**Compilazione**

**Moduli Online**

**Relazione Tecnica**

**prodotto da Lorenzo Cuoghi**

Modena, 17 agosto 2018

**Sommario**

[1 Introduzione 3](#_Toc523826791)

[1.1 Obiettivo 3](#_Toc523826792)

[1.2 Strumenti e linguaggi utilizzati 3](#_Toc523826793)

[2 Funzionalità e utilizzo dell’applicazione 4](#_Toc523826794)

[2.1 Organizzazione del sito 4](#_Toc523826795)

[2.1.1 Pagina principale 4](#_Toc523826796)

[2.1.2 Visualizza/Compila documento 5](#_Toc523826797)

[2.1.3 Submit 6](#_Toc523826798)

[2.1.4 Accesso Staff 7](#_Toc523826799)

[2.1.5 Pagina principale – versione Staff 7](#_Toc523826800)

[2.1.6 Modifica/nuovo documento 8](#_Toc523826801)

[2.1.7 Iscrizioni 8](#_Toc523826802)

[2.1.8 Pagina di amministrazione 9](#_Toc523826803)

[2.2 Tipologie di utenti 9](#_Toc523826804)

[2.1.9 Utente comune 9](#_Toc523826805)

[2.1.10 Personale 9](#_Toc523826806)

[2.1.11 Amministratore 10](#_Toc523826807)

[2.1.12 Use Case Diagram 11](#_Toc523826808)

[2.3 Sintassi per la modifica dei documenti 12](#_Toc523826809)

[2.1.13 Tipi di campo 12](#_Toc523826810)

[2.1.14 Nome dei campi 12](#_Toc523826811)

[2.1.15 Formato dei campi 13](#_Toc523826812)

[3 Organizzazione e implementazione del codice 15](#_Toc523826813)

[3.1 Model 15](#_Toc523826814)

[3.1.1 Class Diagram 15](#_Toc523826815)

[3.1.2 Entity Relationship Diagram 16](#_Toc523826816)

[3.2 View 17](#_Toc523826817)

[3.2.1 Detail view 17](#_Toc523826818)

[3.2.2 Edit view 18](#_Toc523826819)

[3.2.3 Funzione create\_input 19](#_Toc523826820)

[3.2.4 JavaScript/jQuery 20](#_Toc523826821)

[3.2.5 AJAX 20](#_Toc523826822)

[3.3 Controller 21](#_Toc523826823)

[3.3.1 State Diagram 22](#_Toc523826824)

# Introduzione

## Obiettivo

L’obiettivo di questo progetto è la creazione di un sito web accessibile agli studenti (principalmente, ma non solo) in cui sia possibile cercare, visualizzare, compilare e scaricare documenti e moduli.

Il sito deve quindi anche consentire la creazione e la modifica di documenti da parte del personale tramite una sezione in cui sia possibile accedere solo attraverso credenziali riservate.

mmm.. non so se metterlo

Lo scopo del progetto è anche quello di ridurre il cartaceo e il numero di visite alla segreteria e la creazione di un potenziale spazio unificato per reperire moduli, iscriversi a eventi e seminari, compilare certificazioni ecc.

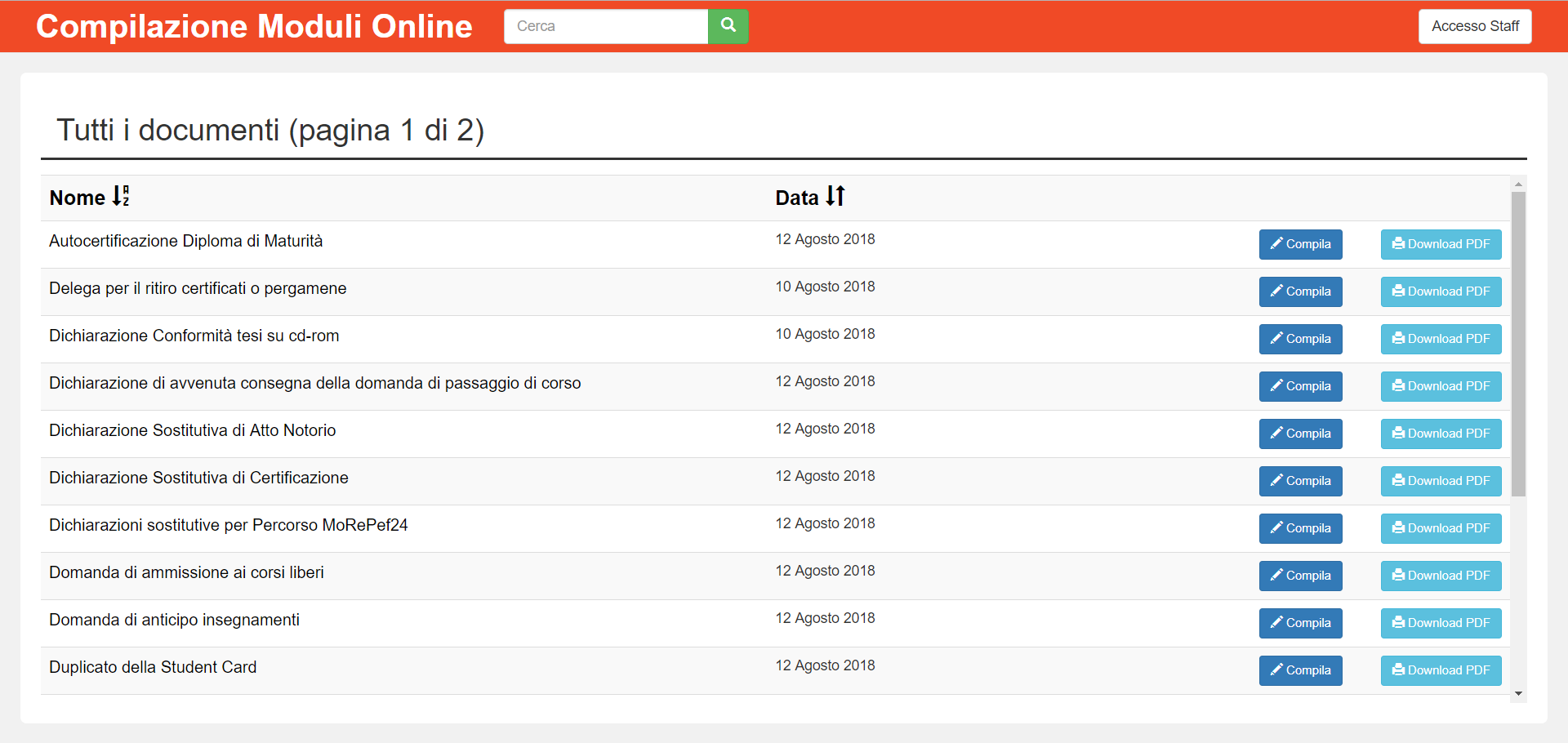
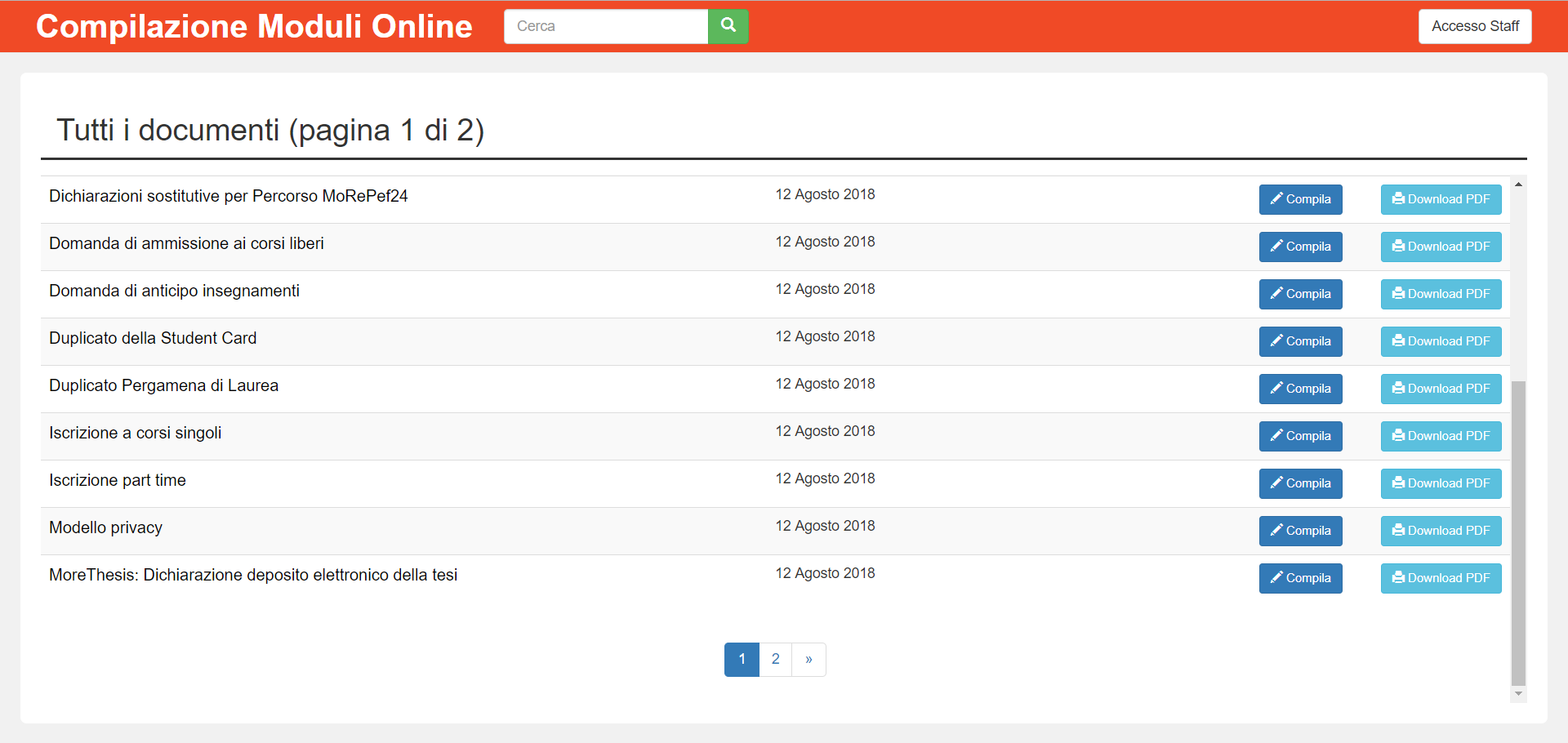
## Strumenti e linguaggi utilizzati

