

CORSO DI PROGRAMMAZIONE WEB E MOBILE

A.A. 2018-2019

TEACHY

<< Filippo Foladore, 894669 >>

Teachy

Il tuo assistente scolastico.

Introduzione

Il progetto consiste in un assistente scolastico per un docente di una scuola primaria e secondaria con cui può gestire le classi e gli studenti presente in esse.

Il funzionamento è semplice grazie all'interfaccia minimale ed auto-esplicativa e la sua asincronia permette di non far dover perdere la concentrazione e l'usabilità in seguito a ricariche delle pagine.

Le tecnologie utilizzate per le interfacce grafiche (HTML & CSS) sono ampiamente utilizzate nei più svariati dispositivi e il codice è scritto in JavaScript, un linguaggio potente e moderno per lo sviluppo web supportato dai browser di tutto il mondo.

Destinatari

Capacità e possibilità tecniche

Il destinatario è indirizzato ai docenti di scuole primarie e secondaria. L'applicazione non richiede capacità specifiche in quanto il funzionamento è chiaro e comparabile ad un registro cartaceo con il vantaggio di essere portatile e completamente sincronizzato qualora si dovesse cambiare dispositivo in corso d'utilizzo. Le interfacce sono chiare e non dispersive in ogni dispositivo che si andrà ad utilizzare sia esso un laptop a casa sul divano, un tablet in metro o sul bus o uno smartphone direttamente in aula o nei corridoi. L'utente che si approccerà all'applicazione in primo luogo troverà un layout a "card" (sezioni tematiche) che non lascia spazio a fraintendimenti arrivando dritto al punto in modo chiaro ed efficace. La quantità e qualità di banda necessaria è scarsa in quanto la UI minimale riduce al minimo la mole di dati caricati per alleggerire e accelerare l'esperienza d'uso anche fuori casa o in luoghi in cui la connessione sia scarsa, inoltre, le chiamate al server presenti asincrone, inviando solamente testo e la necessità di non dover

ricaricare la pagina ad ogni operazione eseguita permettono di alleggerire il carico della banda mantenendo in memoria la maggior parte della pagina.

Linguaggi

I linguaggi usati nelle interfacce sono di uso comune e non tecnici: anche un non docente potrebbe tranquillamente usare l'applicazione grazie all'uso di termini comuni a molti come "classe" o "studente". Un docente si troverà a suo agio nell'interfaccia minimale preferendo il "fare" all' "apparire" di molteplici immagini o testo inutile ai fini dell'usabilità che rischiano di distrarre un utente un po' meno esperto facendosi guidare dall'occhio verso percorsi inutili allo scopo dell'applicazione.

Motivazione

L'applicazione è destinata ad un ambito prettamente educational risultando inutile ad un utente che svolge un'altra professione. Una possibile implementazione alternativa cambiando i testi delle interfacce e tema del progetto potrebbe essere la gestione del personale di un'azienda o delle squadre di un campionato di calcio (es. Fantacalcio).

Data la posizione ricoperta dall'utente si presume che sia in grado di utilizzare l'applicazione da cima a fondo e in caso di difficoltà sono presenti richiami alla guida per imparare ad usare l'applicazione a pieno. Il modello di Bates si distingue in 4 categorie: ricerca attiva e consapevole (searching), monitoraggio (monitoring), esplorazione (browsing) e acquisizione/consapevolezza (awareness). L'utente sarà nella fase di "searching" quando cercherà l'informazione desiderata venendo reindirizzato alla pagina di guida. Il "monitoring" cioè la capacità di assorbire informazioni dall'esterno non è presente poiché la (voluta) scarsità di informazioni superflue è ridotta al minimo. Il "browsing" si verifica nel caso il nuovo utente che capita nella pagina e vedendo le 4 macro-categorie e si trovi disorientato naviga nella sezione delle guide ed impara in maniera attiva lo scopo dell'applicazione e le relative guide diventando pronto ad usare a pieno l'applicazione in caso volesse iscriversi. "Awareness" non è presente (v. monitoring).

	Active	Passive
Directed	Searching	Monitoring
Undirected	Browsing	Being Aware

Modello di valore

L'applicazione nella sua natura ha un valore per quanto riguarda la comodità nell'avere a portata di mano l'intero registro personale senza dover portarsi sulle spalle o in borsa mille fogli. Questo comporta un notevole risparmio di carta, inchiostro ed "ansia" dal perdere documenti riservati al docente. La velocità e semplicità di utilizzo sono lo scheletro del progetto e permettono all'applicazione con pochi click di arrivare al risultato desiderato. Ad un possibile investitore potrebbe interessare la potenziale espansione e integrazione di funzionalità all'interno dell'applicazione (esempi: agenda, condivisione con docenti, importazione/esportazioni classi, ecc...) o il diretto sviluppo di una branca di Teachy ("Teachy4Students") per studenti con funzionalità simili: registro elettronico, classi tematiche per materia per scambiare appunti con i compagni, agenda creata dal docente per fissare compiti in classe/assegnazione compiti.

Flusso dei dati

I contenuti sono forniti direttamente dall'utente e recuperati al login ottenendo così l'effetto di avere una cartella personale contenente i propri dati. L'archiviazione della propria password è criptata in fase di creazione rendendo assai difficile per una persona recuperarla in chiaro anche se questi è il gestore del database stesso. Per esempio una password con testo "test123" verrà criptata e resa una stringa di 60 caratteri che mediante il sito "How secure is my password" stima il tempo che servirebbe ad un computer per crackare la password; il tempo stimato medio è 7 anni Googol, 1 Googol equivale a 10^{100} o 1 seguito da 100 zeri (curiosità: il numero è all'origine del nome di Google). A livello client gli unici dati salvati sono quelli degli id della classe e i dati degli studenti per le operazioni di routine per un registro (ordinamento, ricerca, ecc) e sono cancellate prima della chiusura della pagina per garantire la non persistenza e conseguente privacy.

