



DIETSPACE

TESINA PER IL PROGETTO DIETSPACE

Alessandro Bruna

Matr.165235

A.A 2023-2024

SOMMARIO

Introduzione	4
Overview del prodotto	4
Studio dei requisiti del prodotto	4
Estrapolazione dei requisiti	4
Analisi e specifica dei requisiti funzionali estrapolati	5
Area Messaggistica	7
Ricerca medici	8
Invio di richiesta cura	8
Diagrammi UML (Unified Modeling Language)	9
Use Case Diagram.....	9
Class diagram.....	10
Activity diagram.....	11
Creazione della dieta da parte del medico.	11
Visualizzazione della dieta da parte del paziente	11
State diagram	12
Stato dell'appuntamento	12
Screenshots Sito Web	13
Registrazione.....	13
Login	14
Home	14
Chats	15
Creazione Dieta.....	16
Testing.....	17
Test modello: Crea gli utenti li mette in relazione con l'entita Cura	17
Test Util: Creazione chat, invio e visualizzazione messaggi.	17
Tecnologie utilizzate	18
Django.....	18
JavaScript.....	18
AJAX	19
HTML.....	20
CSS.....	20

Introduzione

Il presente documento descrive lo sviluppo di un sito web per l'ambito medico-dietistico utilizzando il framework Django. Il sito web è stato sviluppato per l'esame di Tecnologie Web erogato dall'università di Modena e Reggio Emilia, tenuto dalla professoressa Claudia Canali e dal professore Nicola Capodieci dunque è un prodotto nato puramente a scopo didattico per la verifica delle conoscenze acquisite durante il corso sopra citato.

Nel seguente documento, verranno presentati i modelli dati utilizzati, le view e il codice JavaScript e AJAX implementato per migliorare l'esperienza utente. Saranno anche fornite informazioni sulle funzionalità del sito, i casi d'uso, il diagramma delle classi e alcuni diagrammi che descrivono le attività o lo stato di entità (per una questione di semplicità non verranno rappresentati tutti i requisiti sia nella specifica che nella rappresentazione mediante i diagrammi).

Overview del prodotto

DietSpace nasce come applicazione desktop di supporto per il campo della nutrizione medica, pensata per facilitare la gestione e il monitoraggio dei pazienti da parte dei medici, oltre a proporsi come strumento di ricerca e consultazione medica per i pazienti stessi. L'applicazione dispone di un'interfaccia intuitiva che facilita l'utilizzo permettendo un'esperienza utente generale gradevole.

Studio dei requisiti del prodotto

Estrapolazione dei requisiti

Si è deciso di sviluppare questo step intervistando, su base volontaria, un insieme di futuri utilizzatori con il fine di comprendere le funzionalità desiderate (requisiti di basso livello).

Il colloquio è stato effettuato da un'analista dei requisiti. Si è anche studiato il livello medio di competenze possedute sia da parte dei pazienti che da parte del medico così da poter creare un sistema dall'utilizzo facile e intuitivo.

Affinché possa essere appresa la linea generale di quello che si è chiesto durante i colloqui, vengono sotto riportate le tre domande principali e le relative risposte più comuni suddivise per medici e pazienti.

DOMANDA	RISPOSTA MEDICO	RISPOSTA PAZIENTE
1. Quali sono i comportamenti o le caratteristiche che il sito deve avere?	<p>Vorrei avere un'area medico (autenticazione e aggiornamenti relativi ai propri dati, che vengono mostrati a pazienti nella ricerca)</p> <p>Vorrei avere un'area integrata per la creazione di diete</p> <p>Area messaggi, in cui poter comunicare facilmente con i propri pazienti</p> <p>Area andamento pazienti, dove poter visualizzare e aggiornare l'andamento del paziente.</p> <p>Gestione appuntamenti</p>	<p>Vorrei avere un'area paziente (autenticazione e aggiornamenti relativi ai propri dati)</p> <p>Poter cercare i medici disponibili, consultare le informazioni e inviare una richiesta per l'aggiunta alla lista pazienti</p> <p>Mi piacerebbe avere un'area messaggi con cui comunicare più velocemente con il medico.</p> <p>Vorrei poter fissare il mio appuntamento in autonomia.</p>
2. Su che dispositivo preferiresti utilizzarlo?	Preferirei un utilizzo attraverso il browser	Mi piacerebbe poter utilizzare un sito.

TABELLA 1 ESTRAPOLAZIONE DEI REQUISITI DI BASSO LIVELLO

Analisi e specifica dei requisiti funzionali estrapolati

Una volta aver concluso la fase precedente, si è passati alla fase di analisi dei requisiti, cioè alla categorizzazione e prioritizzazione degli stessi per migliorare la successiva progettazione e implementazione.

Di seguito vengono mostrati i requisiti del sito web, riportando una M (medico) o una P (paziente), per specificare gli utenti interessati per ogni requisito richiesto.

- 1) Area Personale
 - a) Registrazione utente M/P
 - b) Autenticazione utente M/P
 - c) Aggiornamento dati personali utente M/P
 - d) Caricamento immagine del profilo M/P
 - e) Aggiunta e modifica di uno studio medico M
- 2) Area Chat
 - a) Invio, ricezione e visualizzazioni dei messaggi M/P
- 3) Ricerca medici
 - a) Visualizzazione medici disponibili/curanti P
 - b) Invio richiesta a medico P
- 4) Gestione Appuntamenti
 - a) Prenotazione appuntamento P
 - b) Modifica data/ora P
 - c) Elimina appuntamento M
 - d) Calendario appuntamenti M/P
- 5) Creazione e visualizzazione dieta
 - a) Creazione della dieta M (la creazione della dieta include l'aggiunta di pasti con alimenti nei giorni della settimana)
 - b) Modifica dieta M
 - c) Elimina Dieta M
 - d) Visualizzazione dieta M/P
- 6) Andamento Paziente
 - a) Visualizzazione andamento peso del paziente tramite grafico M/P
 - b) Aggiunta/Rimozione di un peso del paziente M

Per specificare i requisiti funzionali sono state utilizzate delle tabelle di input-output. Le tabelle si compongono di vari campi e contengono le info relative alle funzionalità che l'applicazione deve soddisfare e come deve rispondere agli input immessi dall'utente.

Area Messaggistica

Requisito	<i>Visualizzazione messaggi ricevuti</i>
Descrizione	Il sistema deve consentire agli utenti di visualizzare un elenco di tutti i messaggi ricevuti, includendo informazioni essenziali come mittente, oggetto e data di ricezione.
Utente utilizzatore	Medico e paziente
Input	L'utente clicca sull'icona di messaggistica, selezionando la chat desiderata
Processo	L'applicazione deve ricavare i messaggi ricevuti e inviati (con relativo id) per l'utente
Output	L'applicazione deve mostrare i messaggi ricevuti ed inviati con il relativo contenuto.