* PyCharm: IDE (Integrated Development Environment) utilizzato nella programmazione, in particolare per il linguaggio Python. È sviluppato dalla società ceca JetBrains. Supporta lo sviluppo web con Django;
* Microsoft SQL Server: DBMS relazionale prodotto da Microsoft;
* Django: web framework con licenza open source per lo sviluppo di applicazioni web, scritto in linguaggio Python, che segue il paradigma "Model-View-Controller";
* Python: linguaggio di programmazione dinamico ad alto livello general purpose. Offre un forte supporto all'integrazione con altri linguaggi e programmi ed è fornito di una estesa libreria standard;
* HTML: HyperText Markup Language, linguaggio a marcatori che definisce la struttura dei documenti ipertestuali (pagine web);
* CSS: Cascading Style Sheet, linguaggio che definisce lo stile grafico dei documenti HTML;
* Bootstrap: libreria di componenti front-end open source per lo sviluppo con HTML, CSS e JS. Comprende un sistema di griglia reattivo, componenti precostruiti completi e potenti plugin creati su jQuery;
* JavaScript: linguaggio di scripting lato client orientato agli oggetti, permette di eseguire azioni e gestire eventi all’interno di pagine web;
* jQuery: è una libreria JavaScript per applicazioni web. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché implementare funzionalità AJAX;
* AJAX: Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito refresh da parte dell'utente.
* CKEditor: editor di testo WYSIWYG open source di CKSource che può essere utilizzato all'interno delle pagine web. È scritto in JavaScript ed è distribuito sotto le licenze GPL, LGPL e MPL. È compatibile con la maggior parte dei browser.

# Funzionalità e utilizzo dell’applicazione

## Organizzazione del sito

### Pagina principale

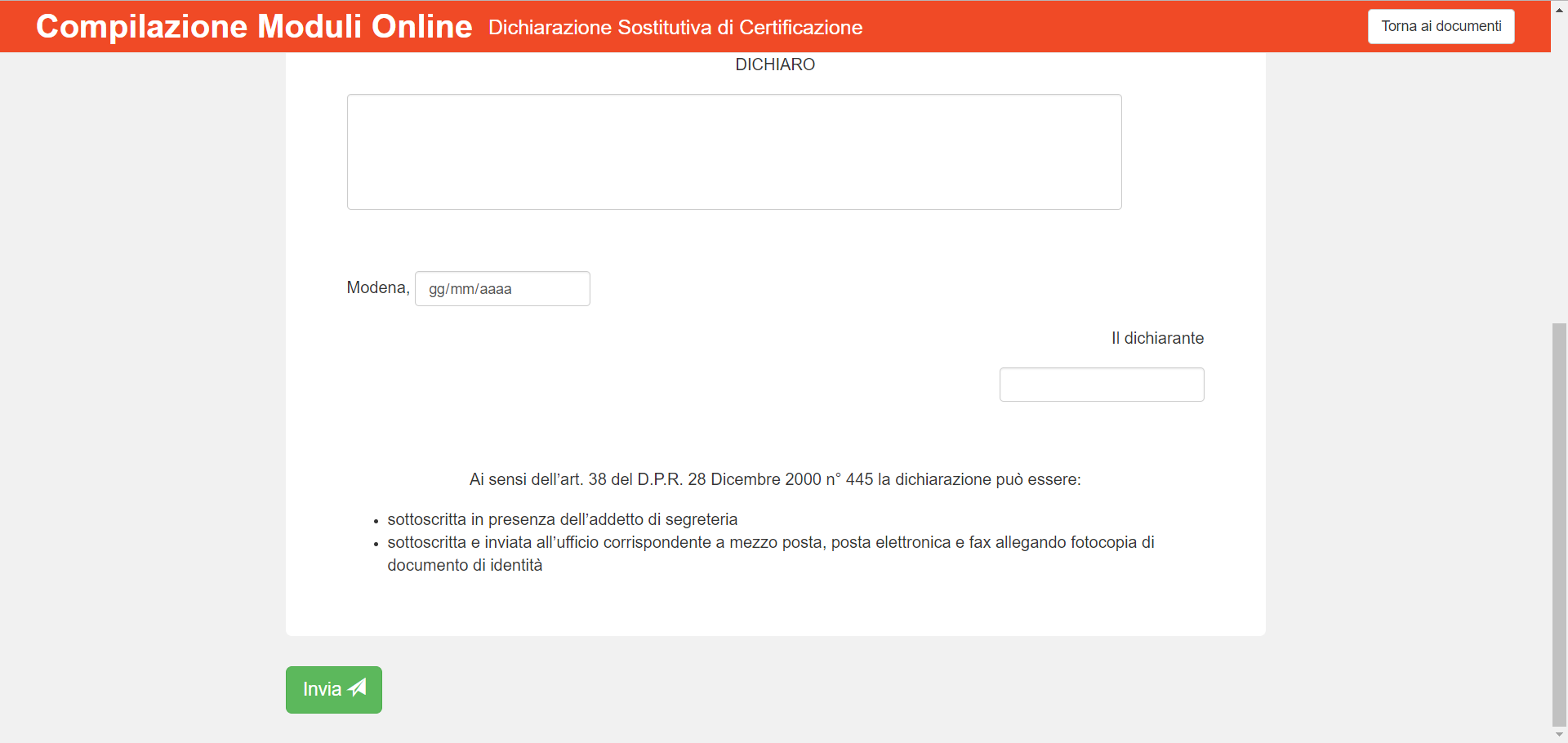
Questa è la home page che si incontra appena si entra nel sito. Presenta l’elenco di tutti i documenti esistenti nel sito mostrando il nome e l’ultima data di modifica, ordinati in ordine alfabetico e suddivisi per pagine di 15 elementi ciascuna, navigabili attraverso i pulsanti dedicati in fondo all’elenco. Possono essere riordinati per nome o per data, entrambi sia ascendente e discendente. Sono presenti per ogni documento i pulsanti “Compila” e “Download PDF”: il primo reindirizza il browser verso la pagina in cui è possibile compilare il documento (anche cliccando sul nome si accede alla stessa funzione), mentre il secondo permette di scaricare il file pdf del documento.