Aspetti tecnologici

L'intera struttura delle pagine è creata con HTML5 (validati con il tool di W3C) arricchito con EJS (Embedded JavaScript) per la visualizzazione veloce dei dati all'interno della pagina senza ricorrere a script esterni. Questi elementi EJS andranno a recuperare alcuni dati dell'utente per poter essere visualizzati in modo quasi istantaneo e usati in fase di scripting a supporto del codice JavaScript. L'archiviazione dei dati è affidata a MongoDB, un database schemaless potente e veloce, con

supporto di Mongoose, modulo dedicato a NodeJS per salvare e recuperare i dati. NodeJS (con arricchimento del routing di Express) è un elemento fondamentale all'interno dell'applicazione che servirà la comunicazione tra la parte client e la parte sottostante dell'applicazione, non visibile all'utente. AJAX permette inoltre di effettuare chiamate asincrone al database per evitare il ricaricamento della pagina per non perdere un attimo con stressanti.

Interfacce

L'applicazione è composta da 7 viste principali.

La home a cui l'utente accederà alla prima visita si presenta con un minimalismo estremo più orientata alla produttività. Sarà presente in alto una barra di navigazione e il body conterrà un layout responsivo a "card" con le 4 macro-categorie con una breve spiegazione del contenuto. Per un utente abituato con un layout di questo tipo sarà immediato accedere alla giusta sezione con un singolo click. L'assenza di scorrimento verticale in versione desktop donando all'applicazione un look "centro di controllo" (dashboard) si presta ad essere user-friendly per quegli utenti meno esperti nella navigazione web a cui effetti grafici elaborati e lo scroll della pagina verso il basso per trovare un'informazione che cercano potrebbe scoraggiare l'uso.