Requisito	<i>Invio dei messaggi</i>
Descrizione	Il sistema deve consentire agli utenti di inviare messaggi agli altri utenti.
Utente utilizzatore	Medico e paziente
Input	L'utente clicca sulla sezione "invia un messaggio" e seleziona la chat in cui vuole mandare un messaggio.
Processo	L'applicazione apre l'interfaccia per l'invio di un messaggio mostrando il campo di testo e il bottone d'invio.
Output	Invio del messaggio specificato verso l'utente destinatario inserito nell'apposito campo.

Ricerca medici

Requisito	<i>Ricerca medici</i>
Descrizione	Il sistema deve consentire agli utenti pazienti di ricercare eventuali medici a cui affidarsi.
Utente utilizzatore	Paziente
Input	L'utente naviga la sezione di ricerca dottori cliccando l'icona nella home
Processo	L'applicazione cerca nel database associato i medici e mostra il risultato in forma tabellare.
Output	L'applicazione mostra i medici disponibili.

Invio di richiesta cura

Requisito	<i>Invio richiesta a medico</i>
Descrizione	L'applicazione deve inoltrare al medico la richiesta del paziente di essere aggiunto alla lista dei pazienti che ha in cura.
Utente utilizzatore	Paziente
Input	L'utente invia una richiesta cliccando l'apposito bottone.
Processo	L'applicazione invia una richiesta al medico aggiungendola alla sua lista delle richieste
Output	L'applicazione deve inoltrare le richieste ai medici.

Diagrammi UML (Unified Modeling Language)

Per una comprensione più approfondita dell'applicazione e del suo utilizzo, si presentano di seguito i diagrammi UML (Unified Modeling Language) relativi a DietSpace. Alcuni di questi diagrammi si concentrano, per motivi di semplicità, solo su scenari specifici.

Use Case Diagram

Il diagramma Use Case rappresenta graficamente i requisiti funzionali che il prodotto deve soddisfare e gli attori coinvolti. Nella Figura 1 è mostrato il class diagram per il sito web analizzato.

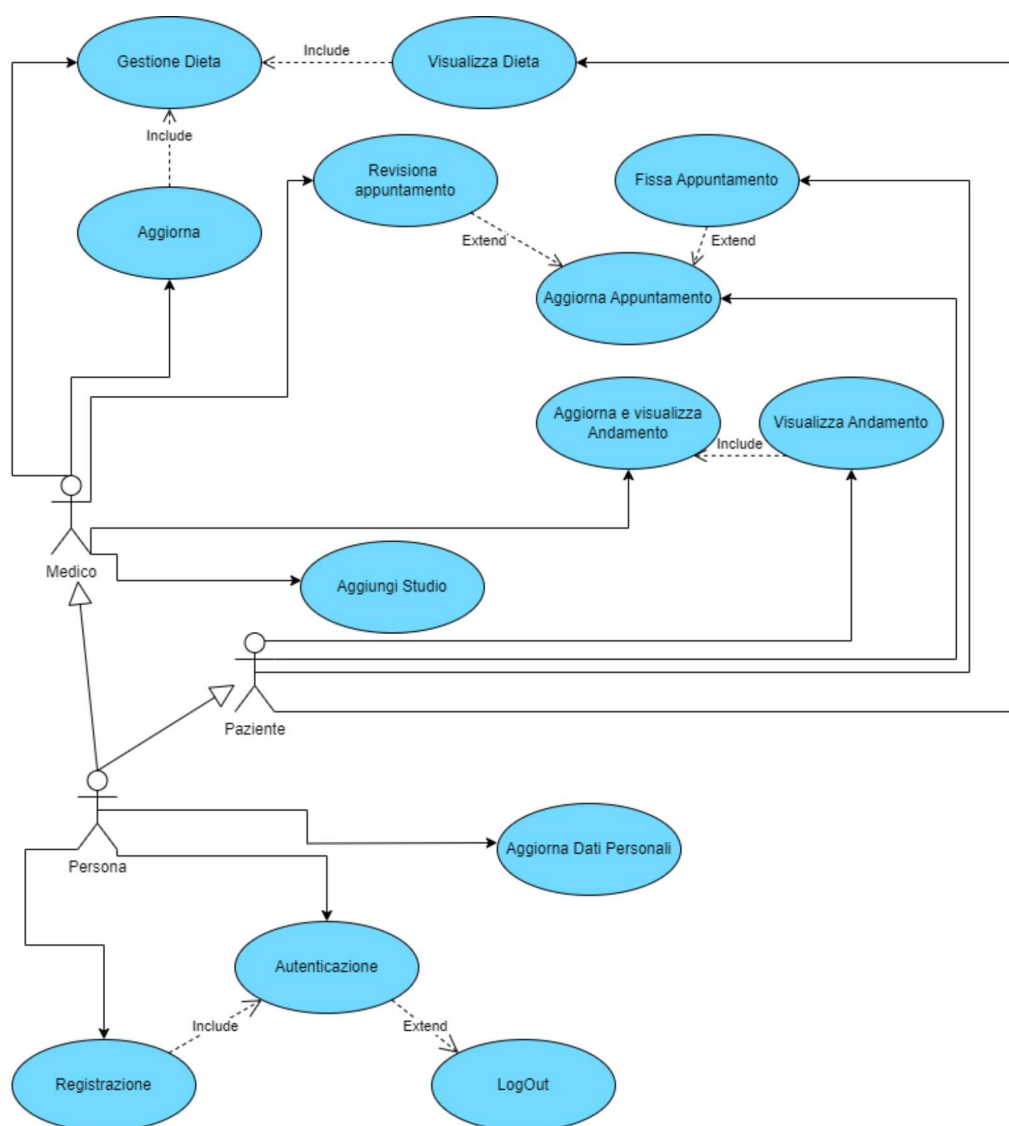


FIGURA 1 USE CASE DIAGRAM

Nel diagramma i casi d'uso relativi alla gestione includono le operazioni di modifica, eliminazione e aggiunta relative che dovrebbero essere inserite tramite altri use case con frecce di extends per semplicità non riportate.

Class diagram

Il diagramma delle classi descrive le relazioni tra le entità. Nella Figura 2 vengono mostrate le classi che compongono DietSpace e le loro relazioni.

