare in modo che possa essere visualizzato un esercizio nel modo corretto. Queste, in particolare, sono: i campi del file json descrivente l'esercizio (nei quali è stata data particolare attenzione a quelli per il *cum narrativo*), i campi che descrivono i verbi nel file json dell'analisi di essi, i campi per le domande di approfondimento, helper e personalizzazione, quelli per la definizione dei distrattori e per la costruzione del dizionario.

Per ognuno di essi è stata realizzata un'apposita pagina nell'Editor, partendo da quella per la strutturazione del file json che descrive l'esercizio.

Come si può vedere nella figura **4.8**, l'Editor è costituito da un Sider posto a sinistra, nel quale è presente un Menu che mostra gli 11 step che dovrà effettuare il docente al fine di costruire un nuovo esercizio. Il Sider affiancherà il Layout dell'Editor per tutta la durata dello svolgimento delle task, mostrando mano a mano quali task sono stati completati.

In questo modo, l'utente riuscirà a:

- Fare una stima del tempo che impiegherà a svolgere le task, sapendone l'esatto numero
- Capire le task che dovrà effettuare, leggendo i nomi dei *Menu.Item*

È importante anche sottolineare che l'utente potrà spostarsi liberamente per le sezioni del Menu, ma solo se esse risultano pronte ad essere visualizzate. Infatti, ogni sezione dalla prima in poi funzionerà con le informazioni che ricaverà da quelle precedenti. Quindi, l'utente potrà spostarsi dall'alto verso il basso solo se vi è passato precedentemente. Potrà spostarsi dal basso verso l'alto a piacimento, per modificare le informazioni che desidera. Nella figura 4.4 viene mostrato l'errore che si verificherà nel momento in cui il docente prova a passare alla task successiva (in questo caso, cliccando su *Inserisci le proposizioni*) senza aver completato la precedente (ovvero *Inserisci la frase*).

Per guidare l'utente nell'Editor, è stato inserito il personaggio di *Cicero* [?]: egli ha l'obiettivo di far capire maggiormente (ed in modo più simpatico) al docente il suo ruolo, ovvero quello di costruire passo passo l'esercizio che i propri studenti dovranno svolgere in futuro. Come è possibile vedere nella Figura 4.1, infatti, la pagina iniziale dell'Editor è introdotta da un Modal nel quale il personaggio di Cicero si presenta, e spiega al docente cosa andrà a realizzare grazie all'Editor. Se l'utente cliccherà su "Inizia", potrà visualizzare la prima pagina del software e costruire l'esercizio; altrimenti, potrà tornare all'Eserciziario.

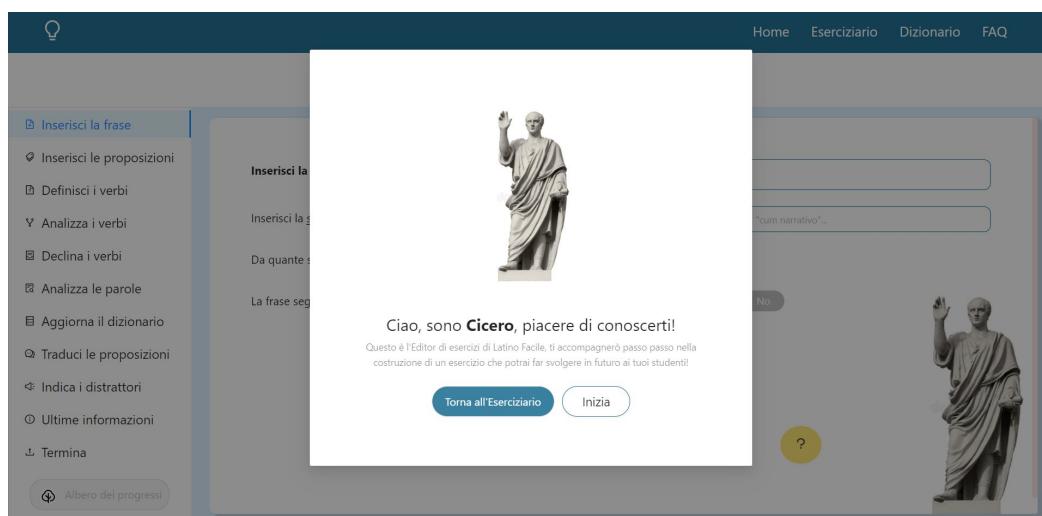


Figura 4.1. Modal iniziale che presenta Cicero

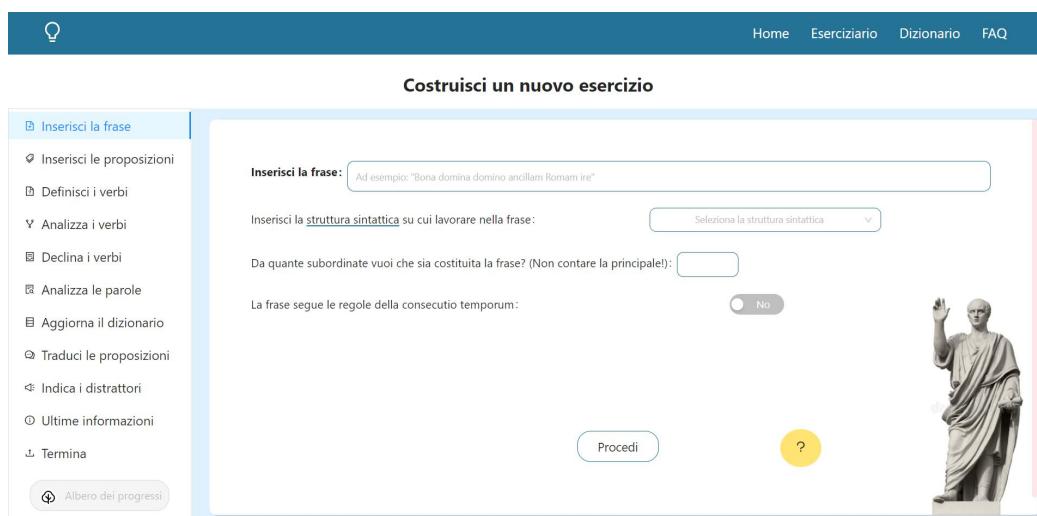


Figura 4.2. Pagina iniziale dell'Editor



Figura 4.3. Select per la struttura sintattica con Input per le nuove strutture

Inserisci la frase Per prima cosa, viene chiesto al docente di inserire la frase per intero. L'inserimento della frase e delle informazioni successive è stato realizzato con l'utilizzo del Component *Form* di Ant Design. Tutti i campi sono obbligatori, quindi dovranno essere compilati tutti affinchè il click sul bottone *Procedi* possa portare l'utente alla prossima task. In questa pagina è richiesto di inserire informazioni circa: la frase completa latina, la struttura sintattica della frase, il numero delle subordinate di cui è composta e se aderisce alle regole della *consecutio temporum* (dettaglio inserito con l'introduzione del *cum narrativo*). Le strutture sintattiche da inserire sono passate come props al Component, e prese precedentemente dal res del fetch dell'API nell'Eserciziario. Esse saranno disponibili all'utente nella Select

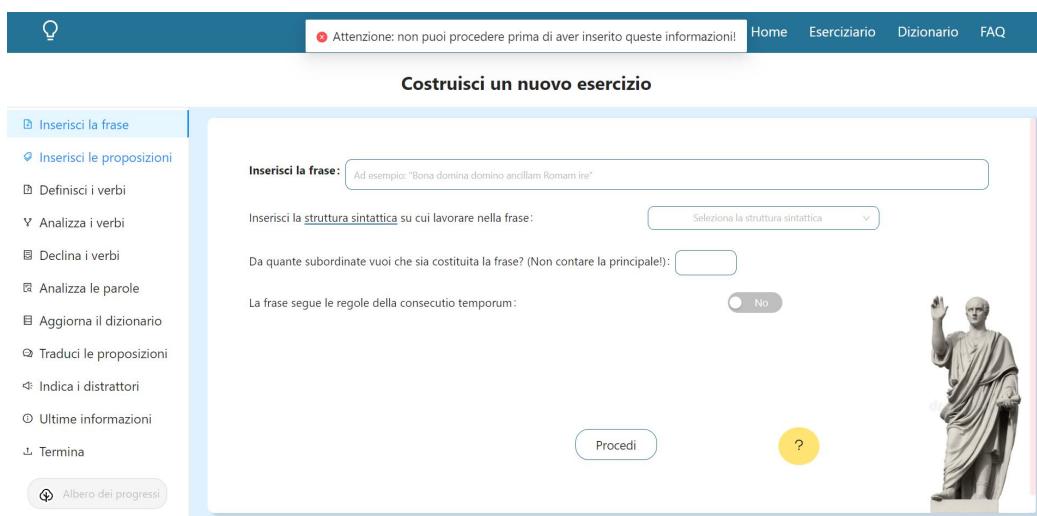


Figura 4.4. Warning visualizzato se ci si sposta ad una sezione non ancora pronta

apposita, tramite la quale potrà anche inserire una nuova struttura, se vuole. Se il docente non sa quale struttura inserire, potrà ricercare un aiuto nel Tooltip che spunterà se passerà il cursore sulla parola "struttura sintattica", sottolineata (Figura 4.5). Il numero di subordinate che l'utente può inserire è scelto in un range tra 1 e 10: questa scelta è stata motivata dal fatto che, essendo il sistema uno strumento compensativo e mirato (almeno per il momento) allo studio del latino del primo anno, si è pensato che inserire più di 10 subordinate nella stessa frase sarà pressocchè impossibile.

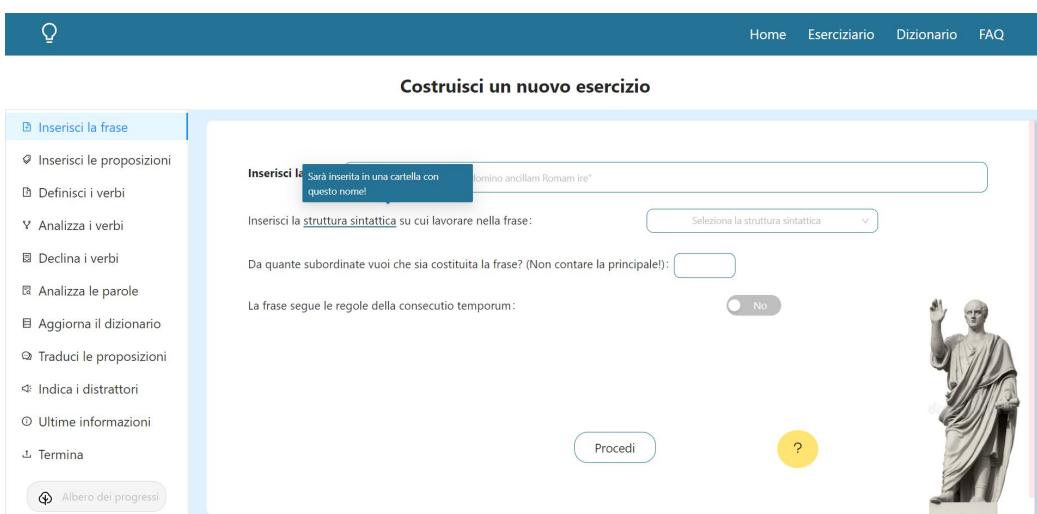


Figura 4.5. Tooltip per la scelta della struttura sintattica

In basso a destra, l'utente potrà visualizzare una guida e le istruzioni per lo svolgimento della task. Cliccando *Guarda un esempio*, si aprirà un Modal che mostrerà una simulazione della task con la frase "*Bona domina domino ancillam Romam ire*", insieme alle spiegazioni, passo per passo. Per muoversi nella pagina il docente potrà cliccare sulle frecce al centro e potrà cliccare su *Capito* ogni volta che vorrà chiudere la vista modale. L'immagine di *Cicero*¹, d'ora in avanti, sarà presente al lato di tutte le pagine dell'Editor.