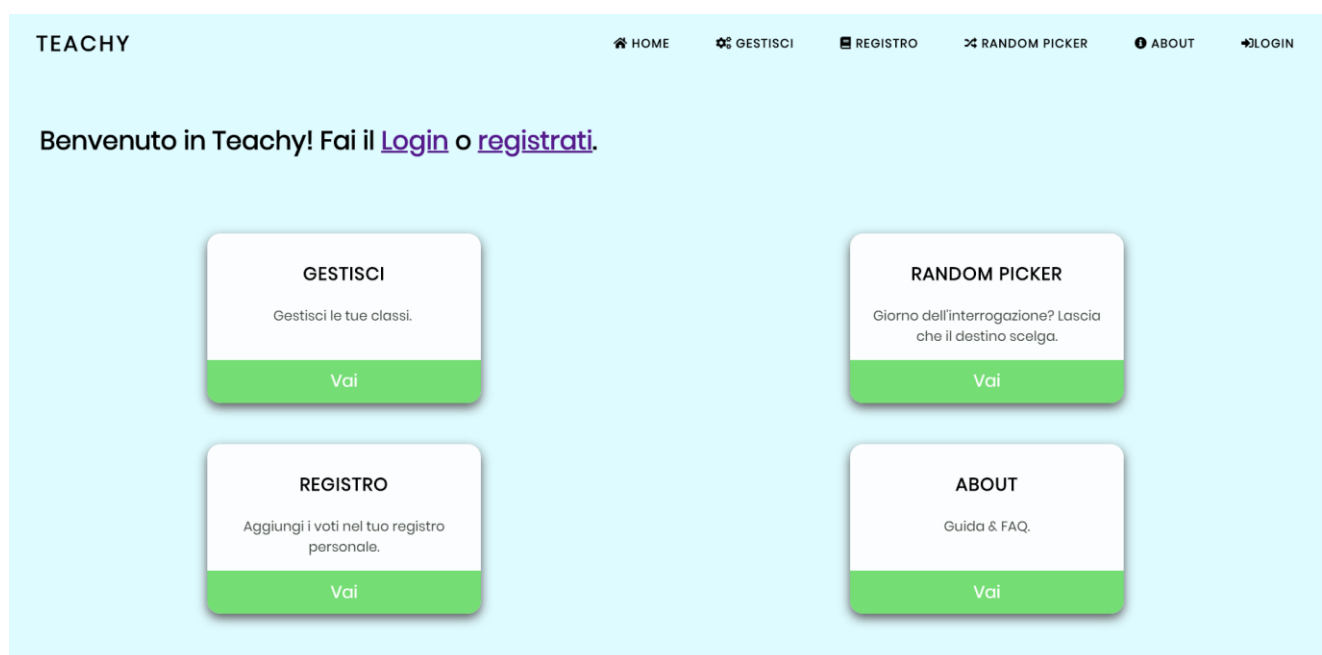


Figura 1.
Home screen su desktop

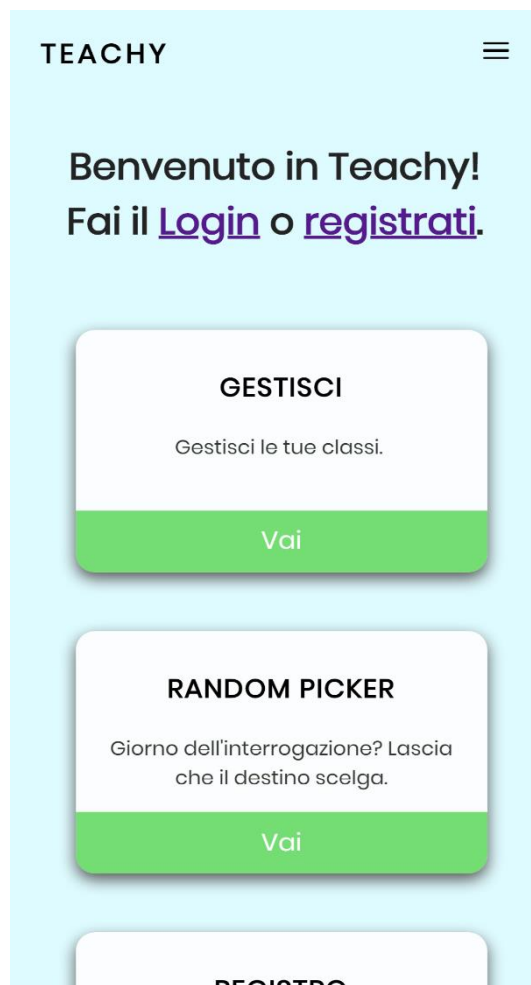
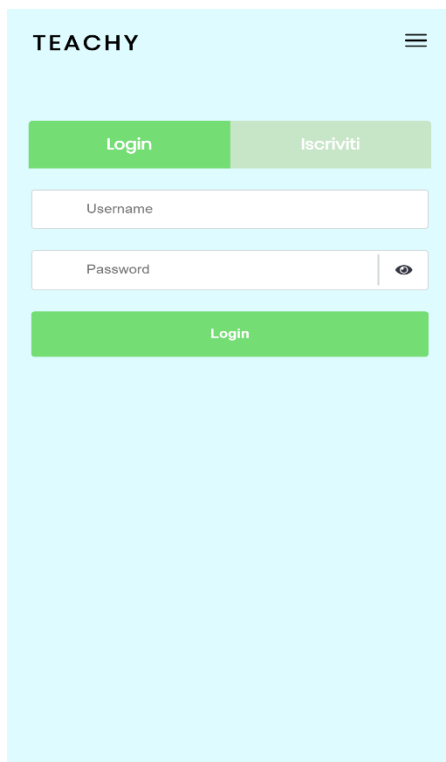


Figura 2.
Home screen su mobile

La sezione di login/registrazione è chiara e precisa con un form centrale con uno switcher in alto che indica la scelta dell'utente, premendo su una voce si andrà a caricare l'altra sezione dello switcher. La login riceve l'username e la password (che può essere mostrata per controllare eventuali errori) mentre la registrazione (iscrizione) necessaria per usare l'applicazione riceve dall'utente il nome, l'email, la scelta di un username, password e conferma della password. In caso di errori definiti a livello scripting (v. sezione Codice) verranno mostrati gli errori commessi in rosso.



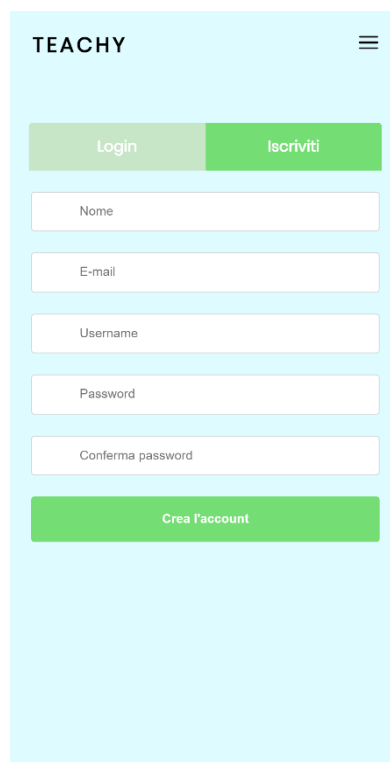
TEACHY

Login | Iscriviti

Username

Password

Login



TEACHY

Login | Iscriviti

Nome

E-mail

Username

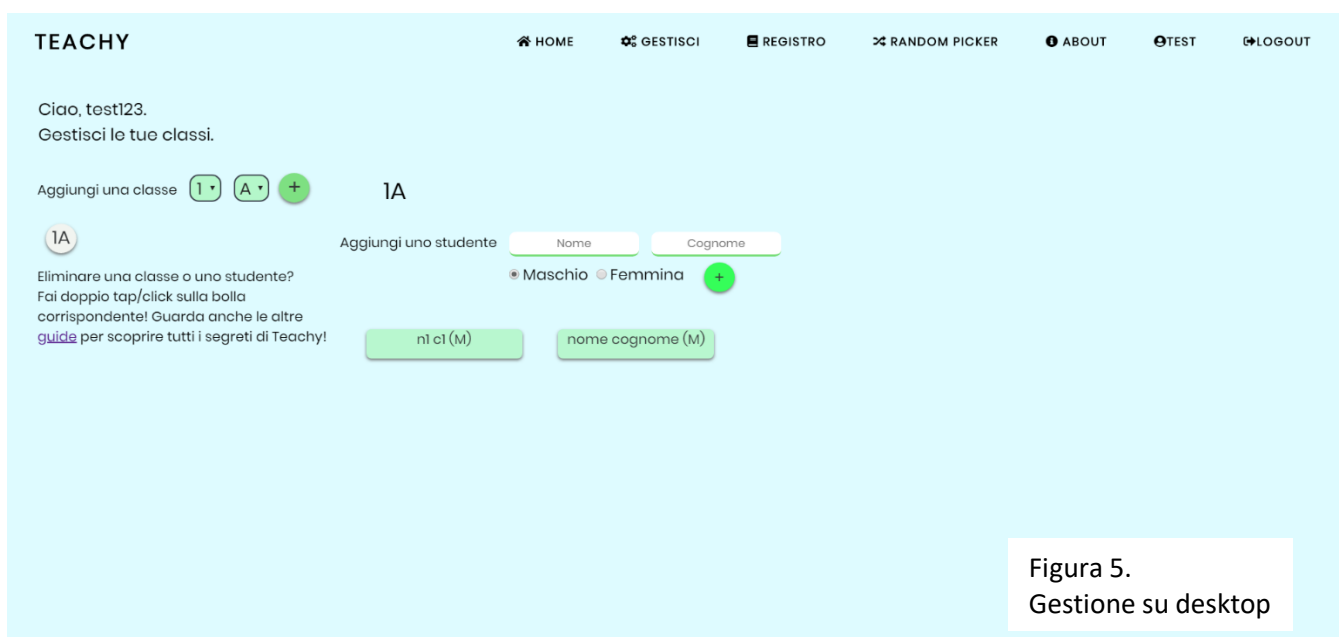
Password

Conferma password

Crea l'account

Figura 3 e 4.
Form di login e di
registrazione su mobile

La sezione di gestione presenta un layout a due colonne in versione desktop. Nella colonna di sinistra dedicata all'inserimento di una classe tramite form e la visualizzazione delle classi già inserite con grafica a bolla contenenti la classe e la sezione e la guida per eliminare una classe e l'invito a visualizzare la guida. Nella parte destra dello schermo una volta cliccata una bolla (classe) verrà visualizzata il nome della classe, un form per l'aggiunta di uno studente e la lista degli studenti già inseriti, lo sviluppo con metodo AJAX permette l'inserimento in sequenza di più studenti senza dover ricaricare pagine ad ogni inserimento.