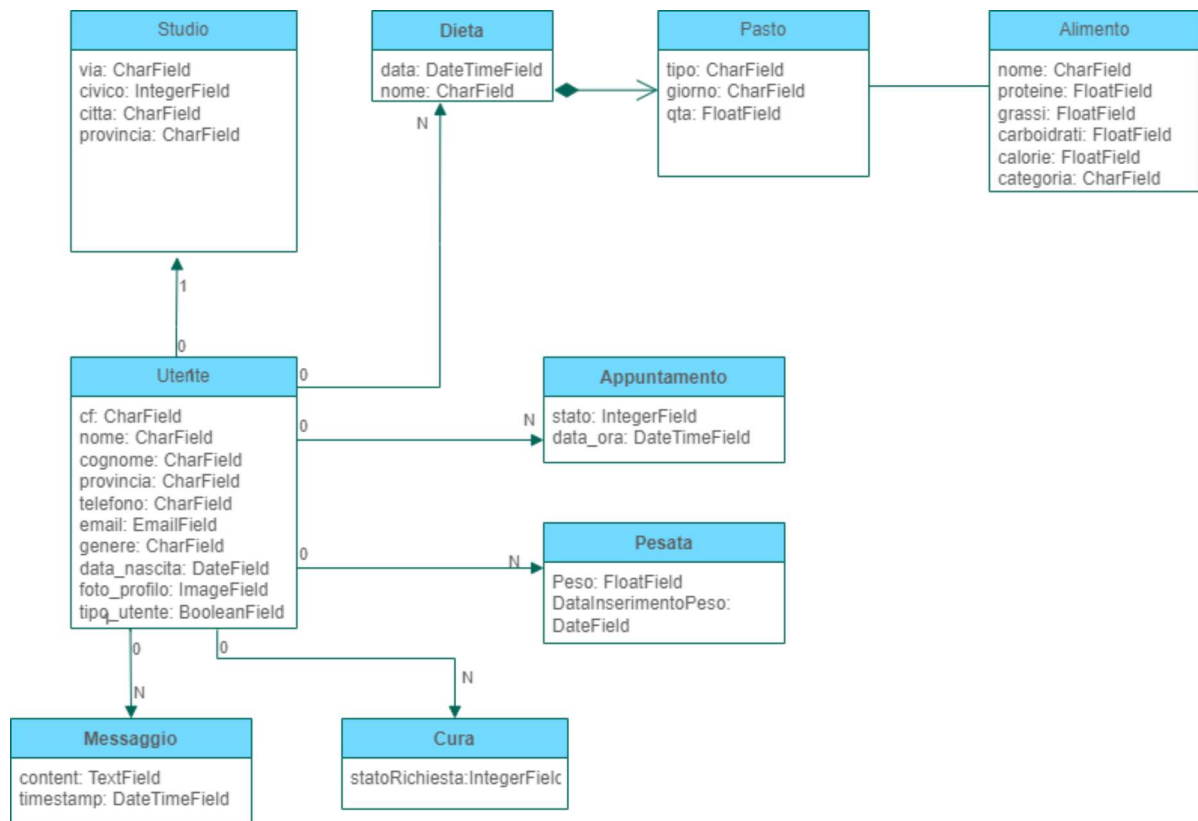


FIGURA 2 DIAGRAMMA DELLE CLASSI

Activity diagram

Il diagramma delle attività descrive l'aspetto dinamico del sistema.

Creazione della dieta da parte del medico.

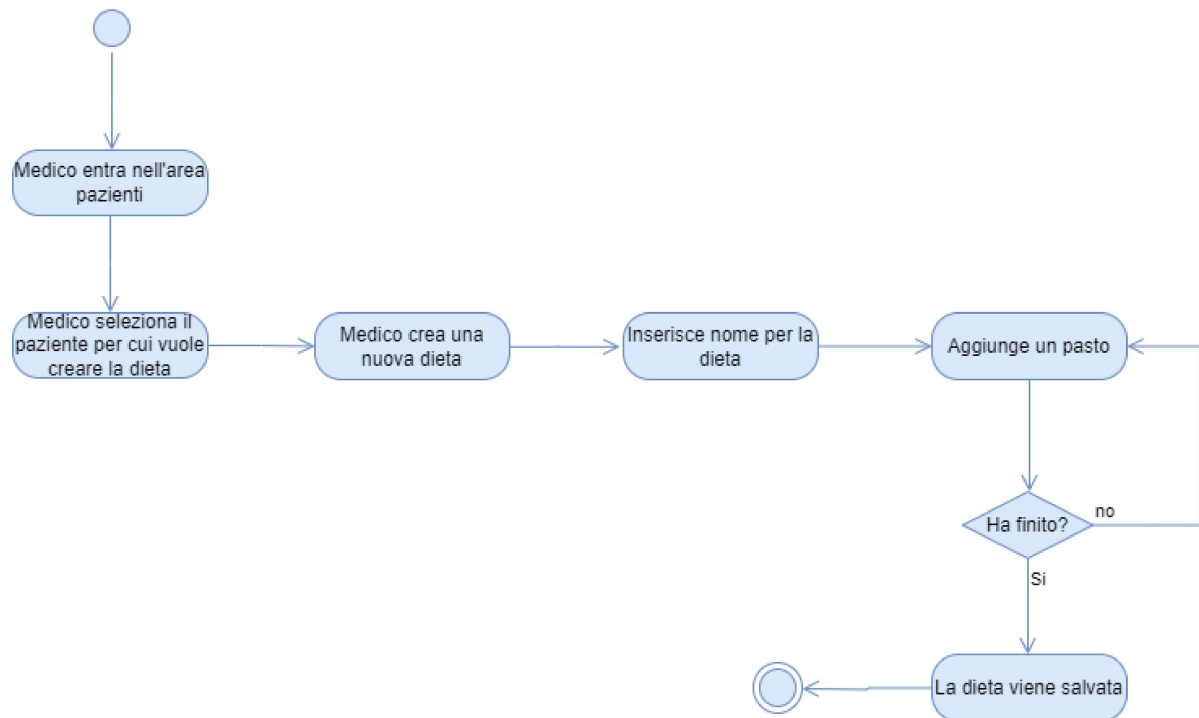


FIGURA 3 ACTIVITY DIAGRAM CREAZIONE DIETA

Visualizzazione della dieta da parte del paziente

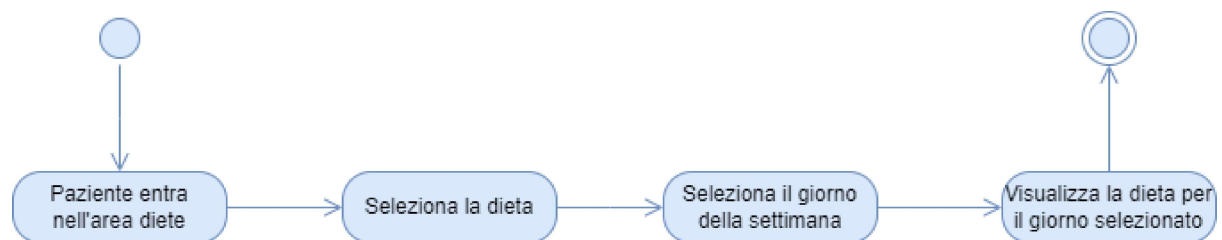


FIGURA 4 ACTIVITY DIAGRAM VISUALIZZAZIONE DIETA

Nella Figura 4 viene rappresentato l'activity diagram inerente alla visualizzazione della dieta da un paziente.