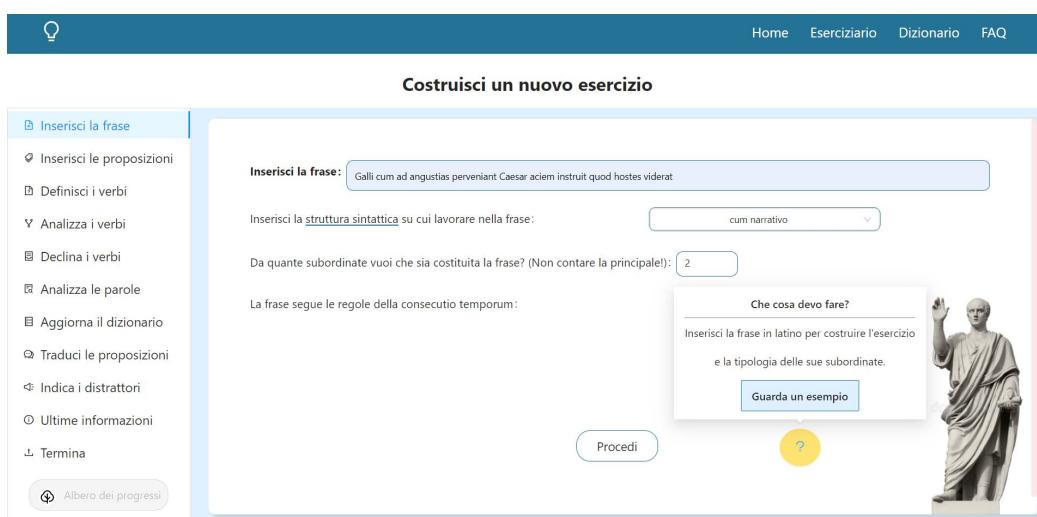


Figura 4.6. Tooltip di help per la prima task

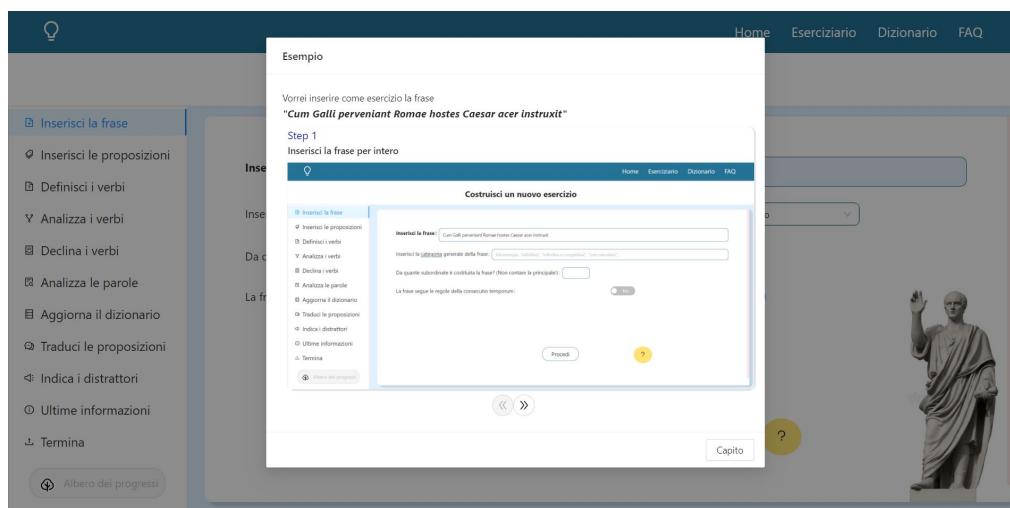


Figura 4.7. Modal di esempio per la prima task

¹Fonte dell'immagine: <https://it.dreamstime.com/statua-di-cicerone-uno-statista-ro-mano-dell-avvocato-oratore-e-del-filosofo-isolato-su-bianco-image139542965>

Inserisci le proposizioni Il secondo step riguarda l'identificazione e la distinzione delle subordinate nella frase e della proposizione principale. In alto al centro l'utente potrà leggere la frase inserita, in modo tale da poter inserire le frasi separatamente senza dimenticarsi le parole. Infatti, è molto importante che l'utente non dimentichi nessuna parola e che inserisca quelle nella frase originale: se non è così, al *Procedi*, verrà visualizzato in alto un warning che chiede all'utente di rileggere la frase e ricontrollare le parole inserite (Figura 4.9).

The screenshot shows the 'Costruisci un nuovo esercizio' (Create a new exercise) page. On the left, a sidebar lists various options: Inserisci la frase, **Inserisci le proposizioni** (which is selected), Definisci i verbi, Analizza i verbi, Declina i verbi, Analizza le parole, Aggiorna il dizionario, Traduci le proposizioni, Indica i distrattori, Ultime informazioni, Termina, and Albero dei progressi. The main area displays the sentence "Galli cum ad angustias perveniant Caesar aciem instruit quod hostes viderat". Below it, there are three input fields: "Proposizione principale:" with "Caesar aciem instruit" and a note "Ad esempio: 'Bona domina domino dicit'...", "Subordinata" with "cum narrativo" and "Galli cum ad angustias perveniant", and another "Subordinata" with "causale" and "quod hostes". To the right is a statue of a Roman figure.

Figura 4.8. Seconda pagina dell'Editor

This screenshot is similar to Figura 4.8, showing the 'Costruisci un nuovo esercizio' page. However, a red warning message at the top left states: "Attenzione: hai inserito o non inserito parole non presenti nella frase originale: ricontrolla prima di proseguire". The main area shows the same sentence and proposition fields. A yellow warning icon with a question mark is visible in the bottom right corner of the main input area.

Figura 4.9. Warning visualizzato quando non si inseriscono tutte le parole

Ogni proposizione verrà inserita in un apposito *Input*, ed è richiesto all'utente di distinguere le subordinate nome per nome per crearne dei campi separati nella

struttura dati e per processare le informazioni per controllare se le tipologie inserite già esistono nel sistema o se, al contrario, sono nuove. Come per la prima pagina e per le successive, l'insegnante potrà in qualsiasi momento far uso del bottone di help in basso a destra per essere maggiormente guidato nell'esecuzione della task.

Definisci i verbi La terza pagina riguarda la definizione e distinzione dei verbi nella frase. La struttura di base riecheggia da quella già presente in quella dell'Esercizio, quando viene chiesto allo studente di individuare i verbi nella frase e di cliccarvi sopra. La task è analoga: viene richiesto al docente di cliccare sopra ai verbi della frase e, se presenti, anche sugli avverbi.

Figura 4.10. Terza pagina dell'Editor

Da questa pagina in poi, l'utente potrà in qualsiasi momento rileggere la frase inserita, distinta per subordinate e colore di esse (se presente nel sistema, grigio altrimenti). In questo modo, il docente viene maggiormente guidato nel momento in cui ha bisogno di ricordarsi dei particolari rileggendo la frase (ad esempio la funzione dei verbi per la consecutio temporum o la traduzione delle subordinate, come si vedrà più avanti). Il bottone che lo permette è posizionato in basso a sinistra della pagina e cita: *"Rileggi la frase inserita"*. Cliccandovi sopra verrà visualizzato un *Tooltip* con le informazioni sopraccitate (Figura 4.11). Come si può notare, accanto al titolo *"La frase che hai inserito è"*, è presente l'icona QuestionOutlined: se l'utente vi cliccherà sopra, verrà portato alla pagina delle FAQ, alla domanda "Perchè le subordinate sono colorate?", la cui risposta spiega la grande importanza che ha l'uso dei colori nella distinzione delle subordinate e parole importanti per i DSA, come si può vedere nella Figura 4.12.

Figura 4.11. Tooltip per la lettura della frase per intero

Figura 4.12. Pagina delle FAQ, Panel sui colori delle subordinate

La distinzione corretta dei verbi è molto importante, in quanto nella fase successiva verrà richiesta la loro analisi, funzionale per l'analogia task nell'Esercizio. È per questo che è stato inserito un controllo per cui, sapendo che ogni proposizione è costituita da un suo verbo e che quindi ci saranno n verbi per n proposizioni presenti, se si cliccherà su meno di n parole il sistema avvertirà l'utente di selezionare tutti i verbi presenti.

Se nella frase non è presente alcun avverbio, si potrà procedere tranquillamente alla task successiva. La prima volta che il bottone "Procedi" viene cliccato, spunterà un

warning che comunica all'utente che non ha selezionato alcun avverbio (nel caso si sia dimenticato di farlo), ma al secondo click il sistema passerà alla pagina successiva, con o senza avverbi selezionati.