TEACHY

HOME GESTISCI REGISTRO RANDOM PICKER ABOUT TEST LOGOUT

Ciao, test123.
Gestisci le tue classi.

Aggiungi una classe 1A +

1A

Aggiungi uno studente

Nome Cognome

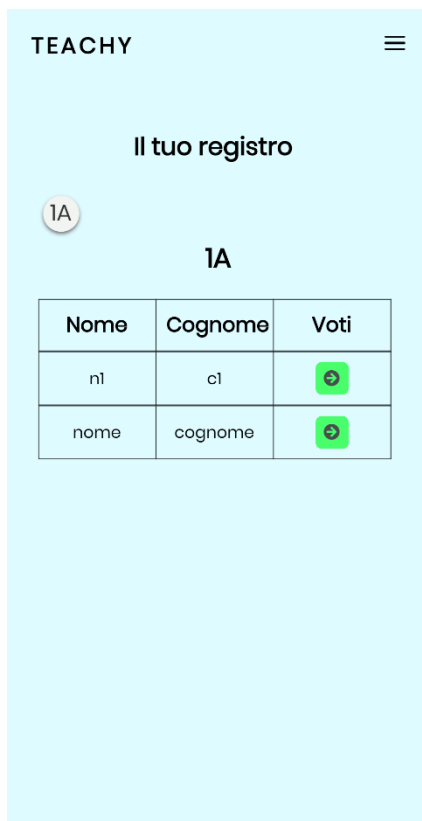
Maschio Femmina

Elimina una classe o uno studente?
Fai doppio tap/click sulla bolla corrispondente! Guarda anche le altre guide per scoprire tutti i segreti di Teachy!

ni ci (M) nome cognome (M)

Figura 5.
Gestione su desktop

La sezione registro è implementata in modo uguale alla sezione di gestione. Nella parte sinistra della pagina ci sarà la lista a bolle delle classi e a destra, al click di una classe, verrà generata una tabella contenente gli studenti, ogni riga di questa tabella ha un pulsante che permette di visualizzare più in dettaglio le informazioni dello studente selezionato.





Nome	Cognome	Voti
n1	c1	
nome	cognome	

Figura 6.
Registro del docente su mobile

La sezione di estrazione è molto semplice, una lista a bolle delle classi e un tutorial testuale per invitare l'utente che deve estrarre uno studente a premere su una classe per far partire il processo di estrazione.

La sezione di guide & informazioni è organizzata con un “accordion”, un elemento HTML che permette di contenere molte informazioni testuali racchiuse per sezioni che vengono aperte quando l’utente ci clicca sopra espandendo l’area e rivelando il testo contenuto. Le aree dell’accordion sono: guide, contatti, informazioni e cancella account. Nell’area ‘Guide’ sono presenti le guide per un utente per effettuare tutte le operazioni necessarie al funzionamento dell’applicazione, nei ‘Contatti’ sono presenti le tre icone dei principali social network (Facebook, Twitter e Instagram), un collegamento alla pagina GitHub contenente il codice usato per sviluppare l’applicazione e un’icona per mandare una mail con il servizio Gmail direttamente allo sviluppatore. Nell’area ‘Info’ è presente una breve descrizione del progetto e nella sezione ‘Cancella account’ è presente un bottone per cancellare il proprio account.