È poi possibile cercare i documenti per nome attraverso il campo Cerca, che mostrerà in tempo reale i documenti corrispondenti alla ricerca.

Infine, il pulsante “Accesso Staff” reindirizza il browser verso la pagina in cui il personale autorizzato può autenticarsi (si veda 2.1.4 e 2.1.5).

(1 o 2 screenshot?)

### Visualizza/Compila documento

In questa pagina si accede attraverso il pulsante “Compila” a fianco dei documenti nella pagina principale. Qui visualizzato il documento per intero ed è possibile compilarlo inserendo i propri dati personali.

In ogni documento potrebbero essere presenti diversi tipi di campi compilabili:

* campo testo
* campo numero
* campo data
* campo e-mail
* campo obbligatorio
* campo commento (textarea)
* radio button 
* campo selezione (select) 
* checkbox

Per confermare l’avvenuta compilazione premere il tasto “Invia” in fondo alla pagina, che porta automaticamente verso la pagina di conferma.

Altrimenti è possibile tornare alla pagina principale grazie al pulsante “Torna ai documenti” in alto a destra o cliccando il titolo “Compilazione Moduli Online”.

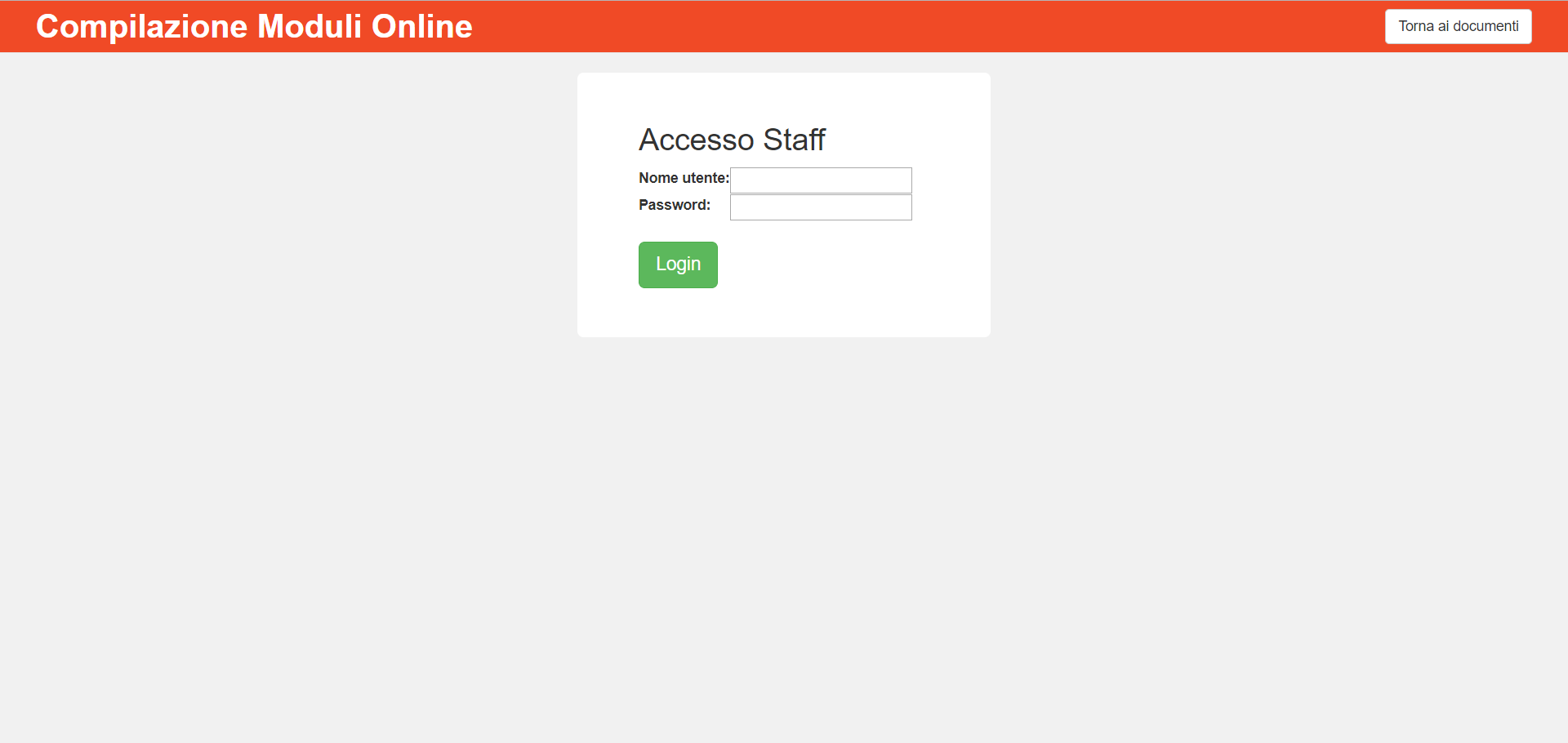
(1 o 2 screenshot?)