Figura 4.13. Warning visualizzato la prima volta se non si cliccano avverbi

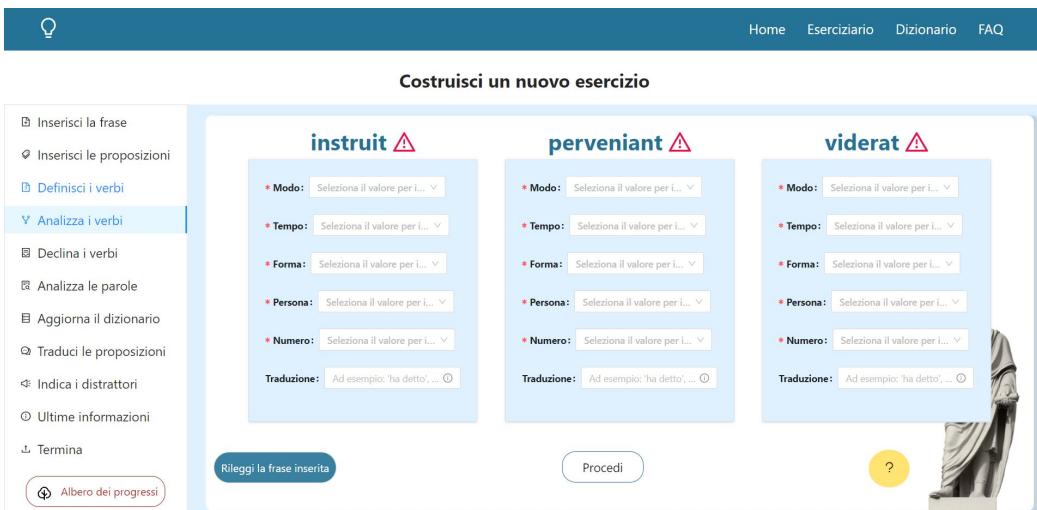


Figura 4.14. Quarta pagina dell'Editor

Analizza i verbi La quarta pagina riguarda l'analisi dei verbi inseriti. Essa è funzionale per la parte di Esercizio in cui viene chiesto allo studente di selezionare i valori di modo, tempo, forma, persona e numero dei verbi della frase. Inoltre, viene richiesta anche la traduzione che il verbo avrà nella frase, in quanto inizieranno ad

essere aggiornati i campi di essa nel dizionario, maggiormente spiegato nella sezione 4.3. Se il verbo è già presente nel sistema, l'utente visualizzerà un'icona di Warning, che gli chiederà se desidera compilare gli spazi con le informazioni già presenti nel file json dei verbi, in modo da non appesantirlo con un lavoro inutile di inserimento di informazioni analoghe nelle form (Figura 4.15).

L'autofill non comprenderà l'*Input* della traduzione, in quanto essa può variare in base a molti fattori tra cui: l'interpretazione che l'insegnante vuole dare al verbo e la funzione di esso (implicita o esplicita, che richiedono traduzioni diverse). Inoltre, il file json dei verbi non prevede un campo per la traduzione proprio perché è libera e personale per ogni frase.

Inoltre, da questa pagina in poi, sarà disponibile per la visualizzazione l'albero dei progressi della frase, che si costruirà man mano che vengono analizzate tutte le parole di essa (Figura 4.17).

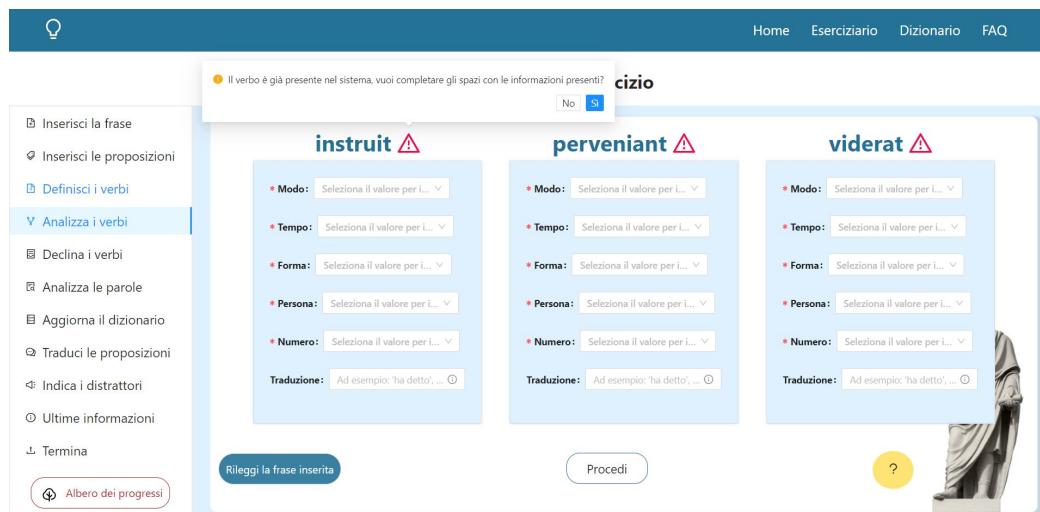


Figura 4.15. Warning per l'autofill dei campi dei verbi già presenti



Figura 4.16. Albero dei progressi visibile nella fase di analisi dei verbi

Declina i verbi La quinta pagina dell’Editor è dedicata alla declinazione dei verbi e all’inserimento delle informazioni funzionali per la costruzione del dizionario di essi e (se la frase è della struttura sintattica del *cum narrativo*) dei campi relativi alla contemporaneità e anteriorità del verbo della subordinata e della consecutio temporum del verbo della principale. Come nella pagina precedente, viene data la possibilità all’utente di realizzare la compilazione automatica dei campi dei verbi già presenti nei file json dell’applicazione.

Figura 4.17. Quinta pagina dell’Editor

L’autofill non riguarderà i campi relativi al *cum narrativo* in quanto non sempre i verbi presenti sono stati configurati per essere presenti in una frase con essi (infatti, si

potrebbe utilizzare un verbo già presente nel sistema ma configurato per un esercizio senza il cum narrativo).

La pagina è stata costruita in base alla presenza o meno di una frase con il cum narrativo ed in particolare, secondo le regole di esso, sono state inserite delle strutture di Select diverse a seconda di quale tipologia proposizione contiene il verbo in questione. Quindi se il verbo appartiene alla proposizione principale verrà chiesto il tempo di esso in base alla consecutio temporum, e se invece appartiene alla proposizione del cum narrativo verrà richiesto cosa esprime il verbo (contemporaneità o anteriorità).

The screenshot shows the 'Declina i verbi' section of the application. On the left, a sidebar lists various options: Inserisci la frase, Inserisci le proposizioni, Definisci i verbi, Analizza i verbi, Declina i verbi (highlighted in blue), Analizza le parole, Aggiorna il dizionario, Traduci le proposizioni, Indica i distrattori, Ultime informazioni, and Termina. Below this is a button for 'Albero dei progressi'. The main area is titled 'Costruisci un nuovo esercizio' and contains three boxes for different verb forms: 'instruct' (Paradigma: instruo, instruis, instruxi, ir; Conjugazione: 3; Funzione: Transitivo; Significati: costruire, attrezzare, dis; Consecutio temporum: Tempo principale; Esprime: Selezione il valore per il verbo; Deponente: No), 'pervenant' (Paradigma: pervenio, pervenis, perva; Conjugazione: 4; Funzione: Intransitivo; Significati: giungere, arrivare...; Esprime: Selezione il valore per il verbo; Deponente: No), and 'viderat' (Paradigma: Ad esempio: "sum, es, f..."; Conjugazione: Selezione il valore per il verbo; Funzione: Selezione il valore per il verbo; Significati: Ad esempio: "essere, es..."; Deponente: No). At the bottom are buttons for 'Rileggi la frase inserita' and 'Procedi'.

Figura 4.18. Campi per il cum narrativo nella pagina Declina i verbi

This screenshot is similar to Figure 4.18 but includes a red error message at the top left: 'Devi compilare tutti i campi!' (You must fill all fields!). The rest of the interface is identical, showing the three verb forms with their respective configuration fields and a sidebar with various menu items.

Figura 4.19. Errore se si tenta di procedere senza aver compilato tutti i campi

È bene precisare che, sia nella pagina dell'analisi dei verbi sia in questa, se si prova a procedere non avendo compilato qualche campo, il sistema ritornerà un messaggio di errore che notifica l'utente riguardo a questo problema.

Figura 4.20. Sesta pagina dell'Editor

Analizza le parole La sesta task è l'analisi delle parole. Ad ogni parola, come nell'analisi dei verbi, è assegnata una *Card* nella quale l'utente potrà inserire le informazioni negli appositi form. Non è previsto il servizio di autofocus in questo caso perché le parole latine non corrispondono ad un unico caso e ad un'unica funzione relativa al proprio caso (quindi, ad esempio, la parola *Caesar* potrebbe essere sia nominativo sia vocativo). In un nuovo lavoro di tirocinio, potrebbe essere utile sviluppare un dizionario con tutti i casi relativi alla parola in questione e chiedere all'utente a quali di essi corrisponda, sempre per rendere più facile il suo inserimento da parte del docente.

La pagina è divisa in sottopagine corrispondenti al numero di subordinate presenti nella frase ed in ognuna di esse vi saranno tante Card quante sono le parole della frase (esclusi i verbi, analizzati precedentemente). Per passare all'analisi delle parole delle altre proposizioni, basterà cliccare sulle icone in basso al centro che raffigurano le frecce per spostarsi da una sottopagina all'altra.

Per la realizzazione di questa pagina sono state prese in considerazione tutte le strutture grammaticali che potrebbero essere presenti in una frase latina: i sostantivi, i pronomi, gli aggettivi, le congiunzioni subordinanti e le preposizioni (gli avverbi non sono stati considerati poiché la loro selezione è avvenuta nella terza pagina dell'Editor). Per ogni parola, l'utente potrà indicare se essa rappresenti un sostantivo

o un aggettivo oppure "Altro", ovvero una congiunzione o una preposizione. A seconda di cosa selezioni il docente nel ToggleButton, verrà visualizzato un form diverso. In particolare, per la parola *cum* è stato inserito un Tooltip informativo che rassicura l'utente che vorrà tradurre il *cum* narrativo in maniera implicita (e quindi non fornire una traduzione esplicita per la congiunzione), come si può vedere nella Figura 4.21. Infatti, la traduzione dovrà obbligatoriamente essere inserita, ma successivamente si potrà indicare al sistema di volergli assegnare una traduzione implicita (e quindi di volerlo tradurre in gerundio insieme al verbo).

Inoltre, come *suffix* degli Input per i sostantivi/aggettivi è stato inserito un Tooltip informativo che notifica l'utente di inserire anche gli articoli, se presenti nella traduzione prima della parola. È molto importante che vengano inseriti tutti gli elementi giusti per la traduzione della frase, funzionali per il task effettivo della traduzione delle proposizioni.