State diagram

Il diagramma degli stati descrive gli stati in cui si trovano le entità durante l'esecuzione di una certa funzionalità. In seguito, si riporta il diagramma degli stati in cui l'entità appuntamento può trovarsi in: elaborazione, accettato, disdetto.

Stato dell'appuntamento

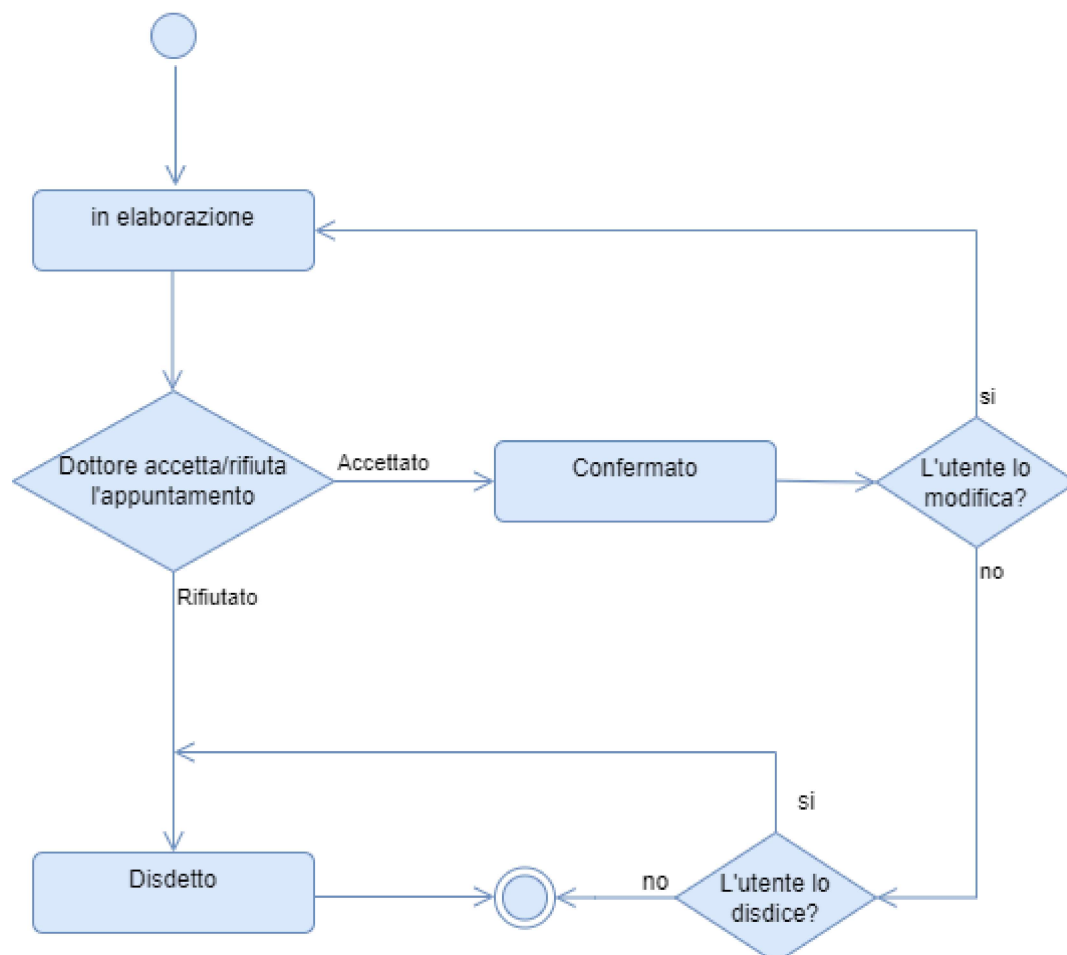


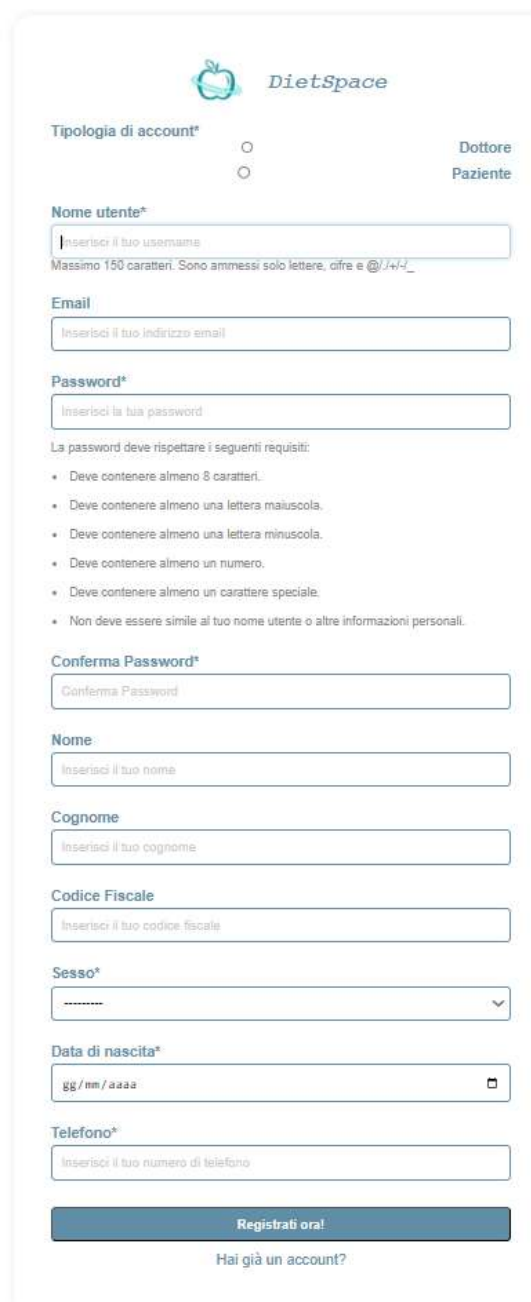
FIGURA 5 STATE DIAGRAM APPUNTAMENTO

Screenshots Sito Web

Per motivi di lunghezza non vengono riportati tutti gli screenshots del sito web ma solo quelli ritenuti salienti.

Registrazione

L'immagine seguente rappresenta l'interfaccia di registrazione di un utente tramite un form da compilare con i propri dati. In questa fase è necessario selezionare il tipo di account desiderato: medico o paziente.