### **Submit**



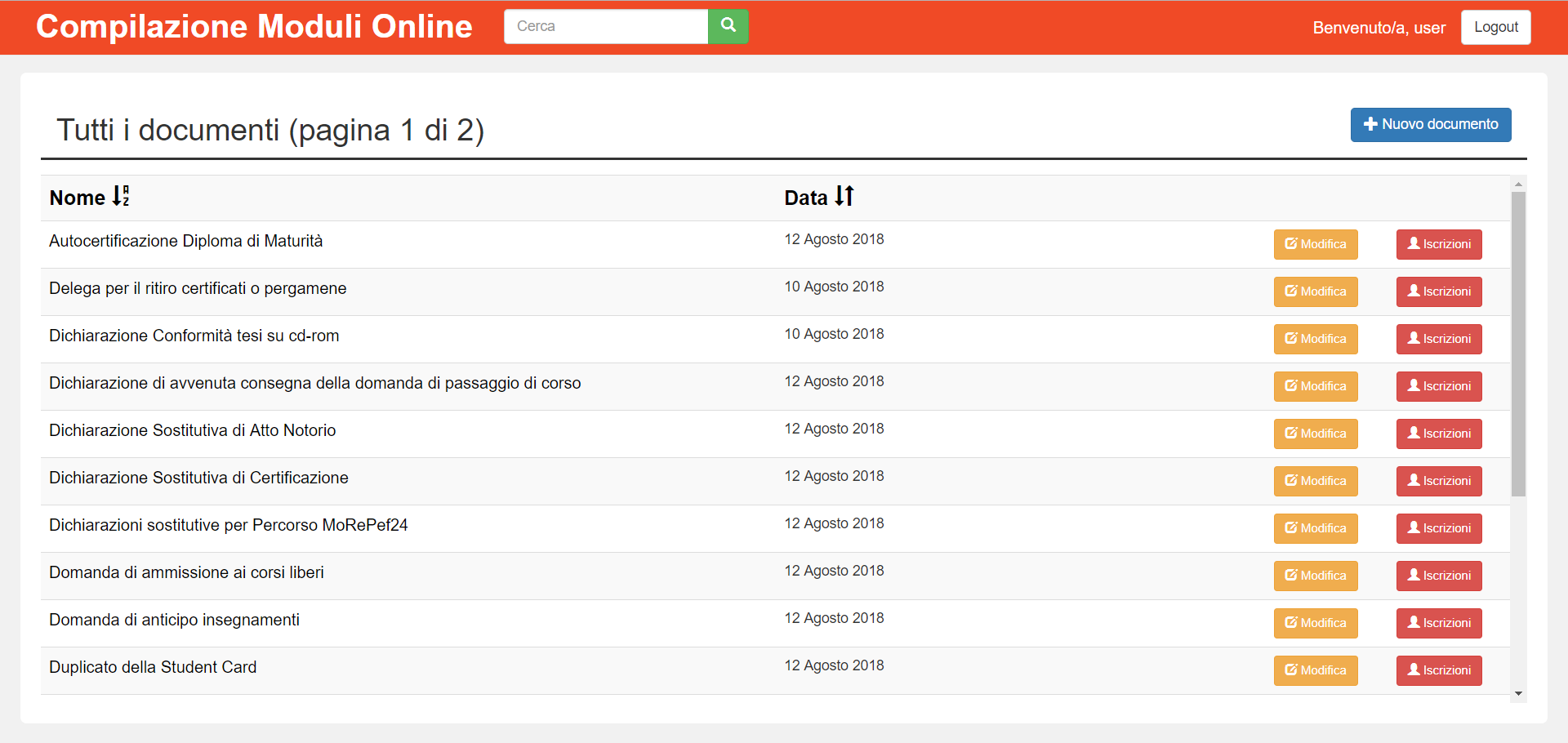
Questa è la pagina in cui si viene portati una volta premuto il pulsante “Invia”. Qui si conferma l’avvenuta registrazione e il salvataggio nel database. Dopo 5 secondi si viene riportati automaticamente alla pagina principale, come indicato dal timer, o in alternativa si può premere “clicca qui” per andarci immediatamente.

### Accesso Staff



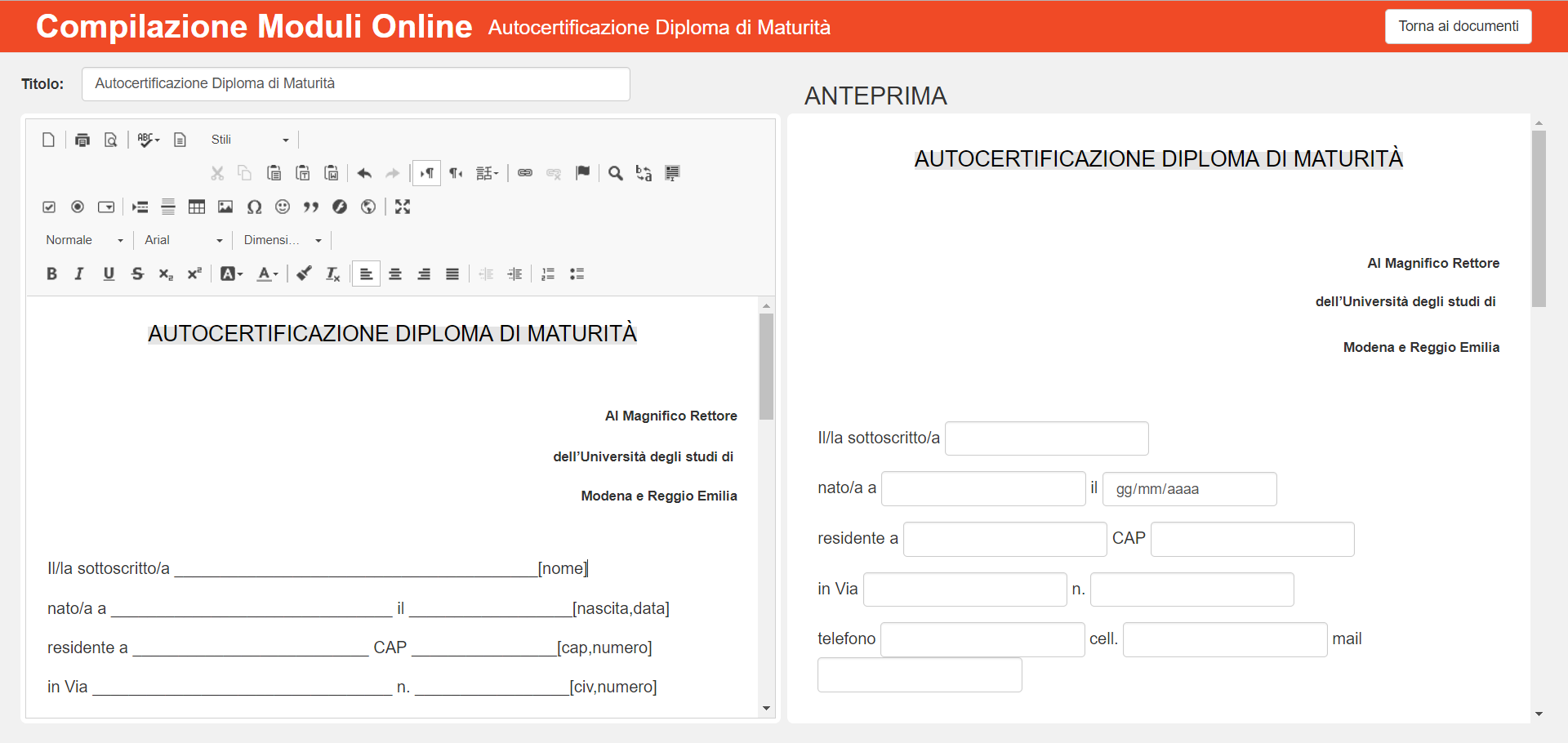
Questa è la pagina di accesso in cui si viene portati dal pulsante “Accesso Staff” della pagina principale. Qui il personale può inserire le proprie credenziali per autenticarsi e accedere alle funzioni riservate nella pagina principale (vedi 2.1.5).