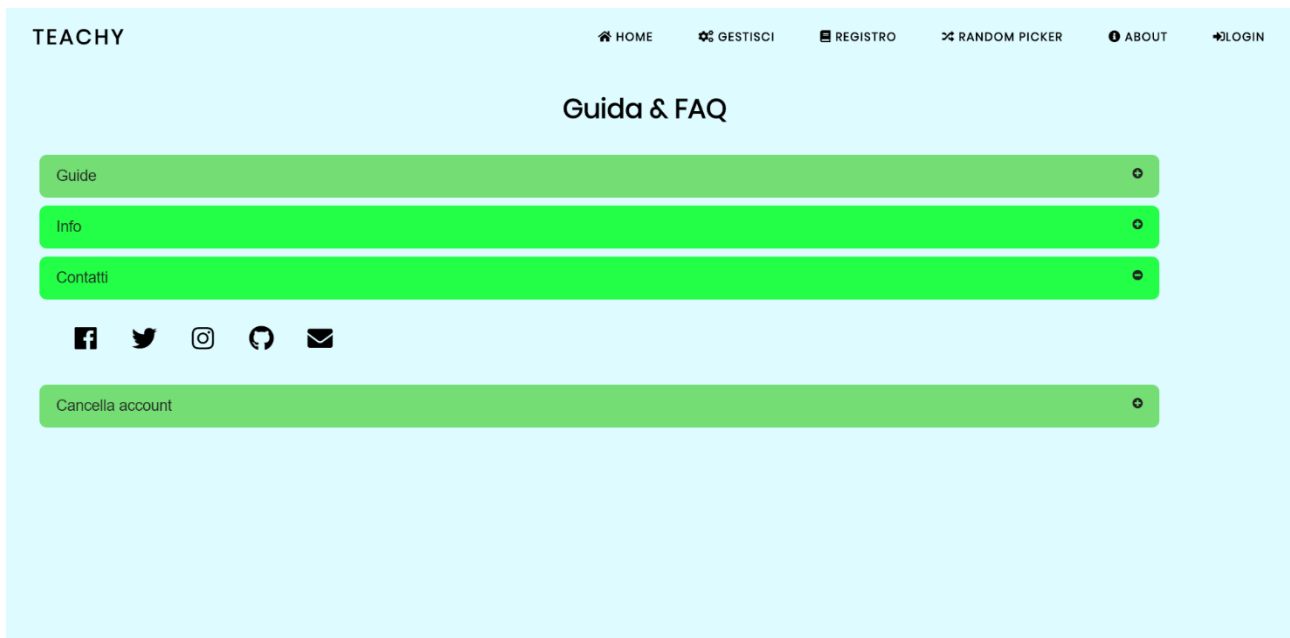


Figura 7.
Sezione dell’accordion aperto

Le interazioni con l'utente in caso di cancellazione (cancellazione classe, studente o account) sono presentate tramite popup chiari che espongono le operazioni da eseguire per portare a termine l'azione. Per le operazioni più delicate è presente un servizio di auto-chiusura che scatta dopo un tot di secondi per prevenire cancellazione accidentali.

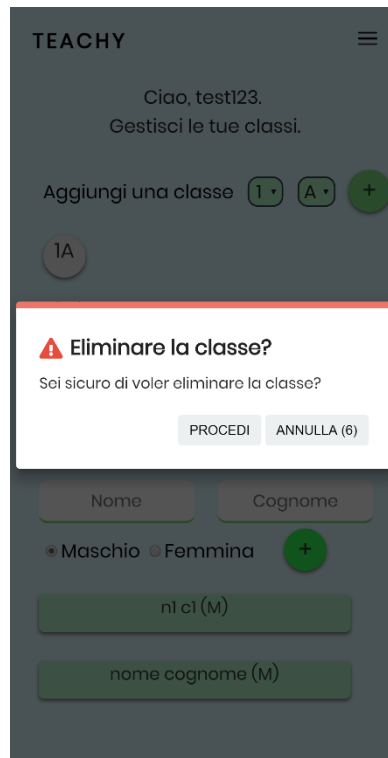



Figura 8.
Popup che chiede conferma dell'azione, nel tasto Annulla è presente il timer

Il menu richiamabile tramite l'icona hamburger () nelle interfacce per smartphone è arricchito con un'animazione semplice di visualizzazione e icone che richiamano l'azione desiderata.

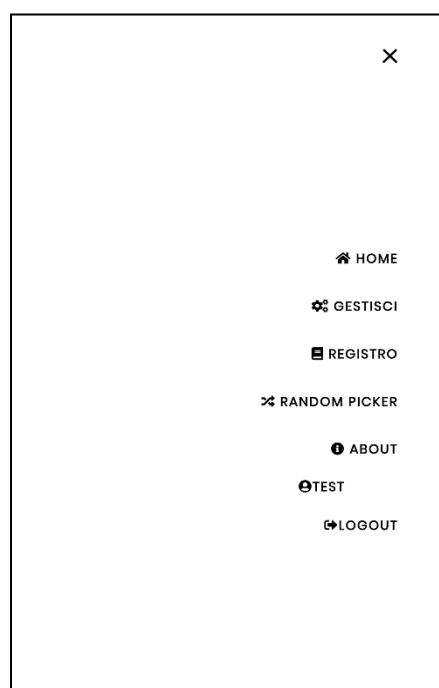
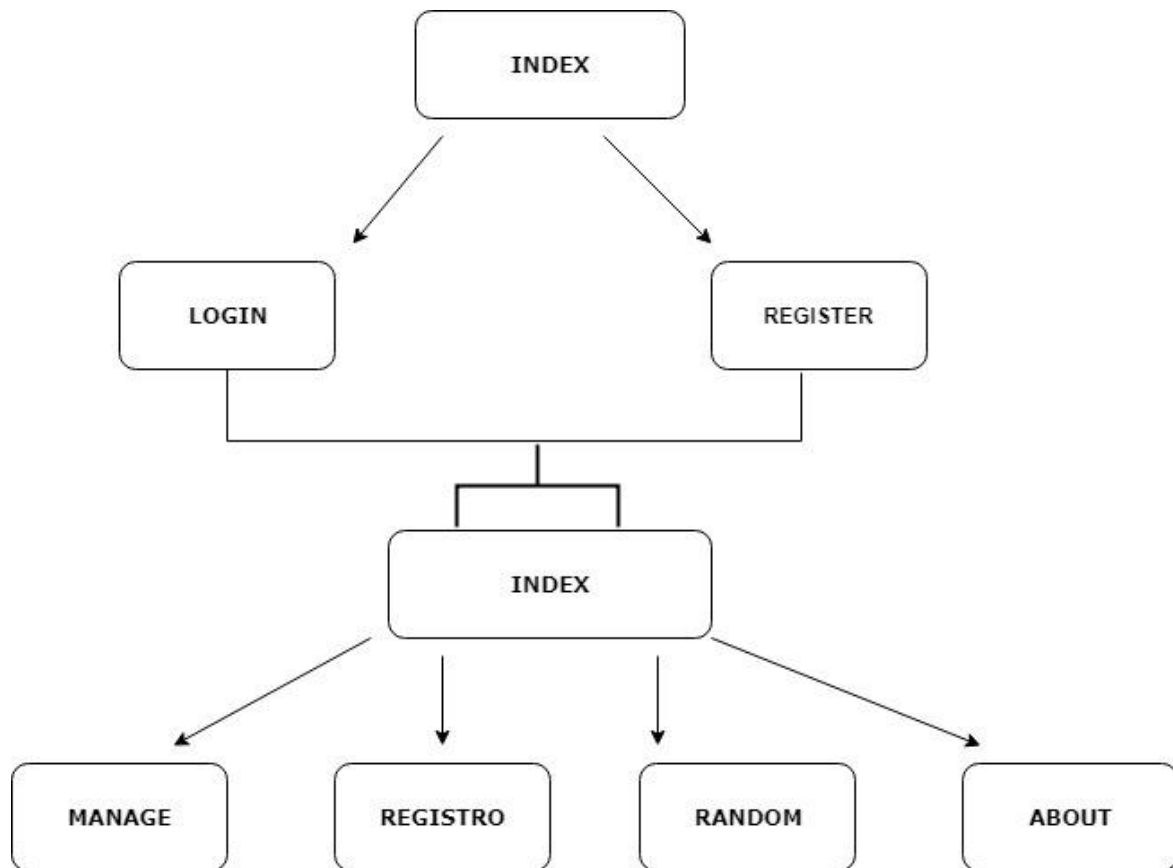