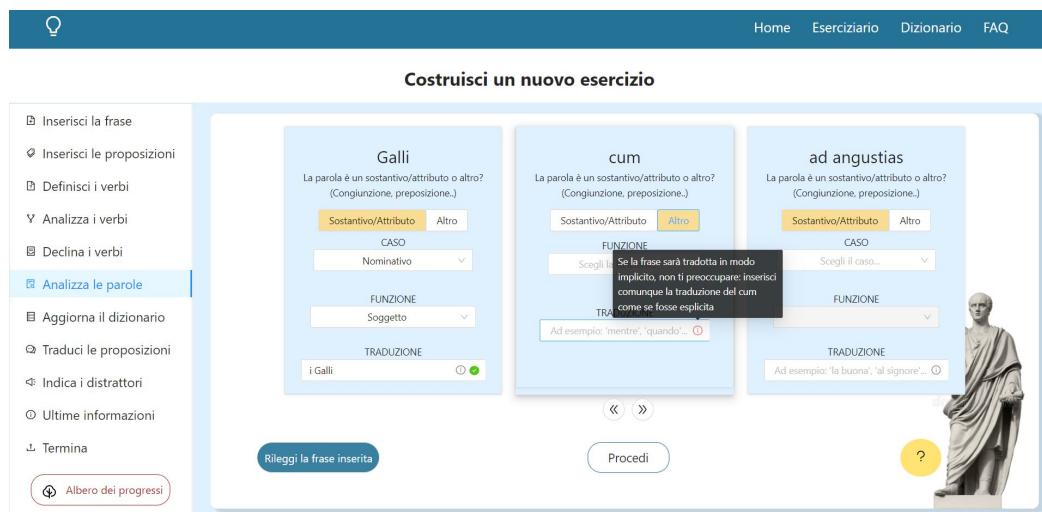


Figura 4.21. Tooltip informativo per la parola *cum*

Inoltre, è stato gestito il caso in cui una parola sia preceduta da una preposizione quale ad esempio: "ad", "in", "propter", ecc in quanto il sistema valuterà questa, insieme alla sua parola successiva, come una stringa unica. Questa scelta è stata motivata dal fatto che tutti gli esercizi presenti nel sistema sono stati costruiti in modo tale da far analizzare allo studente il complemento che si ricava dall'insieme di preposizione e parola (in accusativo, ablativo o altri casi) piuttosto che le parole singolarmente, come il sintagma "ad angustias", nell'esempio.

Al "Procedi", come sempre, il sistema controllerà che l'utente abbia inserito tutte le informazioni per tutte le parole della frase e notificherà l'utente se manca qualcosa.

Figura 4.22. Esempio di dizionario con congiunzione, sostantivo e aggettivo

Aggiorna il dizionario La settima task riguarda l'aggiornamento del dizionario con le parole presenti nella frase inserita che esso non contiene. Questa fase è molto importante per poter visualizzare nell'Esercizio le informazioni inserite per ogni parola attivando il ToggleButton "Aiuto nascosto".

Il dizionario presenta strutture e campi diversi a seconda della funzione grammaticale della parola. Infatti, a seconda che la parola sia un sostantivo, una congiunzione o un attributo, verrà visualizzata una card diversa con campi di form diversi da compilare. Come è facile dedurre, le informazioni ricavate dalla fase precedente sulle parole in questione sono risultate cruciali.

Figura 4.23. Notifica visualizzata quando il dizionario contiene già tutte le parole

Come è possibile vedere nella Figura 4.22, creata appositamente per mostrare cosa si visualizza con parole non presenti nel dizionario, le Card visualizzate richiedono informazioni diverse a seconda se la parola sia un attributo (*sua*), un sostantivo (*patriam*) o una congiunzione (*quia*). Tutti i campi nelle Form sono obbligatori.

Non è detto che una frase contenga parole non presenti nel file json del dizionario, ed in questo caso verrà visualizzata una schermata che informerà l'utente di questo fatto e lo inviterà a procedere alla prossima task (Figura 4.23).

Da questa pagina in poi, l'utente potrà visualizzare anche le parole inserite nell'albero dei progressi, e quindi l'albero completo che il ragazzo utilizzerà nel suo Esercizio.

Traduci le proposizioni L'ottava task riguarda la traduzione delle proposizioni e l'ordine giusto di esse all'interno della frase. Infatti, l'ordine in cui l'utente deciderà di porle, sarà quello con cui verranno analizzate all'interno dell'Esercizio.

Per la realizzazione di questa pagina ed in particolare degli elementi che possono essere selezionati e trascinati in un'altra posizione, è stata utilizzata la libreria *react-beautiful-dnd*², la quale permette di creare facilmente effetti di "drag and drop" e presentarli nel modo più naturale possibile per l'utente.



Figura 4.24. Ottava pagina dell'Editor

Per la traduzione delle subordinate, l'utente potrà visualizzare un Tooltip, il quale lo notifica di inserire anche le congiunzioni che precedono la frase. Questo aspetto è molto importante perché, al "Procedi", il sistema collezionerà tutte le parole e le metterà nell'ordine in cui poi verrà costruita la fase di traduzione nell'Esercizio (Figura 4.29). Infatti, avendo in precedenza l'utente inserito la traduzione delle singole parole, ora il compito del sistema è unirle tutte per creare una frase e controllare che esse coincidano con quelle inserite nella proposizione completa inserita in questa pagina. Quindi, ad esempio, se l'utente avrà tradotto il sintagma "ad angustias" con "strettoie", e nel form della traduzione scrive "nelle strettoie", il sistema lo notificherà dell'accaduto e lo inviterà a cambiare la traduzione.

²<https://www.npmjs.com/package/react-beautiful-dnd>

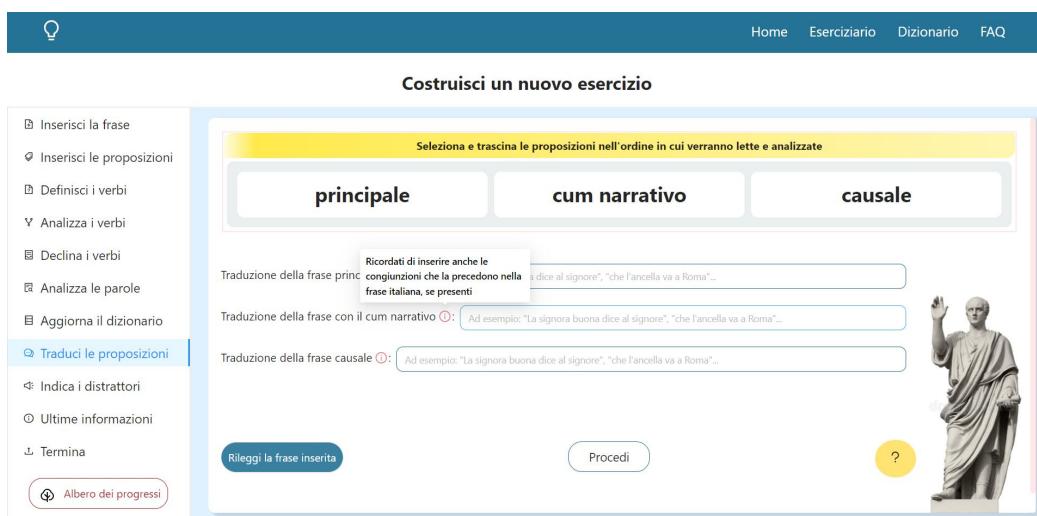


Figura 4.25. Tooltip visualizzato per guidare l'utente nella traduzione delle subordinate

Se nella frase è presente una preposizione con il cum narrativo ed essa non è stata inserita come frase finale da analizzare, al "Procedi" verrà visualizzato un Modal che chiederà all'utente se la traduzione di essa sia implicita o esplicita e, nel caso sia implicita, se vuole fare in modo che l'Esercizio fornisca un'aiuto allo studente per ragionare sul contesto (ovvero in cui verrà data la possibilità di dare la traduzione esplicita della frase).

In particolare, nel Modal sono presenti due Switch, il secondo dei quali verrà mostrato se lo stato del primo sarà attivato. Nel caso, infatti, la traduzione della frase sia esplicita e/o l'utente non voglia dare contesto alla frase, potrà cliccare sul bottone "Continua" in basso a destra, il quale chiuderà la vista modale.

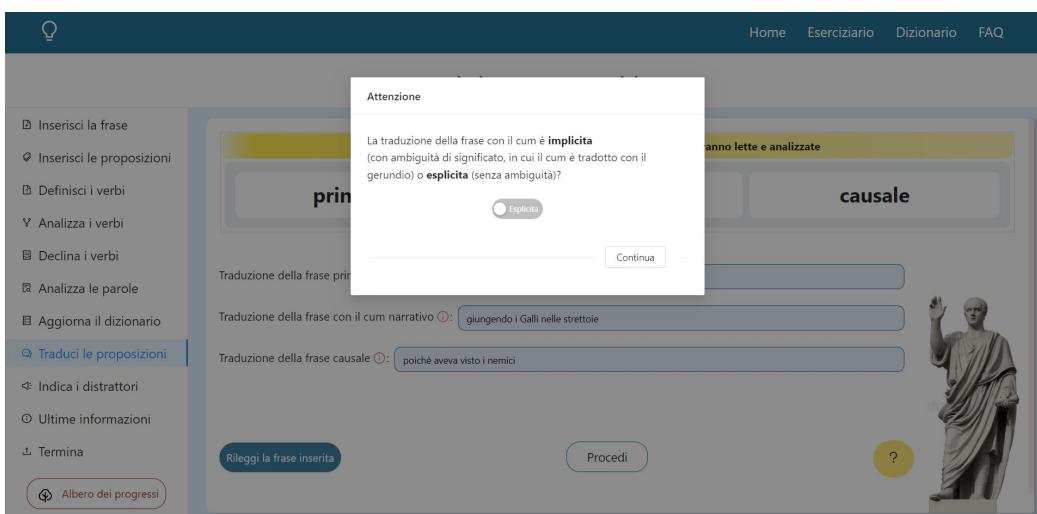


Figura 4.26. Modal per la traduzione del cum narrativo in maniera esplicita

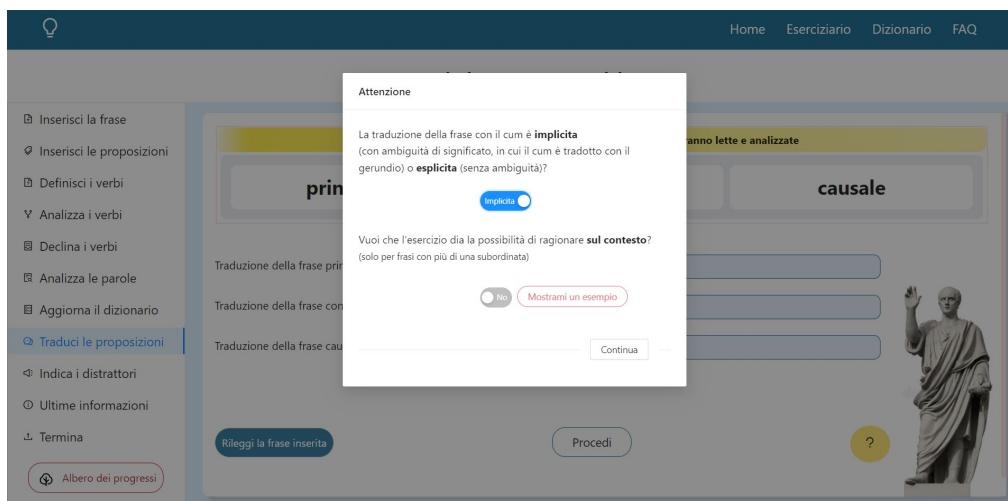


Figura 4.27. Modal per la traduzione del cum narrativo in maniera esplicita (2)

Se il docente ha bisogno di capire in modo più chiaro cosa si intenda per "ragionare sul contesto" in termini di Esercizio, potrà cliccare su *Mostrami un esempio*, che aprirà un Drawer che mostrerà un esempio di frase con il cum narrativo che fa ragionare sul contesto.