### Pagina principale – versione Staff



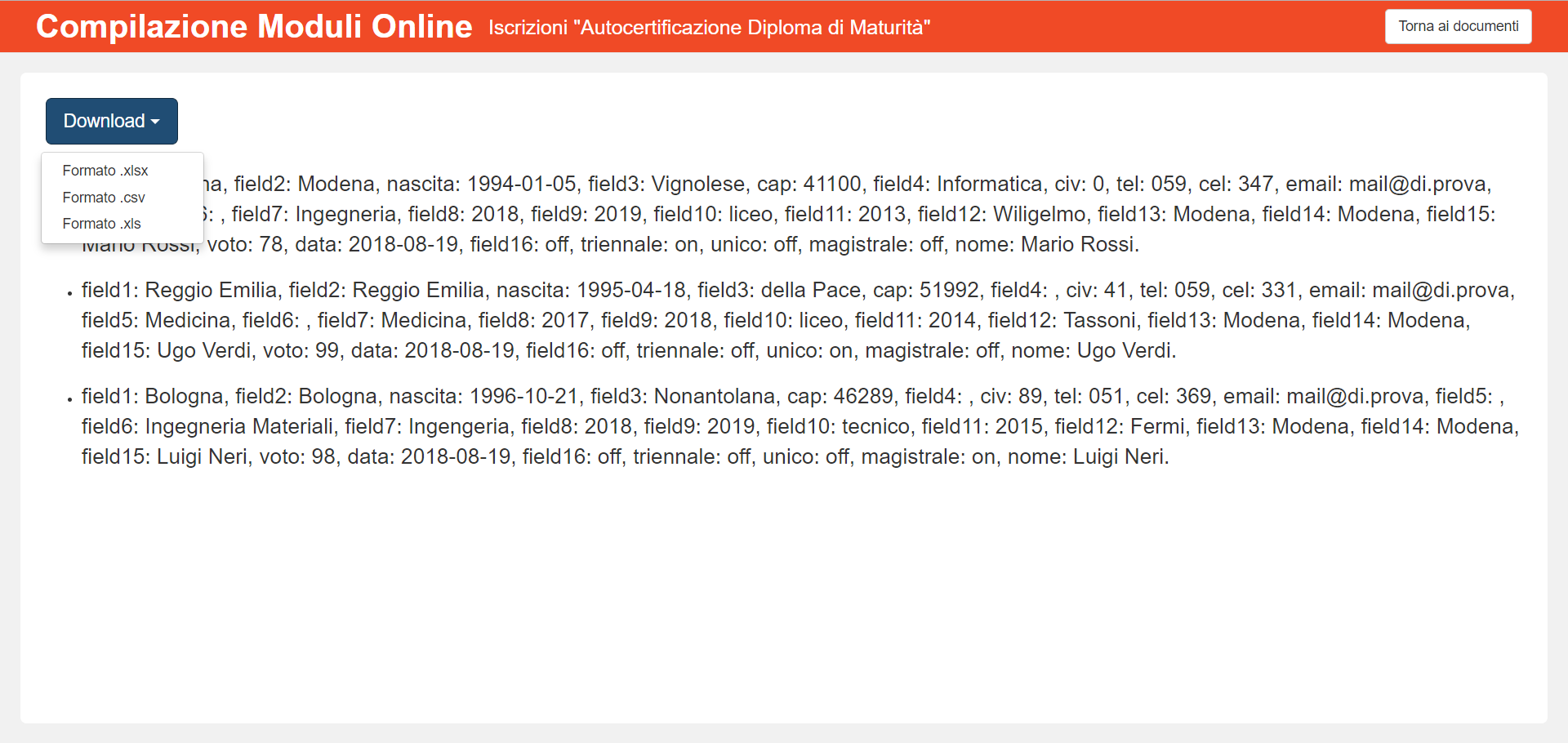
Questa versione della pagina iniziale può essere visualizzata solo dal personale una volta effettuato l’accesso. Sono presenti i pulsanti “Modifica” e “Iscrizioni” al posto di “Compila” e “Download PDF” ed è presente in più il tasto “Nuovo documento”. “Modifica” porta alla pagina che permette di modificare il documento, “Iscrizioni” a quella contenente l’elenco degli iscritti e “Nuovo documento” apre la pagina di modifica con un documento vuoto.

### Modifica/nuovo documento



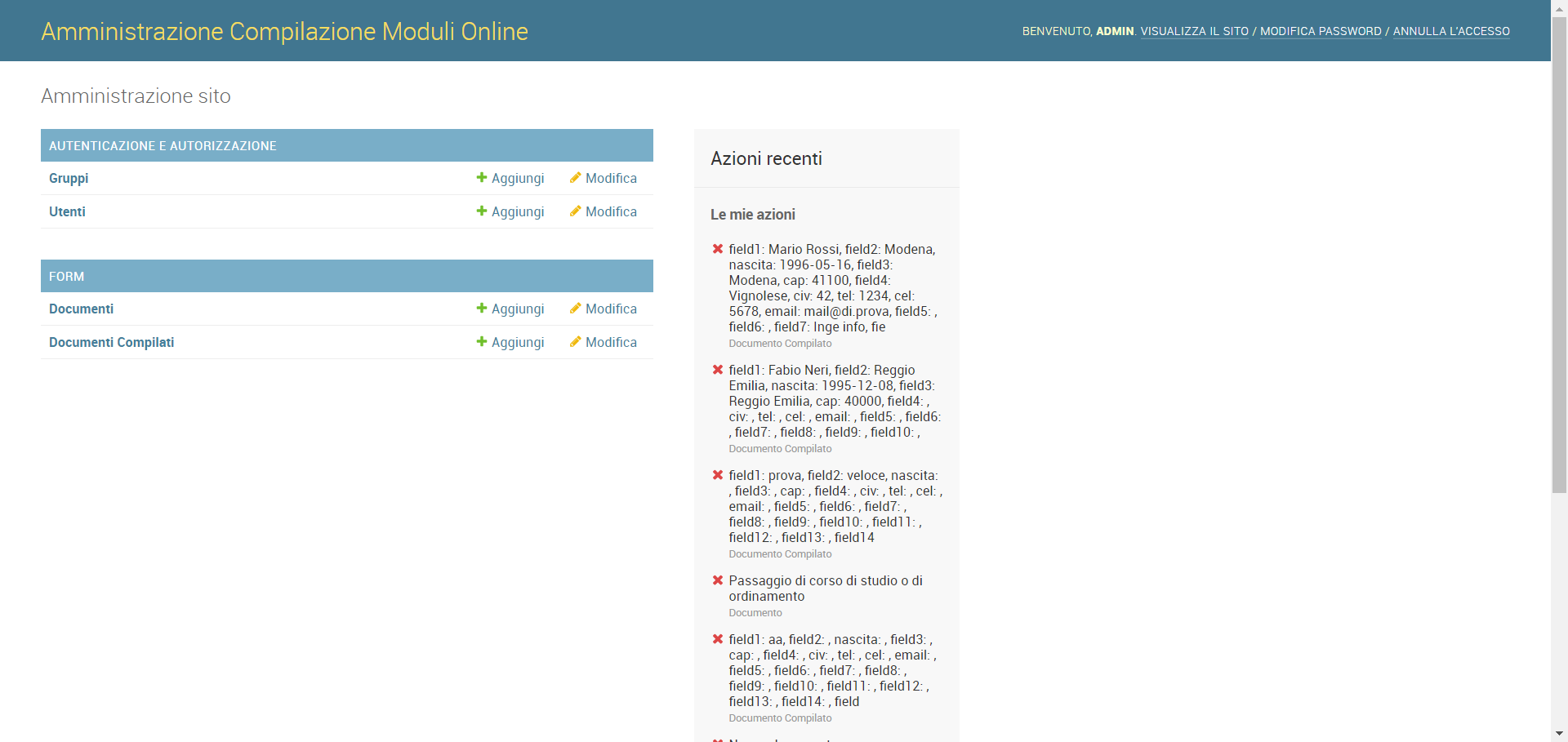
È possibile accedere a questa pagina solo avendo effettuato l’accesso. In questa pagina è possibile modificare il titolo e il contenuto del documento, o crearlo da zero nel caso di un nuovo documento. Non è presente un pulsante di salvataggio in quanto questa funzione è effettuata automaticamente ad ogni modifica. Non è possibile modificare la data in quanto verrà aggiornata automaticamente. Nella parte destra viene mostrata l’anteprima del documento finale come verrà mostrato nella pagina di Compilazione (vedi paragrafo 2.1.2). Nella parte sinistra è possibile modificare il documento grazie all’editor (CKEditor) ispirato ai più famosi programmi di elaborazione testi. La sintassi con cui scrivere i diversi tipi di campo è spiegata nel paragrafo 2.3.

### Iscrizioni



È possibile accedere a questa pagina solo avendo effettuato l’accesso. Qui viene visualizzato l’elenco dei documenti compilati, rappresentati come la lista dei campi compilabili presenti nel documento e i relativi contenuti che sono stati inseriti dall’utente. Attraverso il pulsante “Download” è possibile scaricare un foglio di calcolo nei formati più comuni contenente la lista degli iscritti.