Figura 9.
Navbar richiamato e aperto

Architettura

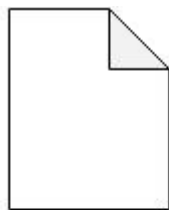
Diagramma gerarchico delle risorse



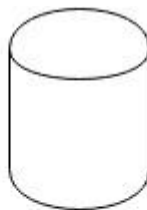
L'applicazione parte dalla home screen (index) e per natura si dovrà per forza fare il login o registrarsi. Una volta eseguita una di queste due azioni l'utente tornerà alla home e sceglierà quale pagina visitare.

Legenda per i diagrammi delle risorse

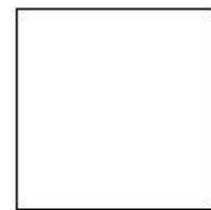
Legenda:



VIEW



MODEL



CONTROLLER

Diagramma descrittivo della risorsa 'login'

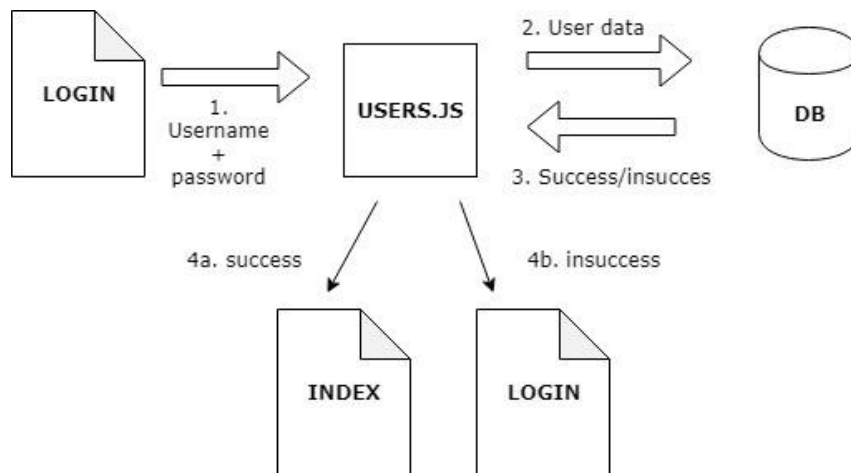


Diagramma descrittivo della risorsa 'register'

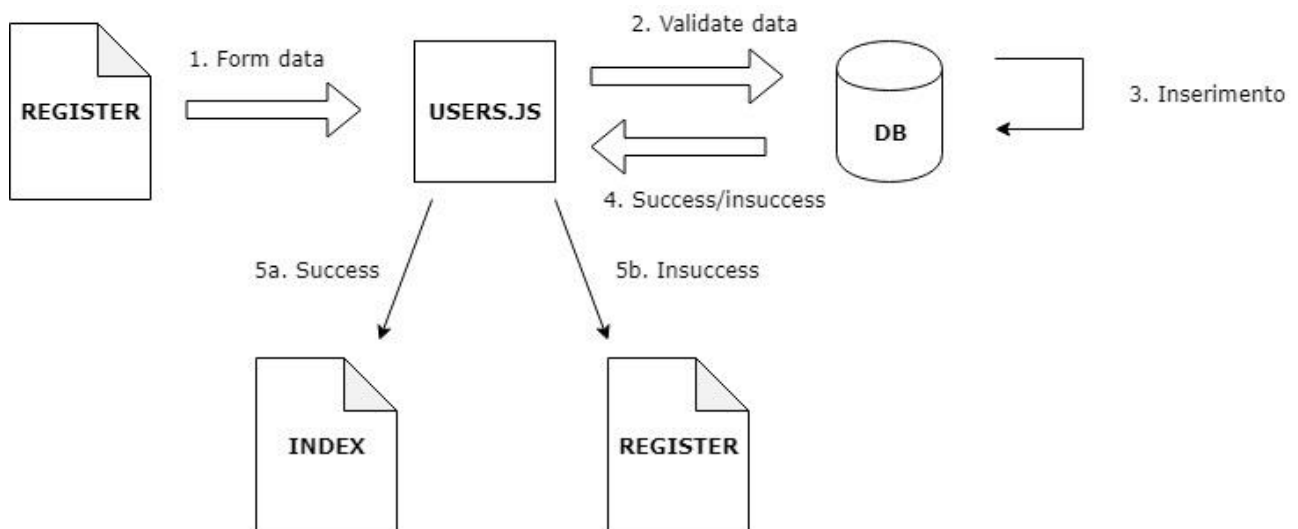


Diagramma descrittivo della risorsa 'manage' (per l'aggiunta di una classe)