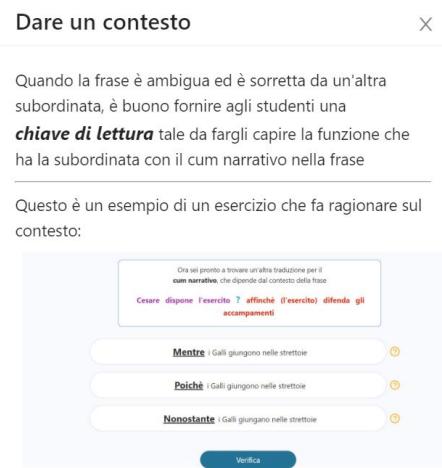


Figura 4.28. Drawer con un esempio di un Esercizio con il contesto

Indica i distrattori La penultima pagina prima di terminare la costruzione dell'esercizio è dedicata alla definizione dei distrattori per la traduzione delle parole singole. È importante precisare che, durante l'intera implementazione dell'Editor, si è sempre pensato che l'inserimento di informazioni libere da parte del docente è frutto delle sue strategie, e quindi non viene dato alcun limite o suggerimento per inserire i distrattori delle parole della frase.

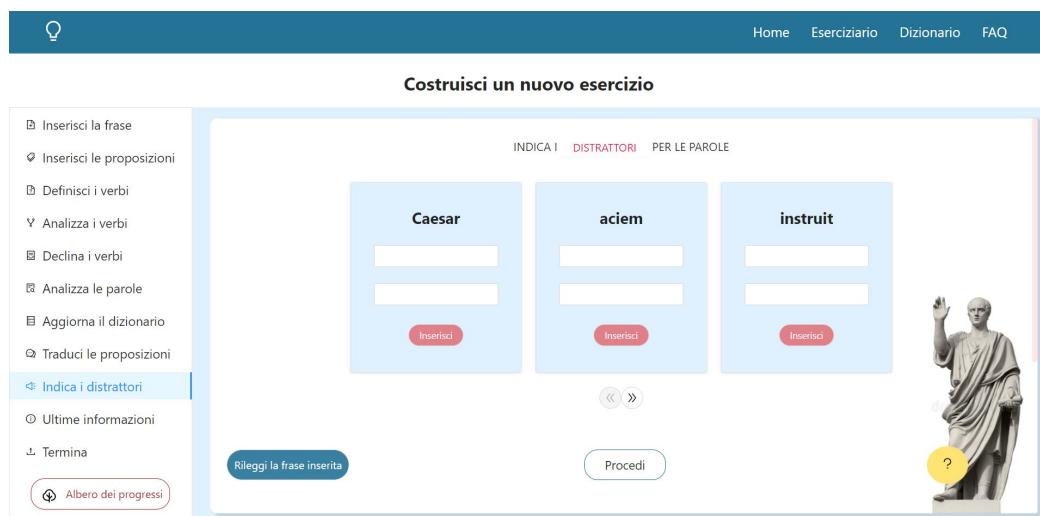


Figura 4.29. Nona pagina dell'Editor

Un possibile lavoro per una tesi seguente potrebbe essere quello di costruire un sistema che genera in automatico i distrattori in base agli errori più comuni negli studenti.

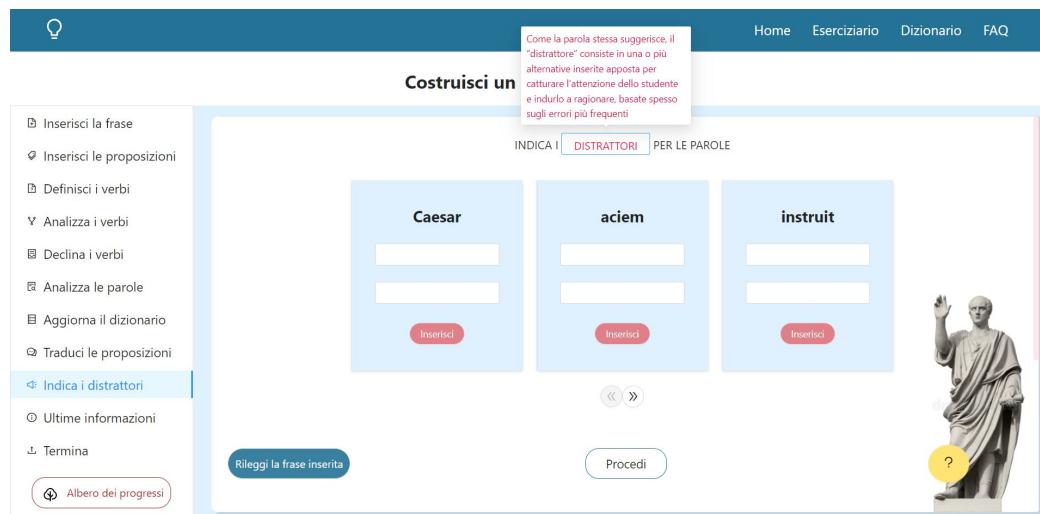


Figura 4.30. Tooltip che spiega i distrattori nella frase

Se l'utente ne avesse il bisogno, potrà passare il cursore sulla parola DISTRATTORI, e visualizzare una piccola spiegazione di cosa sono e di come vengono scelti di solito (Figura 4.30).

Ultime informazioni L'ultima pagina dell'Editor prima della fase finale è dedicata all'inserimento delle informazioni quali le domande di approfondimento, il colore delle subordinate (se ne sono state inserite di nuove), l'inserimento delle regole delle subordinate non presenti nel sistema e, se è stata selezionata l'opzione per dare una traduzione esplicita alla frase del cum narrativo, le opzioni per essa e la risposta corretta.



Figura 4.31. Decima pagina dell'Editor: prima sottopagina

La prima sottopagina è dedicata all'inserimento delle **domande di approfondimento**. Essa è stata realizzata con il componente *Step* di ant-design, il quale è stato utilizzato per differenziare e rendere chiare le domande da inserire in ogni fase. Vi saranno n steps per le n proposizioni presenti nella frase (inclusa quella principale). Se il docente vorrà, potrà visualizzare come il sistema presenta le domande di approfondimento nell'esercizio nel bottone apposito del Divider, che cita "Guarda un esempio in un Esercizio" (Figura 4.32).

Per passare da sottopagina a sottopagina l'utente potrà spostarsi tramite i numeri in basso al centro, che indicano il *Pagination* di essa.

La seconda sottopagina è dedicata all'**assegnazione dei colori** alle subordinate non presenti nel sistema. Il docente potrà scegliere esso da una paletta di sei colori di base (dai quali, volendo, può scegliere liberamente il proprio colore selezionandolo dalla paletta universale che verrà mostrata se si clicca su uno di questi), ed assegnarlo

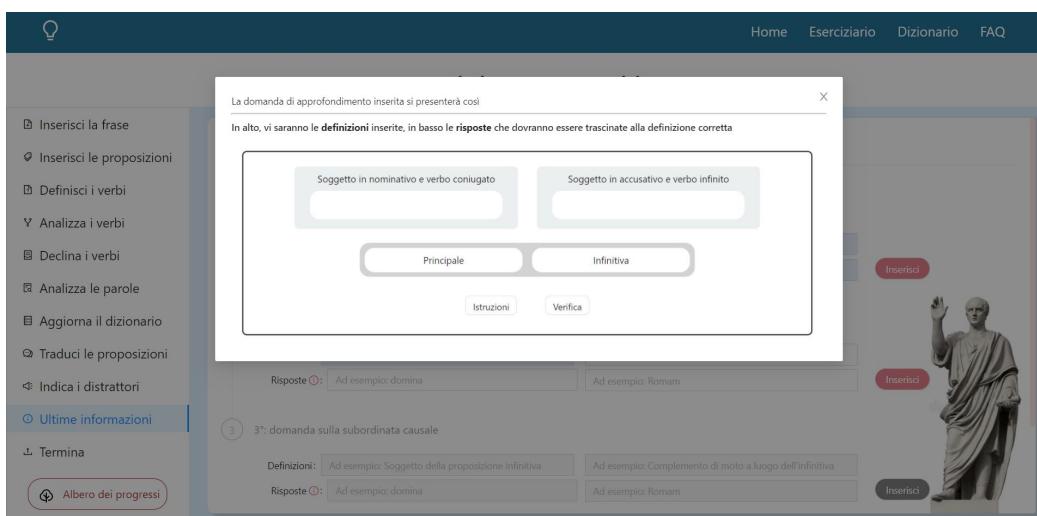


Figura 4.32. Modal per le domande di approfondimento

alla subordinata in questione. Le proposizioni già presenti nel sistema ed il loro colore saranno visibili in una Card sottostante. Appena l’utente sceglierà il colore della subordinata, anch’essa verrà inserita in una Card, per mostrare all’utente che l’operazione è andata a buon fine (Figura 4.37, nella quale, per illustrare, è stata presa la *causale* come esempio nonostante fosse già presente nel sistema).

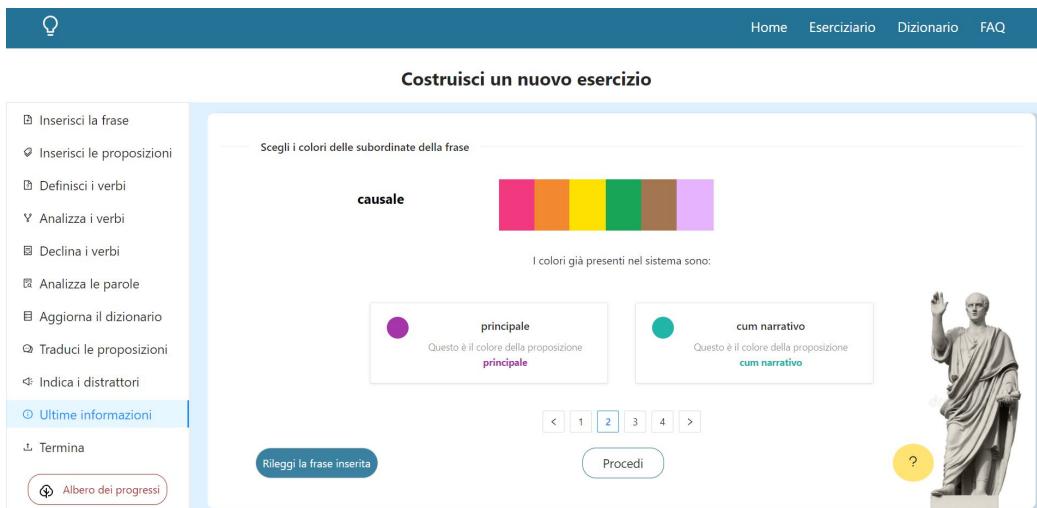


Figura 4.33. Sottopagina per la scelta dei colori delle nuove subordinate

La terza sottopagina riguarda l’inserimento delle **regole** delle subordinate non presenti nel sistema. Questa task è funzionale alla creazione dell’helper che lo studente potrà visualizzare nella prima task dell’Esercizio, in cui deve individuare i verbi. L’helper lo aiuterà a distinguere le subordinate che compongono la frase e le loro

The screenshot shows a sidebar with various options like 'Inserisci la frase', 'Inserisci le proposizioni', etc. The main area is titled 'Costruisci un nuovo esercizio' and has a sub-section 'Scegli i colori delle subordinate della frase'. It displays three cards: 'principale' (purple), 'cum narrativo' (teal), and 'causale' (pink). Below the cards are navigation buttons (< 1 2 3 4 >) and a 'Procedi' button. To the right is a statue of a man in a toga.