### Pagina di amministrazione



Questa pagina è accessibile solo dall’amministratore all’indirizzo “/admin”. Qui è possibile visualizzare, modificare, creare ed eliminare ogni elemento del database come documenti, campi, documenti compilati, campi compilati e ogni loro attributo. È anche possibile creare ed eliminare nuovi utenti.

## Tipologie di utenti

### Utente comune

Rientra in questa categoria ogni utente che accede al sito unicamente per compilare o scaricare documenti, quindi senza effettuare l’accesso. Non gli è possibile utilizzare le pagine di modifica e iscrizioni, né quella di amministrazione.

Appartengono a questa categoria principalmente gli studenti, ma anche ogni altro utente che necessiti delle funzionalità del sito.

### Personale

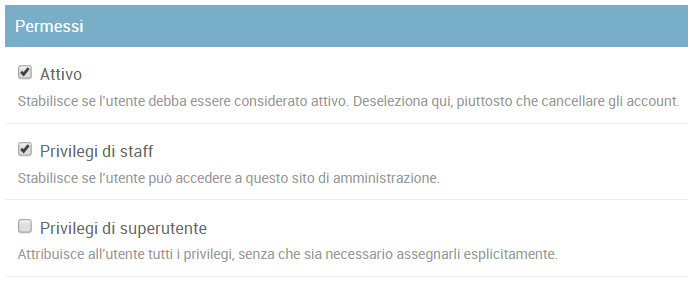
Questa tipologia di utente può effettuare il login con il pulsante “Accesso Staff” della pagina principale e inserire le proprie credenziali nella pagina di login, in modo da visualizzare i pulsanti “Modifica” e “Iscrizioni” che permettono di accedere alle rispettive pagine. È anche possibile disconnettersi tramite il pulsante “Logout” per tornare a visualizzare la pagina principale come un utente comune.

Fanno parte di questa categoria segretarie, docenti o altro personale a cui viene assegnato il compito di creare e modificare documenti.

### Amministratore

La persona incaricata per questo ruolo (probabilmente un docente) possiede tutti i diritti del personale, ma in più può accedere alla pagina di amministrazione, come descritto nel paragrafo 2.1.8.

Avrà come compito principale quello di cancellare i documenti non più utilizzati e di creare le credenziali per l’accesso del personale.

Durante la creazione degli utenti del personale dovrà assicurarsi di assegnare “Privilegi di Staff” tra i permessi, questo è necessario per permettere al personale di caricare e utilizzare le immagini presenti sul server durante la modifica dei documenti.

### Use Case Diagram

## Sintassi per la modifica dei documenti

Bisogna utilizzare questa sintassi durante la modifica del documento (all’interno dell’editor di testo nella parte sinistra della pagina di modifica) per inserirvi dei campi per l’inserimento di dati al suo interno.

### Tipi di campo

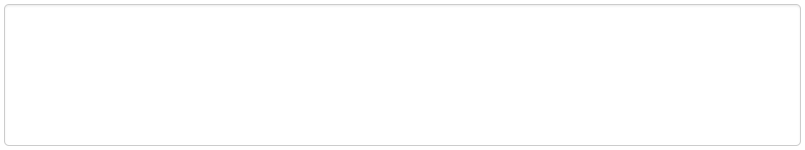
* Campo normale: almeno 7 trattini bassi



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Campo commento: almeno 80 trattini bassi, cioè quasi una riga intera (a carattere 14 o inferiore)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



All’interno dell’editor rimarranno visualizzati i trattini. Invece nella pagina di compilazione del documento verranno visualizzati i riquadri di inserimento, come è visibile nell’anteprima nella parte destra della pagina di modifica.

Per inserire i campi selezione , checkbox e radio button  è sufficiente premere il pulsante relativo nella barra dei comandi dell’editor.

### Nome dei campi

È possibile dare un nome ad ogni campo, che gli verrà assegnato all’interno del database, per poterne capire meglio il contenuto quando si andranno ad esaminare gli iscritti. È molto consigliato farlo, anche se in caso contrario ne verrà assegnato uno automaticamente. La sintassi da seguire è la seguente:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_[nome]

Non devono essere presenti spazi tra i trattini e la parentesi quadra.

Questa sintassi è valida sia per i campi normali che per quelli commento.

Sono permessi unicamente lettere e numeri, se sono presenti spazi o simboli il nome non verrà considerato. È consigliato, per maggiore chiarezza, utilizzare solo caratteri minuscoli per le lettere, anche se vengono accettati anche quelli maiuscoli.

La parentesi quadra e il suo contenuto non verranno visualizzati nella pagina di compilazione, così come nell’anteprima.

### Formato dei campi

È possibile assegnare un formato ad ogni campo, per poter aiutare l’utente nell’inserimento dei dati. Se non ne verrà assegnato nessuno, il formato sarà considerato di testo. La sintassi da seguire è la seguente:

\_\_\_\_\_\_\_\_\_\_\_[nome,formato]

Non devono essere presenti spazi tra la virgola e il nome del formato.

Questa sintassi è valida solo per i campi normali, non per quelli commento.

I formati permessi sono elencati di seguito:

* testo: è permesso l’inserimento di lettere maiuscole e minuscole, numeri, spazi e simboli.



\_\_\_\_\_\_\_\_\_\_\_[cognome,testo]

* numero: sono permessi solamente numeri.



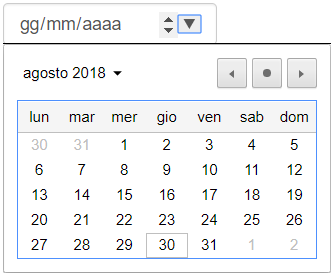
\_\_\_\_\_\_\_\_\_\_\_[anno,numero]

* data: possono essere inserite solamente date nel formato gg/mm/aaaa come indicato.



\_\_\_\_\_\_\_\_\_\_\_[compleanno,data]

È anche possibile selezionare una data grazie alla finestra

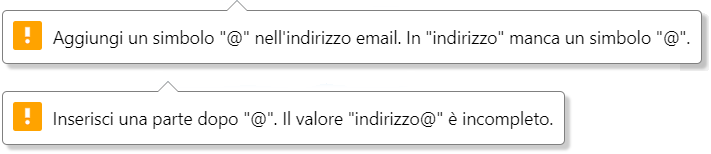


* email: può essere inserito solo testo contenente una @.



\_\_\_\_\_\_\_\_\_\_\_[posta,email]

In caso di formato errato saranno mostrati errori simili ai seguenti

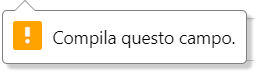


* obbligatorio: come il formato testo, ma non può essere lasciato vuoto, altrimenti verrà mostrato un errore.



\_\_\_\_\_\_\_\_\_\_\_[nome,obbligatorio]

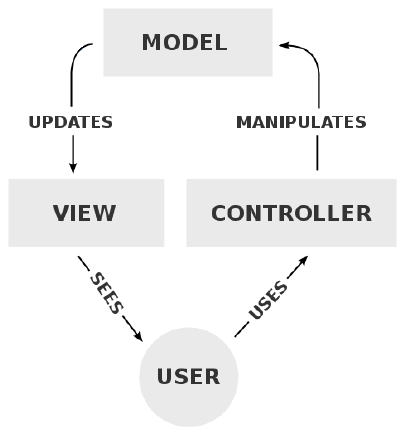
Errore emesso:



La parentesi quadra e il suo contenuto non verrà visualizzata nella pagina di compilazione, così come nell’anteprima.

# Organizzazione e implementazione del codice

Django utilizza lo stile architetturale Model-View-Controller, in grado di separare la logica di presentazione dei dati dalla logica di business.

Questo approccio separa gli elementi MVC in tre file diversi:

* models.py → model
* views.py → view
* urls.py → controller

## Model

Il model cattura il comportamento dell'applicazione in termini di dominio del problema, indipendentemente dall'interfaccia utente.

Esso astrae la rappresentazione dei dati e si occupa della loro gestione all’interno dell'applicazione.