The screenshot displays the registration interface for DietSpace. At the top, the DietSpace logo (an apple icon) and the brand name are visible. Below this, the 'Tipologia di account*' section allows users to select between 'Dottore' and 'Paziente' using radio buttons. The form then proceeds to several input fields: 'Nome utente*' (with a placeholder 'Inserisci il tuo username' and a note about character limits), 'Email' (placeholder 'Inserisci il tuo indirizzo email'), 'Password*' (placeholder 'Inserisci la tua password' and a list of password requirements), 'Conferma Password*' (placeholder 'Conferma Password'), 'Nome' (placeholder 'Inserisci il tuo nome'), 'Cognome' (placeholder 'Inserisci il tuo cognome'), 'Codice Fiscale' (placeholder 'Inserisci il tuo codice fiscale'), 'Sesso*' (a dropdown menu), 'Data di nascita*' (placeholder 'gg/mm/aaaa' with a calendar icon), and 'Telefono*' (placeholder 'Inserisci il tuo numero di telefono'). A prominent blue button labeled 'Registrati ora!' is positioned at the bottom of the form, with a link 'Hai già un account?' located directly beneath it.

Tipologia di account*

☐ Dottore
☐ Paziente

Nome utente*

Inserisci il tuo username

Massimo 150 caratteri. Sono ammessi solo lettere, cifre e @/+/!/_

Email

Inserisci il tuo indirizzo email

Password*

Inserisci la tua password

La password deve rispettare i seguenti requisiti:

- Deve contenere almeno 8 caratteri.
- Deve contenere almeno una lettera maiuscola.
- Deve contenere almeno una lettera minuscola.
- Deve contenere almeno un numero.
- Deve contenere almeno un carattere speciale.
- Non deve essere simile al tuo nome utente o altre informazioni personali.

Conferma Password*

Conferma Password

Nome

Inserisci il tuo nome

Cognome

Inserisci il tuo cognome

Codice Fiscale

Inserisci il tuo codice fiscale

Sesso*

.....

Data di nascita*

gg/mm/aaaa

Telefono*

Inserisci il tuo numero di telefono

Registrati ora!

[Hai già un account?](#)

FIGURA 6 INTERFACCIA DI REGISTRAZIONE

Login

La pagina di login implementa un meccanismo di riconoscimento automatico per verificare se l'utente che tenta di accedere è un paziente o un medico. In caso di "password dimenticata" è possibile avviare le procedure di ripristino cliccando sul link in basso.

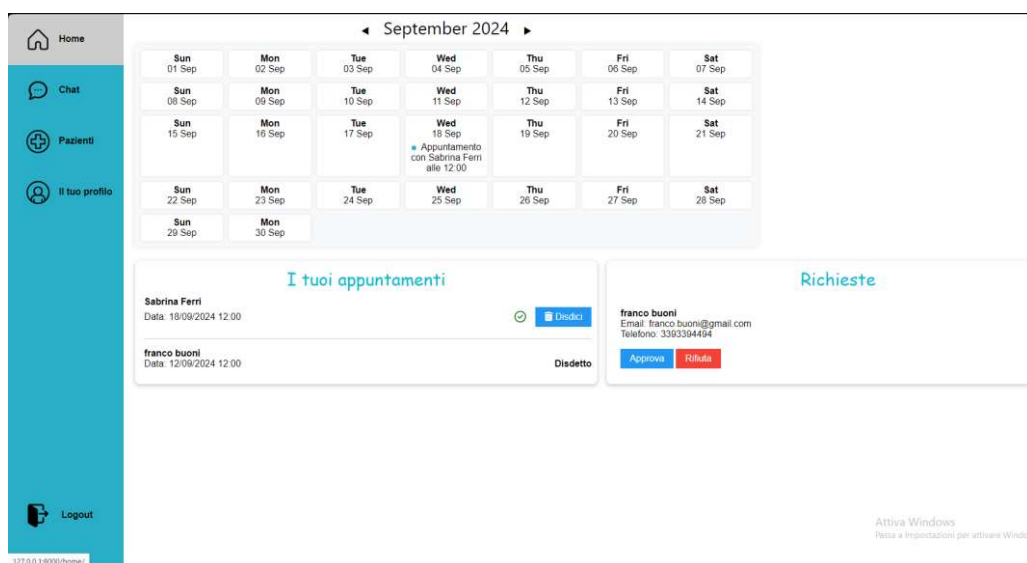


The login form for DietSpace features a logo at the top center consisting of a green apple icon and the text "DietSpace". Below the logo, there are two input fields: "Username*" and "Password*", each with a placeholder text of the same name. A blue "Login" button is positioned below the password field. At the bottom of the form, there are two links: "Non hai un account? Clicca qui!" and "Hai dimenticato la password? Clicca qui!".

FIGURA 7 INTERFACCIA DI LOGIN

Home

La home è diversificata per tipo di utente, in entrambi i casi vengono riportati il calendario con gli appuntamenti, la lista degli appuntamenti con le possibili azioni che si possono eseguire in base al tipo di account, l'icona per visualizzare e modificare il profilo e l'icona chat per entrare nell'area messaggi. Nel riquadro "Richieste" vengono riportate, per il paziente le richieste che ha inviato ai medici, e per il dottore le richieste ricevute con i relativi bottoni per accettarle o rifiutarle. Per il paziente è presente l'area per la visualizzazione dell'andamento del peso attraverso un grafico



The home interface for DietSpace is divided into several sections. On the left is a vertical sidebar with a blue background containing icons and labels for "Home", "Chat", "Pazienti", "Il tuo profilo", and "Logout". The main content area at the top features a calendar for "September 2024" with dates from Sun 01 Sep to Sat 30 Sep. A specific appointment is highlighted on Wednesday, September 18th: "Appuntamento con Sabrina Ferri alle 12:00". Below the calendar, there are two main sections: "I tuoi appuntamenti" and "Richieste". The "I tuoi appuntamenti" section lists two appointments: one with "Sabrina Ferri" on 18/09/2024 at 12:00 with a "Disdetta" button, and another with "franco buoni" on 12/09/2024 at 12:00 with a "Disdetta" button. The "Richieste" section shows a request from "franco buoni" with contact information (Email: franco.buoni@gmail.com, Telefono: 3393394494) and buttons for "Approva" and "Rifiuta". At the bottom right, there is a small "Attiva Windows" watermark.