Figura 4.34. Sottopagina che visualizza le card dei colori di tutte le subordinate

strutture sintattiche. Come nella pagina che riguarda i colori, anch’essa presenta delle Card sottostanti che indicano le regole delle proposizioni nella frase che sono presenti nel sistema. Appena il docente indicherà la struttura sintattica della nuova tipologia di proposizione, verrà inserita nel sistema e ne verrà creata la Card apposita.

The screenshot shows the same sidebar and main title as Figure 4.34. The main area now has a sub-section 'Inserisci le regole delle proposizioni' with a 'Guarda un esempio in un Esercizio' link. It shows a card for 'causale' with fields for 'Soggetto' and 'Verbo'. Below are two cards: 'Definizione principale' (purple) and 'Definizione cum narrativo' (teal). Navigation and 'Procedi' buttons are present, along with a statue of a man in a toga.

Figura 4.35. Terza sottopagina: inserimento delle regole delle nuove proposizioni

La quarta sottopagina sarà presente soltanto nel caso in cui:

- La struttura sintattica della frase è il *cum narrativo*
- Esistono almeno tre proposizioni ed il *cum narrativo* non è la proposizione analizzata alla fine

- Il docente ha precedentemente attivato lo *Switch* della traduzione esplicita con contesto

Soltanto se sono rispettati questi tre casi, l'utente potrà raggiungere la quarta sottopagina, dedicata all'inserimento delle opzioni per la **traduzione esplicita** della proposizione con il cum narrativo. L'utente potrà, secondo le sue strategie didattiche, inserire le opzioni che vorrà (sempre potendo consultare l'esempio di un Esercizio cliccando sul bottone apposito), e successivamente dovrà selezionare la risposta corretta tra i tre *RadioButtons* presenti in basso al centro.

Figura 4.36. Quarta sottopagina per la traduzione esplicita della subordinata

Figura 4.37. Modal di warning che si visualizza la prima volta che si seleziona la prima domanda come risposta corretta

Se selezionerà la prima come risposta corretta, al submit delle informazioni il sistema lo notificherà sull'importanza di non selezionare sempre la prima risposta come risposta corretta, in quanto gli studenti potrebbero "abituarsi" all'idea di che sia sempre questa, senza poi effettivamente ragionarvi sopra.

Al "Procedi", l'utente verrà portato alla pagina finale dell'Editor.

4.2 Fase finale

L'undicesima e ultima parte dell'Editor è stata strutturata in modo tale da mostrare all'utente un riassunto delle informazioni principali che ha inserito per la frase. In particolare, queste sono: la frase completa, la struttura sintattica di essa, la traduzione ed i colori assegnati alle nuove subordinate (se non presenti tutte nel sistema). In questo modo, l'utente potrà controllarle di nuovo e, se vorrà, tornare indietro per modificare quello che vuole cambiare.

Appena sarà pronto ad inserire la nuova frase nel sistema, potrà cliccare su "*Termina e inserisci l'esercizio*", e l'evento di click innescherà il fetch della REST API (nella Figura 4.38 chiamata APIdownload) che si occupa di aggiornare il contenuto di tutti i file json per la visualizzazione delle informazioni corrette in un Esercizio.

```
callAPI() {
  fetch("http://localhost:9000/APIdownload", {
    method: "POST",
    body: JSON.stringify({
      file: this.props.jsonFile,
      tipologie: this.props.tipologie,
      frase: this.props.frase,
      colors: this.props.colors,
      verbi: this.props.analisiVerbi,
      corrispondenze: this.props.corrispondenze,
      dizionario: this.props.dizionario,
      distrattori: this.props.distrattori,
      approfondimento: this.props.approfondimento,
      analisiParole: this.props.analisiParole,
      numero_files: object.keys(this.props.numero_files).includes(
        this.props.jsonFile[this.props.frase]["tipologia"]
      )
        ? this.props.numero_files[
          this.props.jsonFile[this.props.frase]["tipologia"]
        ].length
        : 0,
    )),
    headers: {
      "Content-Type": "application/json",
    },
  });
}
```

Figura 4.38. Fetch della REST API nella fase finale dell'Editor

La chiamata alla REST API, in particolare, utilizza il metodo POST, specifica il formato del request body (in questo caso stringhe *json*), ed il contenuto del body stesso, costituito da tutte le strutture costruite man mano che sono state inserite le informazioni richieste nelle pagine dell'Editor. Queste strutture, sono dizionari ottenuti dal contenuto dei file json presenti nel sistema (quindi non aggiornati) insieme ai contenuti nuovi, e mantenute all'interno della struttura di Slice introdotta e visibile in parte nella Figura 2.3.

L'API lavora in questo modo: per prima cosa controlla che la struttura sintattica della frase sia presente all'interno del sistema ed in caso negativo attraverso il modulo fs crea una nuova directory in cui inserire poi il file json della frase.

```
json = JSON.stringify(frase);
var path = `../exercise/${frase[prop]["tipologia"]}`;
if (!tipologie.includes(frase[prop]["tipologia"])) {
  fs.mkdir(path, (err) => {
    if (err) {
      console.log("error occurred in creating new directory", err);
      return;
    }
    console.log("New directory created successfully");
  });
}
```

Figura 4.39. Creazione della nuova directory nella cartella exercise

Successivamente, crea il file json della nuova frase, nominandolo con il numero delle frasi presenti nella directory, incrementato di una unità e del quale contenuto è il risultato del parsing³ della stringa JSON che descrive il dizionario dell'esercizio. Il numero delle frasi viene passato nel body della request e prelevato dal fetch della REST API eseguito al mounting del Component dell'Eserciziario.

```
fs.writeFile(
  `../exercise/${frase[prop]["tipologia"]}/${numero_files + 1}.json`,
  json,
  (err) => {
    if (err) {
      throw err;
    }
    console.log("Json file is updated.");
  });
}
```

Figura 4.40. Creazione del nuovo file json per la frase

³Metodo che trasforma una stringa del formato JSON in un oggetto Javascript

Lo stesso lavoro verrà effettuato per l'aggiornamento di tutti i file json dedicati a: memorizzazione delle parole (verbi, sostantivi, avverbi e aggettivi/pronomi) del dizionario, corrispondenze tra parola-paradigma-funzione nella frase, memorizzazione delle regole e dei colori delle subordinate e le strutture json dei verbi.

4.3 Strutture dati

La struttura centrale per lo sviluppo dell'Editor è stata lo slice: esso è stato creato con l'apposita funzione di Redux Toolkit (*createSlice*) ed ogni gruppo di reducer è stato dedicato per l'aggiornamento delle strutture che verranno, infine, scritte nei file json. Com'è possibile vedere nella figura 2.3, lo state iniziale di esso è un dizionario di oggetti e variabili, tutti aggiornati, nel corso dell'esecuzione dell'Editor, nei reducer opportuni. In particolare, i reducer sono stati divisi in:

- Quelli per la costruzione e gestione dei campi del dizionario che rappresenta l'esercizio (che prende il nome di *json* nell'initial state);
- Quelli per la costruzione e gestione dei campi nel dizionario dei verbi, che rappresentano modo, tempo, persona, forma e numero, insieme ai flag che indicano se è deponente e se appartiene alla frase del cum narrativo (il dizionario prende il nome di *jsonVerbi* nell'initial state);
- Quelli per la gestione analoga dei dizionari per la rappresentazione delle domande di approfondimento, dei distrattori, delle corrispondenze delle parole nel dizionario, del dizionario stesso e delle strutture per i colori e le regole delle nuove subordinate;
- Quelli per la gestione degli indici per il progresso delle pagine dell'Editor (che servono per effettuare i controlli nel Menu e per visualizzare i Components giusti) e delle pagine relative agli esempi negli aiuti;

Man mano che l'utente procede nelle pagine dell'Editor, vengono richiamati i reducer ed aggiornati i campi nei dizionari. Si prende come esempio il gruppo di reducer per la memorizzazione dei verbi della frase e delle loro proprietà (Figura 4.41). Il dizionario *jsonVerbi*, inizialmente vuoto, viene riempito con le coppie verbo-dizionario (avente quest'ultimo come campi le apposite proprietà) con la funzione *updateJsonVerbi*, richiamata appena vengono definiti i verbi nella terza pagina dell'Editor. Nella quarta pagina dell'Editor, ovvero quella per analizzare i verbi definiti, i campi vengono riempiti con le informazioni apposite tramite il reducer *updateJsonPropVerbi*. Al

termine di queste operazioni, nella pagina principale dell'Editor, viene creato un dizionario globale contenente i verbi già presenti e analizzati nel sistema (presi dal file json apposito), insieme a quelli appena analizzati nelle fasi sopracitate. Questo dizionario verrà passato come prop al Component *Termina* per poi essere scritto nel file dei verbi perchè venga aggiornato.

```
/**porzione di slice per la costruzione del file json per la rappresentazione dei verbi */
changeJsonVerbi(state, action) {
  delete state.jsonVerbi[action.payload]
},
updateJsonVerbi(state, action) {
  state.jsonVerbi[action.payload] = {
    tempo: "",
    modo: "",
    forma: "",
    persona: "",
    numero: "",
    deponente: false,
    "tempo-cum": false
  };
},
updateJsonPropVerbi(state, action) {
  state.jsonVerbi[action.payload[0]][action.payload[1]] = action.payload[2];
},
```