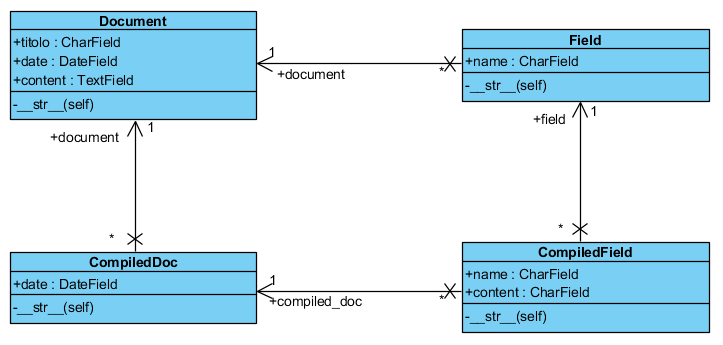
È implementato direttamente come collezione di classi Python che andranno a rappresentare le tabelle del database.

Si basa sul modello Object-Relational Database (ORD), che può essere inteso come una convergenza tra il modello relazionale e quello ad oggetti.

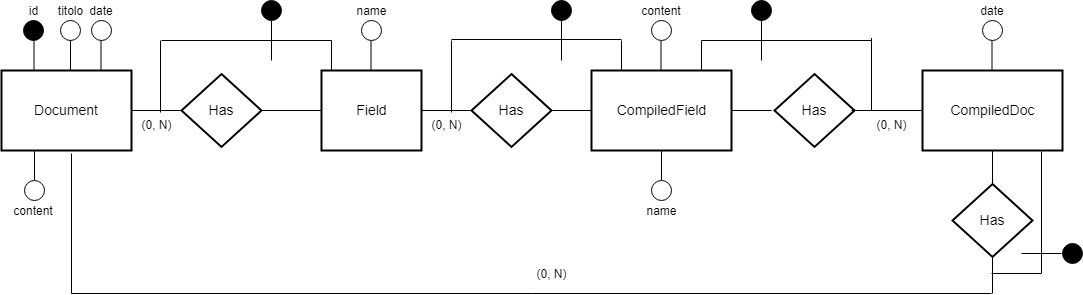
Lo strumento di modellazione privilegiato per questo modello è il diagramma delle classi UML, ma verrà mostrato per completezza anche lo schema ER.

### Class Diagram

Il diagramma delle classi UML viene usato per mostrare le classi che compongono il database con i loro rispettivi metodi e attributi.



### Entity Relationship Diagram

Rappresentazione alternativa del database utilizzando il modello ER. Questa rappresentazione mette in evidenza le Primary Key ed i vincoli, in particolare di Foreign Key, che legano le varie classi dell’applicazione.

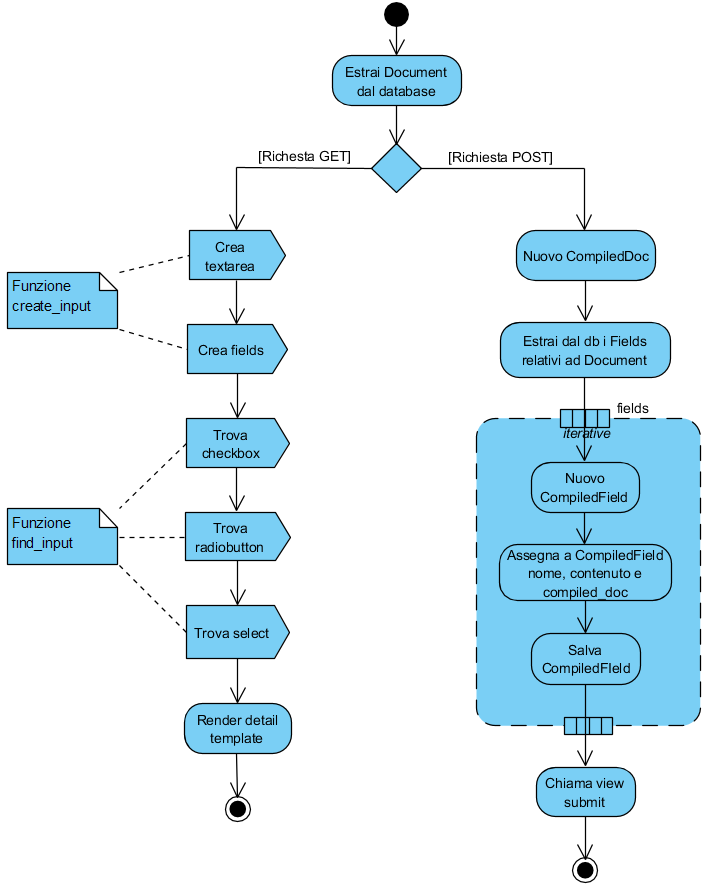
## View

La view ha il compito di mostrare lo stato del model. Può essere vista come l’interfaccia utente del model, infatti un solo model può avere un numero elevato di view diverse.

In Django il file view.py è formato da funzioni Python che si occupano dell’elaborazione e che fanno uso di template per la visualizzazione dei dati.

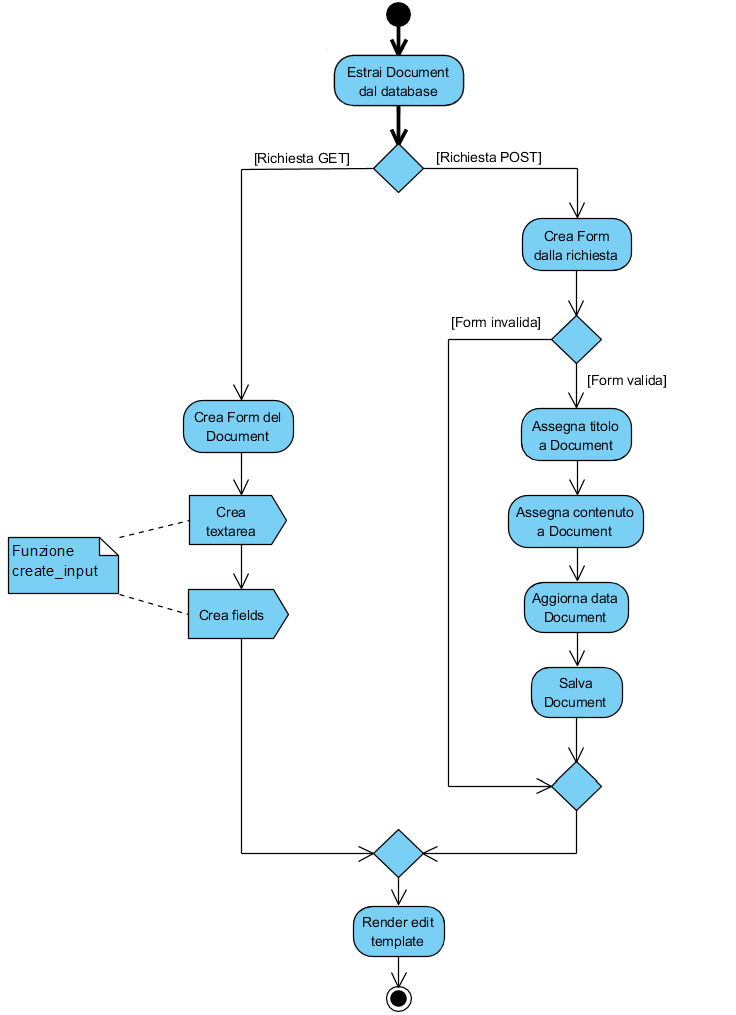
I template sono scritti in HTML e gli elementi grafici sono realizzati in linguaggio CSS con l’aiuto della libreria Bootstrap. Alcune funzionalità, sempre all’interno dei template, sono invece state realizzate con il linguaggio JavaScript.