FIGURA 8 INTERFACCIA HOME

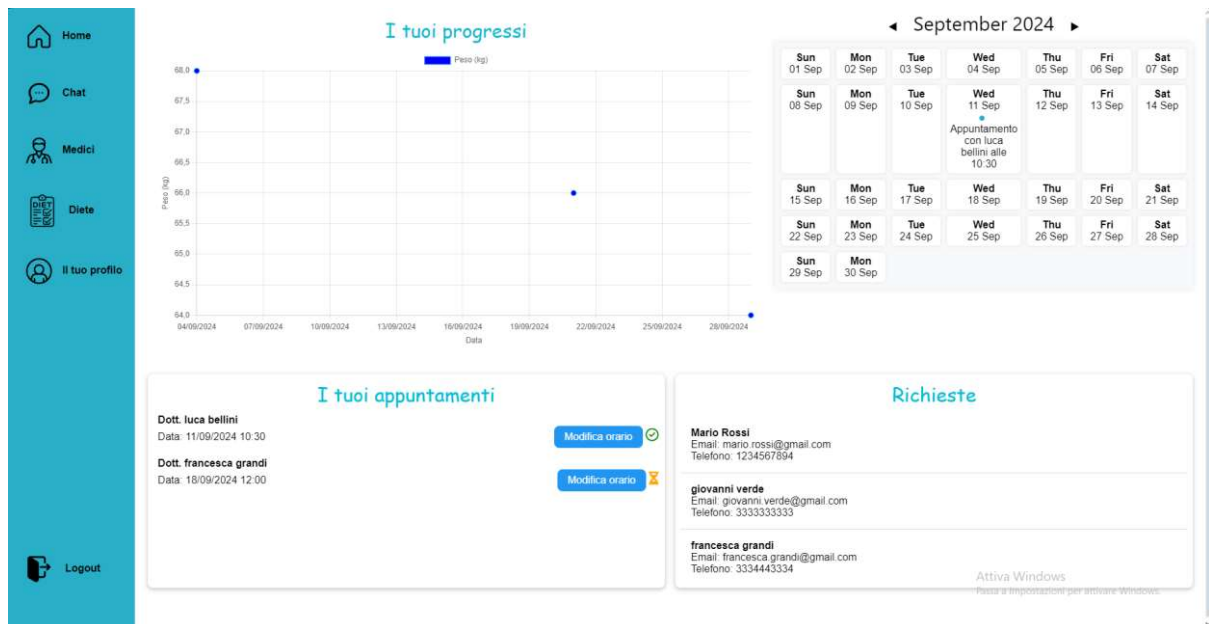


FIGURA 9 INTERFACCIA HOME PAZIENTE

Chats

L'interfaccia seguente mostra la chat in cui sono presenti le chat esistenti a sinistra che quando vengono cliccate aprono la chat con il secondo utente partecipante.

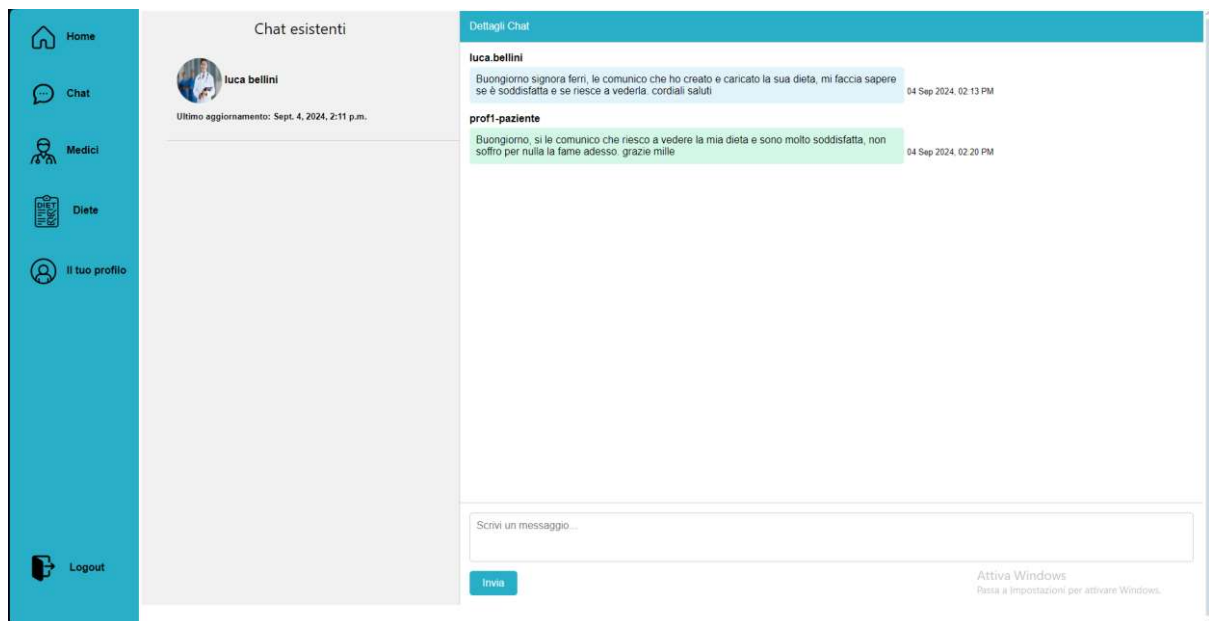


FIGURA 10 INTERFACCIA CHAT

Creazione Dieta

Nelle immagini seguenti sono mostrate le interfacce proposte al medico per la creazione delle diete. È possibile selezionare il giorno, attraverso il menù a tendina, che si vuole compilare e tramite il tasto “+” aggiungere un nuovo alimento. Alla pressione del tasto viene mostrata una nuova finestra in cui è possibile cercare, selezionare e inserire la quantità desiderata dell’alimento che si vuole aggiungere per quel pasto. Una volta aggiunto, per eliminare un alimento bisogna premere sul tasto con l’icona “cestino”.