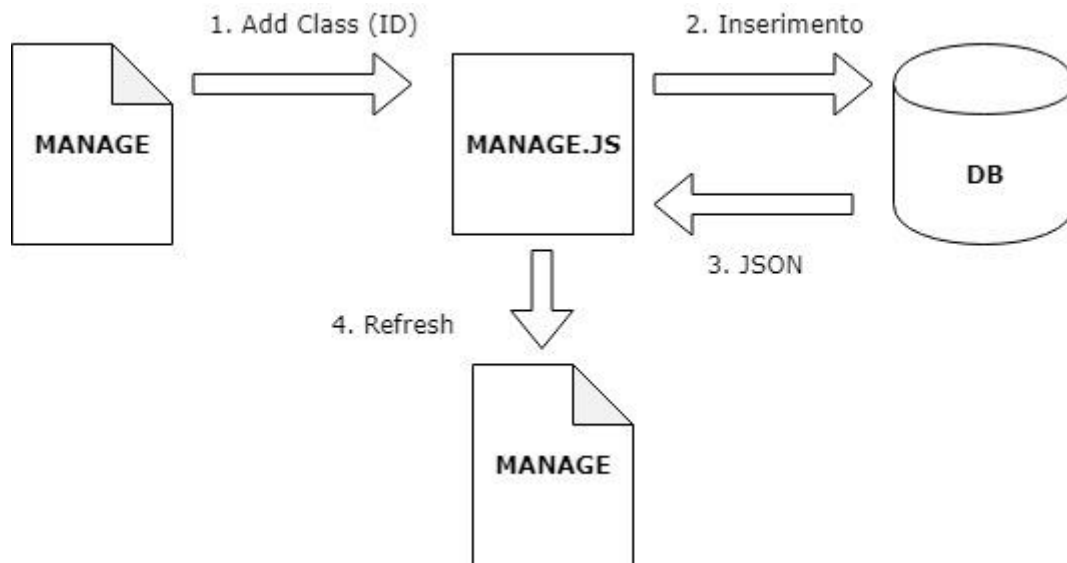


Diagramma descrittivo della risorsa "manage" (per le operazioni con AJAX)



Nota: per le operazioni di eliminazione di una classe e di uno studente il diagramma è uguale e cambierà il punto 1 del diagramma rispettivamente (Delete Class (ID) e Delete Student (ID)).

Diagramma composizione Database MongoDB (scomposto)

TEACHER	
name	string
email	string
username	string
password	string
created	date

CLASSES	
cName	string

STUDENT	
name	string
lname	string
gender	string
grades	array

HTML

Gli HTML validati dal sito W3C, standard mondiale per le tecnologie web, sono la struttura portante del sito e delle interfacce comprendenti elementi di uso comuni quali div (elementi di tipo divide per organizzare la pagina), paragrafi e form (arricchiti con EJS) per garantire la visualizzazione su tutti i browser moderni e smartphone.

Esempi di codice utilizzato sono:

Per il layout a card:

```
<div class="card">
  <div class="cardBody">
    <div class="cardTitle">
      <h3>Gestisci</h3>
    </div>
    <div class="cardDesc">
      <p> Gestisci le tue classi. <br><br></p>
    </div>
    <% if (!user) { %>
      <a href="/users/login"
        class="button">Vai</a>
    <%} else {%>
      <a href="/manage" class="button">Vai</a>
    <%} %>
  </div>
</div>
```

“card” è il contenitore principale della singola card, il body è racchiuso nel “cardBody”. Con la parte di EJS viene selezionato il riferimento del pulsante in base alla presenza o meno di un utente loggato;

Per il form di login/registrazione è stato usato una schermata con uno switcher per selezionare la modalità desiderata:

```
<ul class="switcher">
<li><a href="/users/login"
id="switcherLogin">Login</a>
</li>
<li><a href="/users/register"
id="switcherSignup">Iscriviti</a>
</li>
</ul>
```

CSS

Il foglio di stile che regola la formattazione grafica del documento è contenuto nella cartella 'public/css' e gli elementi di principale interesse sono (eliminando parti di codice non interessante dentro gli elementi presentati):

- `body {overflow: hidden;}` per bloccare lo scorrimento della pagina;
- `#nav:checked` è una checkbox resa invisibile per permettere di switchare tra il menu hamburger e il menu normale durante il resize (o settato dalle media queries al caricamento) della pagina, quando viene attivato (senza interazione dell'utente) questo va ad apportare le modifiche alla navbar quali: ruotare la prima e la terza riga del menu hamburger e nascondere quella in mezzo, dare al nav-wrapper (il contenitore della navbar) un colore bianco, renderlo a tutto schermo e far apparire in sequenza gli elementi della navbar con un delay di 0.1s ciascuno. Essendo la checkbox legata al menu hamburger tramite proprietà "for" di HTML questa verrà attivata alla pressione dell'icona del menu;
- `.nav-wrapper ul li:nth-child(x) a {transition-delay: 0.xs;}` per far apparire in sequenza gli elementi dell'hamburger menu;
- `.nav-btn i:nth-child(x) {...}` per creare le 3 righe che compongono l'hamburger menu;
- `.cardContainer {grid-template-columns: repeat(auto-fit, minmax(510px, 1fr));}` + `.card{display:grid;}` servono per ottenere il layout a "cards". Grid-template-columns specifica che il contenuto del contenitore si dovrà ripetere con gli elementi per un minimo di 510px e un massimo di 1 frazione di schermo. Con repeat si può specificare il numero di colonne/righe e la larghezza che queste devono occupare: in questo caso avremo che una card andrà ad occupare un minimo di 510px di schermo per un massimo di una frazione;
- `#mainManage` e `#ajaxContainer` sono definiti per il layout a due colonne con la proprietà float settata a sinistra e width al 25% della finestra per la colonna `#mainManage`, `#ajaxContainer` si andrà a posizionare nello spazio libero lasciato (il rimanente 75%) in caso di versione desktop e semplicemente sotto in versione mobile;

- L'accordion usato per la pagina della guide è ottenuto da button opportunamente stilizzati con animazioni e icone che cambiano in base se l'accordion è aperto o chiuso tramite le pseudo classi di CSS :after e is-open.

NodeJS & JavaScript

NodeJS è l'anello di congiunzione che permette la comunicazione tra il client e il server e tra server e database. Il file `app.js` è il file che contiene la parte backend cioè la parte non visibile all'utente e si apre con la definizione di tutti i moduli aggiuntivi necessari al corretto funzionamento dell'applicazione.