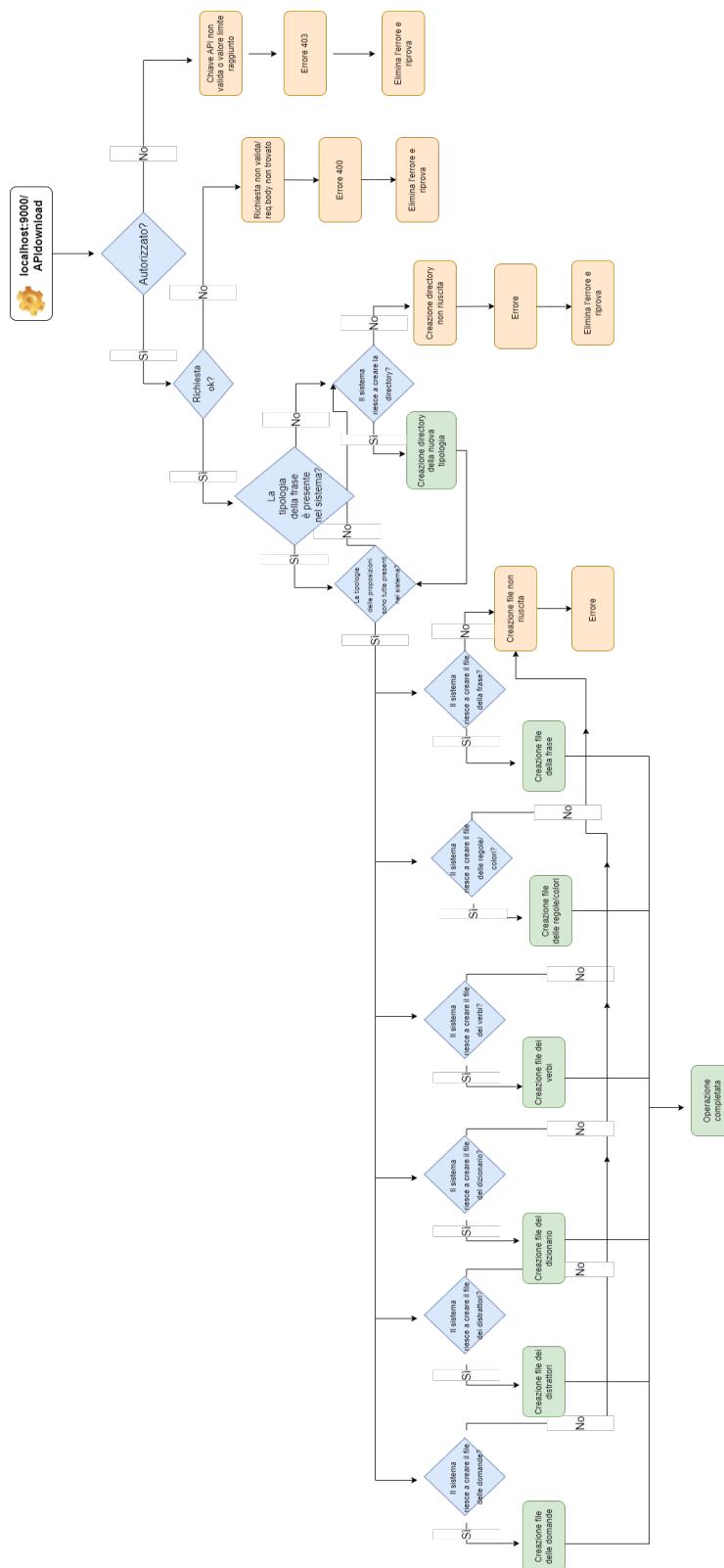
Figura 4.41. Porzione dei reducer per la gestione del dizionario jsonVerbi

```
/**porzione di codice per l'aggiornamento delle parole nel dizionario */
for (var p in paroleDizionario) {
  if (Object.keys(paroleDizionario[p]).includes("parteDiscorso")) {
    var diz_cp = {};
    for (let i in dizionario_copy["aggettivi-pronomi"]) {
      diz_cp[i] = dizionario_copy["aggettivi-pronomi"][i];
    }
    diz_cp[p] = paroleDizionario[p];
    dizionario_copy["aggettivi-pronomi"] = diz_cp;
  } else if (Object.keys(paroleDizionario[p]).includes("genere")) {
    /**porzione per inserire i sostantivi nel dizionario */
    var diz_cp = {};
    for (let i in dizionario_copy["sostantivi"]) {
      diz_cp[i] = dizionario_copy["sostantivi"][i];
    }
    diz_cp[p] = paroleDizionario[p];
    dizionario_copy["sostantivi"] = diz_cp;
  } else [
    /**porzione per inserire le congiunzioni nel dizionario */
    var diz_cp = {};
    for (let i in dizionario_copy["congiunzioni"]) {
      diz_cp[i] = dizionario_copy["congiunzioni"][i];
    }
    diz_cp[p] = paroleDizionario[p];
    dizionario_copy["congiunzioni"] = diz_cp;
  ]
}
```

Figura 4.42. Porzione di codice per lo smistamento delle parole nel dizionario

Lo stesso procedimento viene effettuato per l'aggiornamento di tutti i file json del sistema, con gli appositi controlli, se necessario. Un esempio di questo sono i controlli effettuati per aggiornare opportunatamente il file del dizionario, avente come chiavi le tipologie delle parole (sostantivo, congiunzione, aggettivo, ecc...) e come valori i dizionari descriventi le parole stesse. In questo caso, nel dizionario costruito dal sistema, sono presenti tutte le parole con i relativi campi. Per questo, i campi presenti serviranno come criteri per smistare le parole nella sezione giusta del dizionario (Figura 4.42).

4.4 Diagramma di flusso dell'API finale



Capitolo 5

Test di usabilità e sperimentazione

5.1 Testing del cum narrativo

La nuova tipologia del cum narrativo è stata testata su due studenti del quinto liceo, uno normotipo ed uno con DSA (in particolare con dislessia). La modalità di valutazione delle persone è stata la **Think Aloud**¹: essa ha permesso di individuare i dubbi e le difficoltà riscontrate, di mettere in discussione svariati dettagli e di arrivare al risultato finale.

Primo soggetto Il primo test è stato effettuato sul soggetto normotipo, ed è stata richiesta l'analisi di due frasi:

- *Galli, cum ad angustias perveniant, Caesar aciem instruit ut exercitus castra defendat;*
- *Consul cum milites urbem oppugnarent hortatus est equites ut subvenirent peditibus*

L'analisi di entrambe le frasi ha richiesto al soggetto 30 minuti: 20 minuti per la prima frase e 10 per la seconda (lo studente, infatti, ha preso dimestichezza con la piattaforma e non ha avuto problemi a completare il secondo esercizio).

¹Questa metodologia prevede che l'utente, nel corso del test, esprima ad alta voce i propri pensieri e commenti le proprie azioni mentre interagisce con la User Interface (UI) oggetto del test. Un osservatore è presente per stimolare l'utente senza influenzarlo e per raccogliere le sue osservazioni. [10]

Inizialmente, il soggetto ha notato il bottone *Istruzioni* ed ha fatto ricorso a esso per capire l'obiettivo della prima task. Successivamente, non l'ha più voluto utilizzare ritenendo sufficiente ed intuitivo il materiale presente sull'interfaccia. Non ha invece mai pensato a premere il bottone *Aiuto Nascosto*, cosa che ha fatto sotto sollecitazione nel momento in cui si è trovato in difficoltà nello svolgimento di determinate task e nell'analisi di parole quali ad esempio *peditibus* e *exercitus*. Nella traduzione del cum narrativo, ha erroneamente invertito l'ordine tra soggetto e verbo ed è riuscito a ragionare facendo utilizzo dell'helper nella figura 3.20 per trovare la risposta corretta.

I suoi commenti, al termine del testing sono stati relativi all'utilità della piattaforma. Il soggetto ha apprezzato particolarmente il format per cui le proposizioni vengono analizzate singolarmente e vengano forniti aiuti e spunti di ragionamento per arrivare alla risposta corretta. La mappa concettuale è stata valutata in modo positivo, soprattutto per il ripasso dei concetti fondamentali del cum narrativo.

Secondo soggetto Il secondo test, effettuato su una studentessa dislessica, è stato condotto richiedendo l'analisi della frase "*Galli, cum ad angustias perveniant, Caesar aciem instruit ut exercitus castra defendat*", ritenuta sufficiente per raccogliere le impressioni sulle nuove aggiunte alla piattaforma. Infatti, la studentessa era stata già coinvolta nel testing del lavoro di tesi precedente, quindi sapeva già muoversi nell'interfaccia, cosa che ha fatto in modo particolarmente intuitivo. Questo ha destato il mio personale stupore in quanto il fatto di ricordarsi come funzionasse la piattaforma ha significato, a mio avviso, che la ragazza ne fosse rimasta colpita.

Il tempo impiegato per svolgere l'esercizio è stato di circa 20 minuti. Ha fatto particolare utilizzo del bottone di *Aiuto Nascosto* per svolgere le task e per leggere il Tooltip delle parole, ed è rimasta entusiasta di scoprire che il dizionario fornisse anche la traduzione delle parole (cosa che non ricordava dal testing precedente). Per l'analisi delle parole, ha utilizzato anche l'Albero del ripasso, che ha giudicato molto utile e d'aiuto per comprendere al meglio la suddivisione della frase in proposizioni distinte.

Nella traduzione implicita del cum narrativo, ha voluto subito cliccare sul bottone di help e, grazie a quello, ha saputo ragionare sul corretto ordine tra soggetto e verbo. Ha applicato lo stesso comportamento per ragionare sui concetti di contemporaneità e anteriorità nell'analisi del verbo della proposizione con il cum, facendo uso dell'apposita tabella di aiuto (Figura 3.19).

Anche per lei, la mappa concettuale risulta utile per il ripasso.

5.2 Testing dell'Editor

La fase di test dell'Editor è stata condotta su due professori del liceo classico, sempre con la tecnica del Think Aloud.

Primo soggetto Il primo test è stato effettuato sulla docente che partecipa attivamente allo sviluppo della piattaforma e che, quindi, la conosce bene. L'Editor però, al momento del testing, non era mai stato presentato alla professoressa, ragion per cui si è ritenuto opportuno effettuare su di lei il primo test e si sono ritenuti validi i risultati ottenuti.

Il testing ha avuto una durata complessiva di un'ora. Inizialmente, il docente non ha compreso a pieno le finalità del software, e si è chiesto come mai chiedesse informazioni dettagliate e quasi "scontate" sulla frase latina. Dopo aver compreso che tutte le richieste erano volte alla costruzione delle strutture dati per la rappresentazione dell'esercizio, non ha avuto più dubbi sul senso delle richieste delle pagine. La frase che si è inserita è "*Galli cum ad angustias perveniant Caesar aciem instruit quod hostes viderat*". Infatti, si è richiesto al tester di pensare ad una frase con strutture verbali già presenti in un altro esercizio e con un nuovo tipo di subordinata, al fine di poter mostrare e provare tutte le funzionalità dell'Editor.