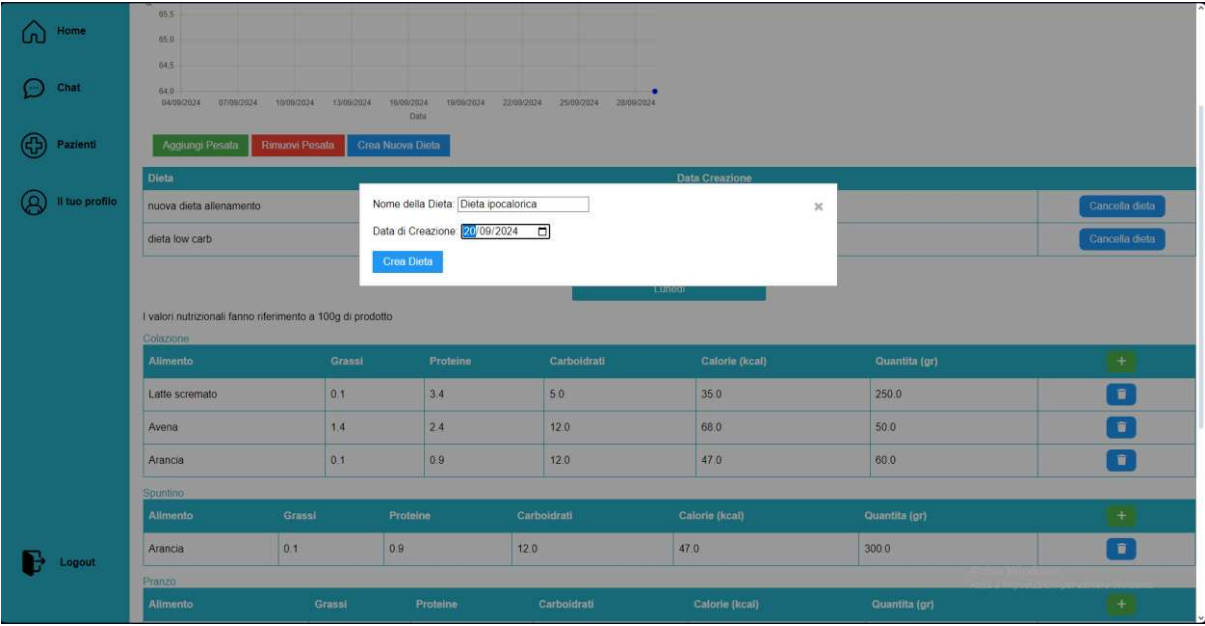


FIGURA 4 CREAZIONE DIETA

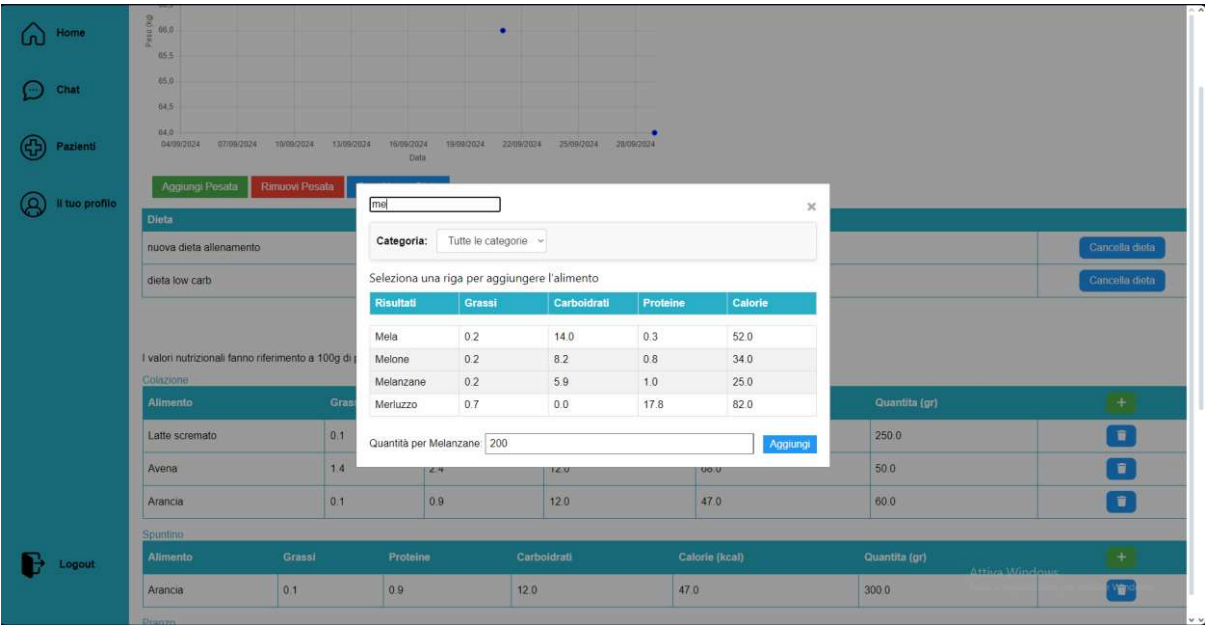


FIGURA 5 AGGIUNTA PASTO

Testing

Di seguito vengono riportati dei test che sono stati sviluppati per la creazione dell'utente e per la chat.

Test modello: Crea gli utenti li mette in relazione con l'entità Cura

```
class UtenteModelTestCase(TestCase):
    def setUp(self):
        self.dottore = Utente.objects.create_user(
            username='dottore', email='dottore@example.com', password='password', tipo_utente=True, # Dottore
            cf='DITDR170A01F205Z', nome='Mario', cognome='Rossi', provincia='RM', telefono='3331234567', genere='M', data_nascita='1970-01-01', foto_profilo='avatar-dottore.png'
        )
        self.paziente = Utente.objects.create_user(
            username='paziente', email='paziente@example.com', password='password', tipo_utente=False, cf='PZNTST80A01F205Z',
            nome='Luigi', cognome='Bianchi', provincia='MI', telefono='3337654321', genere='M', data_nascita='1980-01-01', foto_profilo='avatar-paziente.png'
        )

    def test_creazione_utenti(self):
        self.assertTrue(self.dottore.tipo_utente)
        self.assertFalse(self.paziente.tipo_utente)

    def test_creazione_cura(self):
        cura = Cura.objects.create(
            idDottore=self.dottore,
            idPaziente=self.paziente,
            statoRichiesta=0
        )
        self.assertEqual(cura.idDottore, self.dottore)
        self.assertEqual(cura.idPaziente, self.paziente)

    def test_cura_con_utenti_invalidi(self):
        self.paziente.tipo_utente = True
        self.paziente.save()
        with self.assertRaises(ValidationError):
            cura = Cura(idDottore=self.dottore, idPaziente=self.paziente)
            cura.clean()

        self.dottore.tipo_utente = False
        self.dottore.save()
        with self.assertRaises(ValidationError):
            cura = Cura(idDottore=self.dottore, idPaziente=self.paziente)
            cura.clean()
```

Test Util: Creazione chat, invio e visualizzazione messaggi.

```
class ChatMessageTestCase(TestCase):
    def setUp(self):
        self.utente1 = Utente.objects.create_user(
            username='dottore', email='dottore@example.com', password='password', tipo_utente=True, # Dottore
            cf='DITDR170A01F205Z', nome='Mario', cognome='Rossi', provincia='RM', telefono='3331234567', genere='M', data_nascita='1970-01-01', foto_profilo='avatar-dottore.png'
        )
        self.utente2 = Utente.objects.create_user(
            username='paziente', email='paziente@example.com', password='password', tipo_utente=False, # Paziente
            cf='PZNTST80A01F205Z', nome='Luigi', cognome='Bianchi', provincia='MI', telefono='3337654321', genere='M', data_nascita='1980-01-01', foto_profilo='avatar-paziente.png'
        )
        self.chat = Chat.objects.create()
        self.chat.participants.set([self.utente1, self.utente2])

    def test_creazione_chat(self):
        self.assertIn(self.utente1, self.chat.participants.all())
        self.assertIn(self.utente2, self.chat.participants.all())

    def test_creazione_message(self):
        messaggio = Message.objects.create(chat=self.chat, sender=self.utente1, content="Ciao!")
        self.assertEqual(messaggio.chat, self.chat)
        self.assertEqual(messaggio.sender, self.utente1)
        self.assertEqual(messaggio.content, "Ciao!")

    def test_visualizzazione_messaggio(self):
        messaggio = Message.objects.create(chat=self.chat, sender=self.utente1, content="Ciao!")
        serialized_message = messaggio.serialize()
        self.assertEqual(serialized_message['content'], "Ciao!")
        self.assertEqual(serialized_message['sender']['username'], self.utente1.username)
```