Esempio di integrazione di modulo: `const mongoose = require('mongoose');` con cui si richiede il modulo "mongoose", un middleware per comunicare con il database schemaless MongoDB.

I file di route sono usati per una maggiore pulizia e modularità del codice con cui si impostano i percorsi da seguire nei diversi casi di URL navigati dall'utente.

Esempio di file di route: `let users = require('./routes/users');` e `app.use('/users', users);` specifica che per ogni richiesta fatta ad un URL contenente 'users' si dovrà usare il file di route corretto specificato nella cartella 'routes'.

Esempio del file di route 'users.js' interpellato in caso di navigazione in un URL con '/users':

```
router.get('/register', function(req, res){
  res.render('register');
});
```

specifica che un URL così composto: '/users/register' si andrà ad eseguire la funzione di renderizzare la pagina 'register'.

NodeJS per la gestione dei dati è strettamente collegato al database; per fare ciò è necessario innanzitutto collegare l'applicazione al database. Grazie a Mongoose l'operazione di collegamento è molto semplice: `mongoose.connect(config.database);` connette il database specificato nel file 'database.js' nella cartella 'config'.

Con il routing si andrà inoltre ad interpellare il database per scambiare dati per eseguire le operazioni CRUD (Create, Read, Update, Delete).

Esempio di operazione di aggiunta al database di una classe:

```
router.post('/add/:id', function(req,res){
  var c = req.body.cNum; //classe
  var s = req.body.cSez; //sezione
  var addClass = c+s;
  var data = {cName: addClass};
  Teacher.findOneAndUpdate(
    {_id: req.params.id},
    {$push: {classes:data}},
    {upsert:true}, //se non esiste aggiunge
    function(err, succ){
      if (err) {res.send('Qualcosa è andato storto, riprova')}
      else {
        res.redirect('back');
      }
    }
  );
})
```

In questa funzione POST vengono passati come parametri l'URL visitato '/add/:id' dove :id è aggiunto lato server e specifica l'id dell'utente loggato al sito. Si definiscono in seguito le variabili interne alla funzione quali 'c' ed 's' che verranno unite per formare il dato effettivo da inserire ('data'). Viene poi chiamata una funzione appartenente alle API di Mongoose sullo schema del database Teacher per interrogare il server. Questa funzione cerca all'interno del documento un id corrispondente a quello passato come parametro (e recuperabile con 'req.params.id') e aggiunge il dato all'interno del database nella posizione specificata nelle opzioni della funzione.

Se l'utente non ha più bisogno di una classe può eliminarla tramite un doppio tap/click gestito dal modulo JavaScript 'Interact'. Una chiamata di questo tipo è specificato dalla funzione (con eliminazione delle parti superflue):

```
interact('.tap-target').on('doubletap', function (event){..})
```

All'interno della chiamata Interact è presente un popup che compare e chiede conferma dell'operazione, una volta confermata è lanciata una chiamata AJAX gestita da JQuery (nel codice è presente anche una chiamata di tipo XMLHttpRequest):

```
$.ajax({
  url: '/manage/deleteClass/' + localStorage.getItem('classID'),
  contentType: 'application/json',
  type: 'POST',
  success: function (data) {
    $('#classList').empty();
    for (var i=0; i < data['classes'].length; i++){
      $('#classList').append(
        "<li class='classList tap-target'" + "id= ' " +
        data['classes'][i]['_id'] + "'>" +
        data['classes'][i]['cName'] +
        "</li>");
    }
    $("#className").empty();
    $("#students").empty();
    $('#invisibleId').empty();
    location.reload();
  })
})
```

La seguente chiamata sarà indirizzata all'URL specificato con l'aggiunta dell'ID usato poi nel backend per recuperare il corretto documento, il ContentType è settato al JSON per una gestione senza errori dei dati manipolati. Quando la chiamata sarà terminato con esito positivo verrà svuotata la lista delle classi (bolle) e ricaricata la pagina per rendere effettive le modifiche.

In fase di registrazione vengono definite regole per poter creare un account create con il modulo JQuery "jQuery-validation". La validazione avviene ad ogni input dell'utente ed è regolato dalle regole di lunghezza minima di nome, username, email e la definizione della password che deve essere di almeno 5 caratteri e contenere un numero attraverso le regular expression (RegEx). Estrazione di codice dal file "form.js", aggiunta del metodo customizzato 'secure' per la password:

```
$.validator.addMethod('secure', function(value, element) {
  return this.optional(element)
    || value.length >= 5 //maggiore di 5 caratteri
    && /\d/.test(value) //un numero
    && /[a-z]/i.test(value); //caratteri
}, 'La password deve essere di almeno 5 caratteri, contenente almeno un numero.');
```

Conclusioni

Il progetto qui presentato si propone come un tentativo di 'svecchiare' le scuole primarie ancora troppo attaccate al cartaceo cercando di mantenere di alto livello l'esperienza d'uso e l'interfaccia grafica per non avere compromessi dal punto di vista qualitativo. Il progetto può essere rifinito e reso più efficiente ma anche ampliato come menzionato prima ad esempio con una nuova applicazione indirizzata agli studenti o ai genitori. La sua modularità di pensiero rende l'applicazione facile da convertire in qualsiasi idea che abbracci il tema di "salvare informazioni e recuperarle". Con un'espansione il prodotto potrà rivelarsi interessante per una ipotetica funzione condivisa tra insegnanti o studenti di uno stesso istituto.

Nota bibliografica e sitografica

1. <https://validator.w3.org/>
2. <https://uigradients.com/>
3. <https://www.npmjs.com/>
4. <https://nodejs.org/it/>
5. <https://www.mongodb.com/it>
6. <https://mongoosejs.com/>
7. <https://github.com/>
8. <https://fontawesome.com/>
9. <http://interactjs.io/docs/>
10. <https://jquery.com/>
11. <https://fonts.google.com/>
12. <https://css-tricks.com/>
13. <https://developer.mozilla.org/it/>
14. <https://www.draw.io/>