La docente ha ritenuto molto utile la funzionalità di completamento automatico per i verbi e non ha avuto difficoltà nell'esecuzione delle task. Non ha mai fatto uso del bottone di help. Non ha ritenuto particolarmente necessarie le spiegazioni relative al *conto* e ai *distrattori*, cosa che però ha personalmente giustificato dicendo che forse la ragione è dovuta al fatto di essere coinvolta nel progetto e quindi di sapere già la ragione dietro a determinate spiegazioni. Ha particolarmente apprezzato la pagina della scelta dei colori, soprattutto per il fatto che la palette permette di scegliere tutte le gradazioni di essi.

Secondo soggetto Il secondo test è stato effettuato su un'altra docente di latino. Inizialmente, non conoscendo lei la piattaforma, è stata fatta una panoramica di ciò che offre la stessa, in particolare mostrando l'esercizio sul cum narrativo su cui ho lavorato in questo tirocinio. Alla tester è stato richiesto, successivamente, di provare ad inserire nell'Editor la stessa frase del primo soggetto; questa scelta è stata motivata da due fattori: è stata ritenuta la frase giusta per testare tutte le funzionalità del software ed in più non si è voluto appesantire il lavoro del tester facendogli pensare alla frase da inserire, non essendo quello l'obiettivo di questa fase. La fase di testing è durata 50 minuti.

Il soggetto ha inizialmente avuto difficoltà a trovare il bottone di help in basso a destra, affermando di non avervi fatto caso e di non aver pensato immediatamente a potervi cliccare. Ha ritenuto quasi tutte le pagine semplici ed intuitive, ma alla task dedicata all'inserimento della traduzione e al drag and drop delle proposizioni ha fatto difficoltà nel distinguere i due compiti. Infatti, dopo aver ordinato le proposizioni, ha pensato di cliccare subito su "Procedi" e non si era accorta di dover inserire le traduzioni delle subordinate. Soltanto sotto sollecitazione ha potuto poi inserire le informazioni nei form appositi.

Alla pagina dedicata all'inserimento dei distrattori, ha ritenuto utile il Tooltip informativo che spiega il loro significato e cosa devono indicare e grazie a quello non ha avuto problemi a pensarli, utilizzando anche la tecnica del "falso amico", che non si era pensata in precedenza, dandoci così l'idea di poterla sviluppare in una tesi futura per automatizzare la generazione dei distrattori.

Una seconda pagina su cui ha avuto difficoltà è stata quella relativa all'inserimento delle domande di approfondimento, in quanto ha ritenuto poco chiara la dicitura "*domande sulle tipologie in generale*". Soltanto dopo una spiegazione e l'aiuto della professoressa che segue il progetto ha potuto pensare alle domande e risposte da inserire.

Nella pagina dedicata all'inserimento delle opzioni per la traduzione esplicita della frase ha particolarmente apprezzato il Modal di avvertimento che è spuntato nel momento in cui ha pensato di inserire la prima risposta come risposta corretta: ha infatti ammesso di non aver pensato a questa possibilità, ed ha scelto di cambiare l'ordine delle risposte.

Al termine del testing sono state poste varie domande alla persona sul suo giudizio complessivo nei confronti del software. La docente ha affermato di non sentirsi affaticata dalle task, ma dal tempo impiegato per inserire la frase e le informazioni relative a essa. Ha giudicato positivamente l'interfaccia, non ritenendola difficile da capire e abbastanza intuitiva. Nonostante ciò, ha espresso la necessità di un tutorial introduttivo da seguire prima dell'utilizzo della pagina, in quanto ha affermato che un docente alle prime armi con la piattaforma potrebbe trovarlo molto d'aiuto.

5.3 Considerazioni finali

Avendo testato le UI soltanto su quattro soggetti, ovviamente non è possibile trarre delle conclusioni certe e valide sull'usabilità del software; nonostante questo, alla luce delle informazioni raccolte, si può affermare che le criticità riscontrate sono state soprattutto relative alla lunghezza dell'Editor, cosa da porre in considerazione per un lavoro futuro che richiederà l'introduzione di un dizionario esterno dal quale attingere le informazioni sulla frase senza che il docente debba inserirle una per una. Per avere una visione più chiara e ampia sulle necessità degli utenti che si interacceranno alla piattaforma, bisognerà trovare una gamma maggiore di utenti su cui poter effettuare test accurati. Il testing sarà maggiormente attendibile nel momento in cui verrà inserita una gestione delle utenze e quindi potranno essere chiaramente definite le pagine che può gestire il docente e quelle che può utilizzare lo studente.

Conclusioni

Il lavoro di tirocinio ha avuto come primo obiettivo la modularizzazione del codice della tesi precedente, per poi introdurre una nuova tipologia più complessa e permettere ai docenti di inserire qualsiasi frase al fine di estendere l'Eserciziario e far lavorare gli studenti sulla base delle proprie esigenze didattiche. Le informazioni sono state gestite e organizzate con la struttura dati del dizionario nidificato, ritenuta indispensabile e la migliore nell'ottica di un'architettura del software facilmente estendibile da nuove strutture di help e tipologie.

Il tirocinio è stato multidisciplinare: infatti, il lavoro tecnico è stato arricchito e sostenuto da un'educazione continua nei confronti della dislessia, di come i ragazzi la vivano e delle difficoltà che la scuola italiana incontri nel costruire metodologie inclusive efficaci, che non discriminino o lascino indietro gli studenti con necessità speciali. Questo aspetto l'ho tenuto particolarmente a cuore, nutrendo un grande interesse nei confronti delle discipline pedagogiche e della didattica inclusiva ed un altrettanto grande desiderio di contribuire positivamente alla vita dei ragazzi nella scuola.

Avere alla base una causa tanto forte quanto attuale rende il progetto di "Latino facile: supporti per DSA" un grande punto di partenza per una generalizzazione ulteriore del player nei confronti di altre lingue e materie scolastiche.

La realizzazione dell'Editor di esercizi è stato il primo passo per una facile estensione della piattaforma, che potrà permettere in futuro la creazione di un vera e propria grammatica latina online, con supporti e particolari attenzioni per disturbi specifici dell'apprendimento, assente in questo momento.

Come precisato nei capitoli dell'elaborato, l'Editor e i Components dell'Esercizio dovranno necessariamente essere supportati da un solido sistema di backend, che possa in primo luogo gestire le utenze in modo tale da suddividere nel modo corretto i servizi offerti dall'applicazione, e successivamente permettere agli studenti e professori di personalizzare il sistema, con gli esercizi più opportuni per i docenti e le mappe concettuali personali per i ragazzi. Il sistema potrà, inoltre, costituire un sistema di

storage per il dizionario e alleggerire quindi il lavoro richiesto da parte dell'Editor. In vista di una demo definitiva, inoltre, futuri lavori di tirocinio potrebbero coinvolgere la realizzazione di un video tutorial per fare in modo che i futuri utenti della piattaforma possano sentirsi completamente guidati e motivati al suo utilizzo.

Essendo il progetto nato con una ragione alla base molto delicata, è importante che ogni piccolo contributo futuro alla piattaforma possa essere mosso ricordandosi di avere in mano due strumenti: la voglia di implementare cose sempre nuove e usabili da una parte e la visione di un domani in cui il lavoro dei docenti e (soprattutto) dei ragazzi normotipo e DSA possa sentirsi positivamente sostenuto e affiancato grazie all'applicazione.

Bibliografia

- [1] CORNOLDI, C. "Le difficoltà di apprendimento a scuola", Il Mulino, Bologna, 1999
- [2] LEGGE 170/2010. "Nuove norme in materia di disturbi specifici di apprendimento in ambito scolastico", articolo 5.
- [3] FACEBOOK INC. *Componenti e Props*. Available from:
<https://it.reactjs.org/docs/components-and-props.html>
- [4] DAN ABRAMOV AND THE REDUX DOCUMENTATION AUTHORS. *Redux*. Available from: <https://redux.js.org/>
- [5] OPENJS FOUNDATION. *The Node.js fs module*. Available from: <https://nodejs.dev/learn/the-nodejs-fs-module>
- [6] WEBKID. *React Flow- Introduction Docs*. Available from: <https://reactflow.dev/docs/>
- [7] FERRARI MATTHEW G. *React-Color-List*, dalla repository:
<https://github.com/matthewgferrari/react-color-list>
- [8] WORLD HEALTH ORGANIZATION, "International Statistical Classification of Diseases and Related Health Problems 10th Revision, Version for 2007". Available from: https://www.icdarnfo1.edu.it/wp-content/uploads/2020/07/Normativa_Classificazione-Internazionale-DSA-1.pdf
- [9] SCHIAVO G., MANA M., MICH O., ARICI M., "Tecnologie digitali e DSA", Editore Provincia autonoma di Trento - IPRASE, 2016
- [10] PAOLINI, M., "Thinking Aloud – Un Metodo Empirico per Testare l'Usabilità delle Interfacce". Available from: <https://matteopaolini.com/thinking-a-loud-un-metodo-empirico-per-testare-lusabilita-delle-interfacce>

Ringraziamenti

Dopo mesi e mesi di lavoro, è strano realizzare che la fatica piú grande sia trovare le parole giuste per esprimere l'enorme senso di gratitudine che provo in questo momento.

Il mio primo ringraziamento è dedicato al professor Sterbini e alla professores-sa Fabbri, per la disponibilità e fiducia rivoltami, e per avermi trasmesso tutta la loro dedizione nel mettere sempre noi studenti al primo posto. Li ringrazio anche per essersi raccontati con me ed avermi lasciato dei pezzettini della loro vita che mi porto a casa nel bagaglio di questi speciali tre anni.

Ringrazio mamma e papà, per aver vissuto le mie insicurezze e il mio cambiamento di rotta universitaria senza aver mai dubitato un attimo del fatto che potessi farcela.

Un ringraziamento speciale va a Francesca, che mi supporta da 13 anni, ricordandomi ogni volta di quanto sia importante mettere testa e cuore in tutte le cose.

Ringrazio i miei compagni di viaggio di *Mai una laurea*, a cui voglio un bene incredibile. Li ringrazio per aver condiviso con me questi tre anni e per aver reso ogni momento prezioso: dal memory delle macchinette di Fisica, ai pranzi in mensa, fino alle lunghe chiacchierate sul futuro.

Ringrazio in modo particolare Lucia, che è diventata ormai un punto di riferimento.

Ringrazio i *KOS*, in particolare Silvia, Valentina e Bagi per essermi state accanto in questi anni, per incoraggiarmi ogni volta a mettermi in gioco e per avermi ascoltata con pazienza nel momento in cui ne ho avuto il bisogno.

Volevo ringraziare anche Alessia, che dal 2013 è una grandissima luce nella mia vita. Perchè, nonostante i tanti chilometri di distanza e l'impossibilità di vederci spesso, è sempre lì a darmi coraggio.

Ringrazio Lorenzo, il regalo più bello che questa università mi potesse fare.

Le tue azioni sono i tuoi monumenti

Wonder

R.J. Palacio