Tecnologie utilizzate

Django

Django è un framework di sviluppo web lato server per Python che consente di creare applicazioni web robuste e scalabili in modo rapido ed efficiente. Django gestisce molte delle complessità del backend, come la gestione dei dati, la sicurezza e la scalabilità, permettendo agli sviluppatori di concentrarsi sulla costruzione di funzionalità applicative.

In un sito web sviluppato con Django, il framework è utilizzato per gestire l'interazione con il database, l'autenticazione degli utenti, la gestione delle sessioni e il routing delle richieste HTTP. Django genera anche i file HTML da inviare al client ma potrebbero servire dei file CSS e JavaScript per creare un'esperienza utente completa. Ad esempio, nel contesto del progetto descritto, Django è stato utilizzato per gestire le relazioni tra dottori e pazienti, gestire gli appuntamenti e conservare i dati relativi alla dieta degli utenti e ai messaggi.

JavaScript

JavaScript è un linguaggio di programmazione lato client utilizzato per creare contenuti web interattivi e dinamici. Grazie alla sua capacità di manipolare il DOM (Document Object Model) e gestire eventi, JavaScript permette agli sviluppatori di rendere le pagine web più interattive, rispondendo in tempo reale alle azioni degli utenti senza la necessità di ricaricare la pagina.

Nel contesto di un sito web sviluppato con Django, JavaScript è utilizzato per migliorare l'esperienza utente attraverso funzionalità interattive come la convalida dei form lato client, il filtraggio dei contenuti in tempo reale, e la gestione delle animazioni e degli effetti visivi. Ad esempio, JavaScript è stato impiegato per creare una funzione di ricerca in tempo reale che mostra i risultati dei cibi cercati filtrando gli elementi di una lista man mano che l'utente digita, migliorando così l'usabilità e la velocità dell'applicazione.

```

function filterTableAlimento() {
    const categoryFilter = document.getElementById('categoryFilter').value.toLowerCase();
    const searchInput = document.getElementById('search').value.toLowerCase();

    const rows = document.getElementById('alimentiTbody').getElementsByTagName('tr');

    for (let i = 0; i < rows.length; i++) {
        const firstCell = rows[i].getElementsByTagName('td')[0]; // Prima cella (nome dell'alimento)
        const categoryCell = rows[i].getAttribute('data-categoria'); // Categoria dell'alimento

        if (firstCell) {
            const txtValue = firstCell.textContent || firstCell.innerText;

            // Controlla se il valore cercato è contenuto nel testo della cella
            const matchesSearch = txtValue.toLowerCase().includes(searchInput);

            // Controlla se la categoria corrisponde al filtro selezionato
            const matchesCategory = categoryFilter === "" || categoryCell.toLowerCase() === categoryFilter;

            // Mostra o nasconde la riga in base ai criteri di ricerca e categoria
            rows[i].style.display = (matchesSearch && matchesCategory) ? '' : 'none';
        }
    }
}

```

AJAX

AJAX (Asynchronous JavaScript And XML) è una tecnologia di sviluppo web che consente di aggiornare le pagine web in modo asincrono, senza dover ricaricare l'intera pagina. AJAX utilizza una combinazione di JavaScript e XML (o altri formati dati come JSON) per inviare richieste al server in background e ricevere i dati richiesti senza interrompere l'interazione dell'utente con la pagina. Ciò consente di creare un'esperienza utente più fluida e reattiva, in cui le modifiche e gli aggiornamenti possono essere applicati dinamicamente senza dover ricaricare completamente la pagina.

Nel contesto del sito web sviluppato con Django, AJAX è stato utilizzato per migliorare l'interattività e l'efficienza dell'applicazione. Ad esempio, le richieste AJAX sono state utilizzate per consentire agli utenti di inviare messaggi, aggiungere i pasti alle tabelle che formano la dieta senza dover ricaricare l'intera pagina. In questo modo, è possibile ottenere risposte immediate e visualizzare le modifiche apportate senza interruzioni o ritardi nell'esperienza utente.

L'immagine di seguito rappresenta la funzione che permette di caricare i messaggi della chat senza ricaricare l'intera pagina.

```

function loadChat(chatId) {
  document.getElementById('chat_id').value = chatId;

  fetch(`/messages/${chatId}/`, {
    headers: {
      'X-Requested-With': 'XMLHttpRequest'
    }
  })
  .then(response => response.json())
  .then(data => {
    const messagesContainer = document.getElementById('messages');
    messagesContainer.innerHTML = '';
    data.messages.forEach(message => {
      const messageElement = document.createElement('div');
      messageElement.classList.add('message');

      const messageClass = message.sender.username === `${request.user.username}` ? 'sent' : 'received';
      messageElement.classList.add(messageClass);

      messageElement.innerHTML = `
        <div class="sender">${message.sender.username}</div>
        <div class="text">${message.content}</div>
        <small>${message.timestamp}</small>
      `;
      messagesContainer.appendChild(messageElement);
    });
    messagesContainer.scrollTop = messagesContainer.scrollHeight;
  })
  .catch(error => console.error('Errore:', error));
}

```

HTML

HTML (HyperText Markup Language) è il linguaggio di markup utilizzato per creare la struttura di base delle pagine web. In un progetto Django, HTML definisce l'ossatura delle pagine, permettendo di inserire contenuti dinamici generati dal backend. Ad esempio, viene utilizzato per costruire le pagine di login e registrazione, dove Django inserisce automaticamente i dati degli utenti e i messaggi di errore.

CSS

CSS (Cascading Style Sheets) è il linguaggio usato per controllare l'aspetto delle pagine web, inclusi layout, colori e font. In un'applicazione Django, CSS è utilizzato per migliorare la presentazione delle pagine, rendendole più esteticamente gradevoli e usabili. Viene impiegato per stilizzare le tabelle dei pasti nella dieta, migliorando la leggibilità e l'aspetto visivo. Nel progetto questi sono i files css utilizzati:

```

appointment.css
chat.css
diet.css
doctors.css
modals.css
styles.css
x.css

```

Presenti nella cartella “static/css”