### Detail view

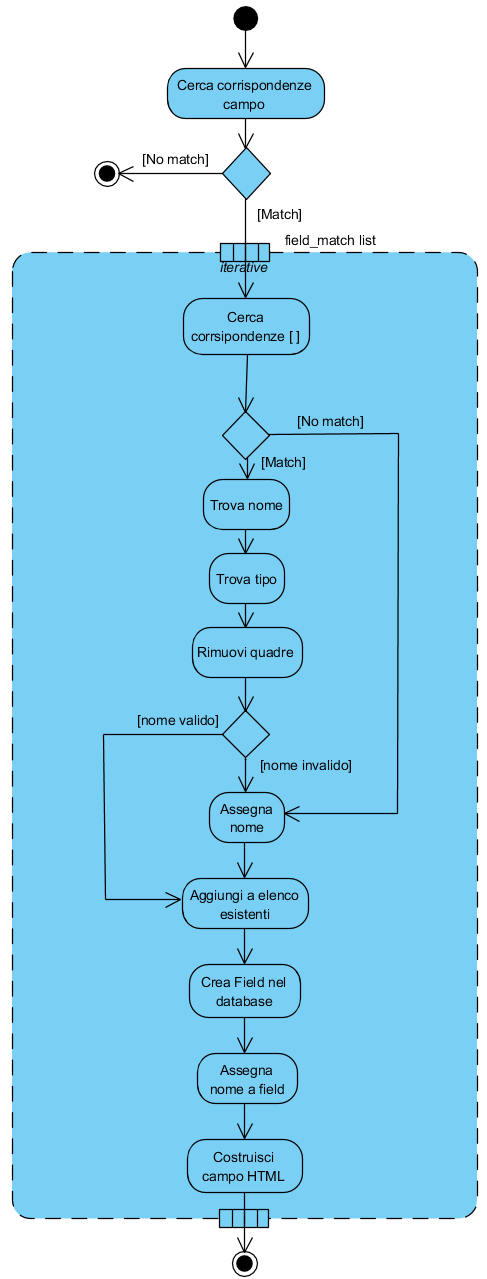
La detail view gestisce la visualizzazione della pagina di compilazione del documento ed è una delle view più interessanti, insieme alla edit view, in quanto trasforma gli underscore nei diversi tipi di campi HTML compilabili ed effettua il salvataggio dei documenti compilati. Viene qui mostrato il suo funzionamento   
con un activity diagram:

### Edit view

La edit view gestisce la visualizzazione della pagina di modifica del documento ed ha un funzionamento simile alla detail view, con le differenze che lavora sui Document e non sui CompiledDoc e non ha bisogno della funzione find\_input (spiegata nel paragrafo successivo).



### Funzione create\_input

La funzione create\_input lavora sul contenuto del documento. Essa contiene le procedure necessarie per la creazione dei campi compilabili, sia campi normali che campi commento con i relativi nomi e formati, che corrisponde alla funzione chiave per la realizzazione dell’intero progetto. La funzione find\_input è del tutto simile, a parte il fatto che non inserisce un campo HTML dato che lavora con i select, checkbox e radiobutton, in cui è già presente.

### JavaScript/jQuery

Con il linguaggio JavaScript e la libreria jQuery è stata realizzata la funzionalità di reindirizzamento automatico all’interno della pagina di conferma iscrizione (2.1.3) insieme al relativo timer di countdown:

**function** *countdown*() {  
 **var** timeleft = 5;  
 **var** downloadTimer = *setInterval*(**function**(){  
 timeleft--;  
 ***document***.getElementById(**"countdowntimer"**).**textContent** = timeleft.toString();  
 **if**(timeleft <= 0)  
 *clearInterval*(downloadTimer);  
 },1000);  
}  
**function** *redirect*(){  
 ***window***.**location**.**href** = **"**{% **url 'form:index'** %}**"**;  
}  
  
**$**(***document***).ready(**function**() {  
 *countdown*();  
 ***window***.setTimeout(*redirect*, 5000);  
});

### AJAX

Con AJAX sono state realizzate le funzioni di ricerca in tempo reale della pagina principale e del salvataggio automatico nella pagina di modifica.

* Ricerca in tempo reale:

**$**(***document***).ready(**function**() {  
 **$**(**"#ajaxsearch"**).on(**'keyup'**, **function**() {  
 **$**.ajax({  
 **type**: **"get"**,  
 **url**: **"**{% **url 'form:index'** %}**"**,  
 **data**: {**"search"**: **$**(**this**).val()},  
 success: **function**(data) {  
 **$**(**".largejumbo"**).html(data);  
 }  
 });  
 });  
});

* Salvataggio automatico:

**function** *getCookie*(name) {  
 **var** cookieValue = **null**;  
 **if** (***document***.**cookie** && ***document***.**cookie** !== **''**) {  
 **var** cookies = ***document***.**cookie**.split(**';'**);  
 **for** (**var** i = 0; i < cookies.**length**; i++) {  
 **var** cookie = **jQuery**.trim(cookies[i]);  
 *// Does this cookie string begin with the name we want?* **if** (cookie.substring(0, name.**length** + 1) === (name + **'='**)) {  
 cookieValue = *decodeURIComponent*(cookie.substring(name.**length** + 1));  
 **break**;  
 }  
 }  
 }  
 **return** cookieValue;  
}  
**function** *csrfSafeMethod*(method) {  
 *// these HTTP methods do not require CSRF protection* **return** (/^(GET|HEAD|OPTIONS|TRACE)$/.test(method));  
}  
  
**$**(***document***).ready(**function**() {  
 editor.on(**'change'**, **function**() {  
 **var** csrftoken = *getCookie*(**'csrftoken'**);  
 **$**.ajaxSetup({  
 beforeSend: **function**(xhr, settings) {  
 **if** (!*csrfSafeMethod*(settings.**type**) && !**this**.**crossDomain**) {  
 xhr.setRequestHeader(**"X-CSRFToken"**, csrftoken);  
 }  
 }  
 });  
 **$**.ajax({  
 **type**: **"post"**,  
 **url**: **"**{% **url 'form:edit' document**.**id** %}**"**,  
 **data**: {**"titolo"**: **$**(**"#ajaxtitle"**).val(),  
 **"content"**: editor.getData().toString()},  
 success: **function**(data) {  
 **$**(**"#modajax2"**).html(**$**(**'#modajax2'**, data).html());  
 }  
 });  
 });  
 **$**(**"#ajaxtitle"**).on(**'keyup'**, **function**() {  
 **var** csrftoken = *getCookie*(**'csrftoken'**);  
 **$**.ajaxSetup({  
 beforeSend: **function**(xhr, settings) {  
 **if** (!*csrfSafeMethod*(settings.**type**) && !**this**.**crossDomain**) {  
 xhr.setRequestHeader(**"X-CSRFToken"**, csrftoken);  
 }  
 }  
 });  
 **$**.ajax({  
 **type**: **"post"**,  
 **url**: **"**{% **url 'form:edit' document**.**id** %}**"**,  
 **data**: {**"titolo"**: **$**(**"#ajaxtitle"**).val(),  
 **"content"**: editor.getData()},  
 success: **function**(data) {  
 **$**(**".navbar-title"**).html(**$**(**'.navbar-title'**, data).html());  
 }  
 });  
 });  
});

## Controller

La terza parte, il controller, è il gestore di eventi che processano le interazioni con l'utente.

Può modificare il modello in risposta alle azioni dell'utente notificate dalla view e può richiedere aggiornamento delle view.

In Django è realizzato attraverso file urls.py, che ha l’obiettivo di mappare gli URL richiesti sulle opportune view Python che generano le pagine da mostrare all'utente.

### State Diagram

Nel seguente diagramma viene rappresentato il cambiamento di stato dell’applicazione dal punto di vista della pagina visualizzata (quindi della view attiva). I nomi degli stati saranno omonimi alle view e ad i relativi template.

