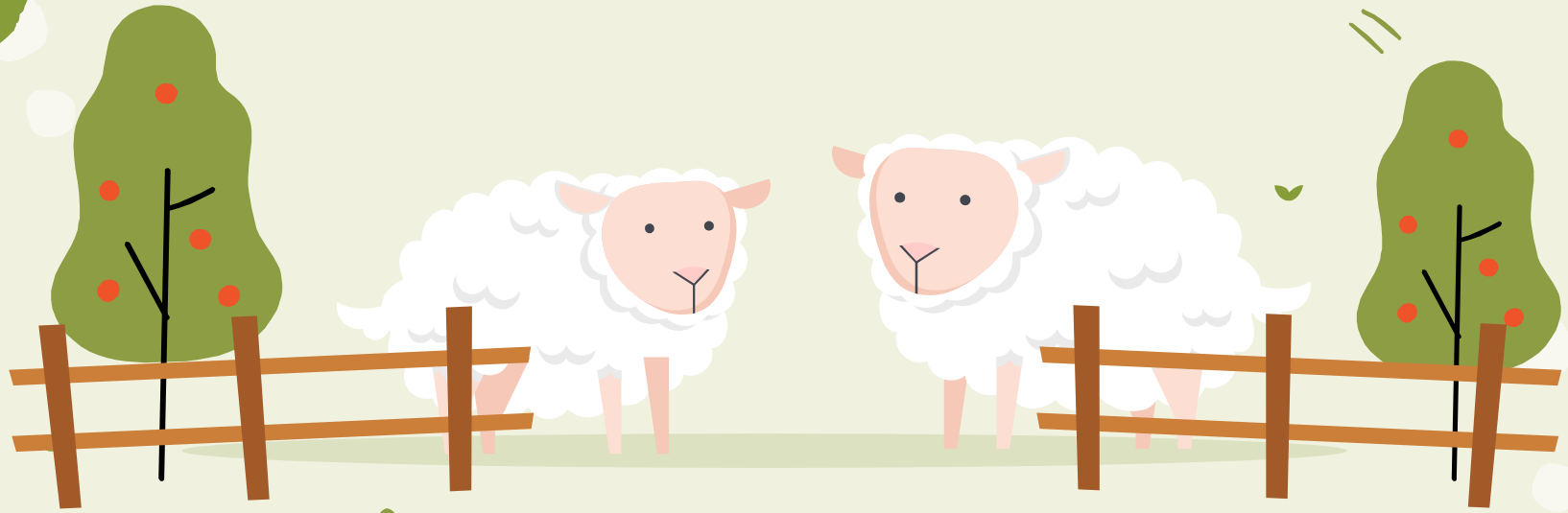


Il rifugio Del gatto Bosisio

Luca Garau - Alessandro Pani - Diego Portas - Filippo Concas

I contenuti

01

Requisiti

Requisiti e progetto del sistema

02

Codice

Descrizione del codice

03

Test Unitari

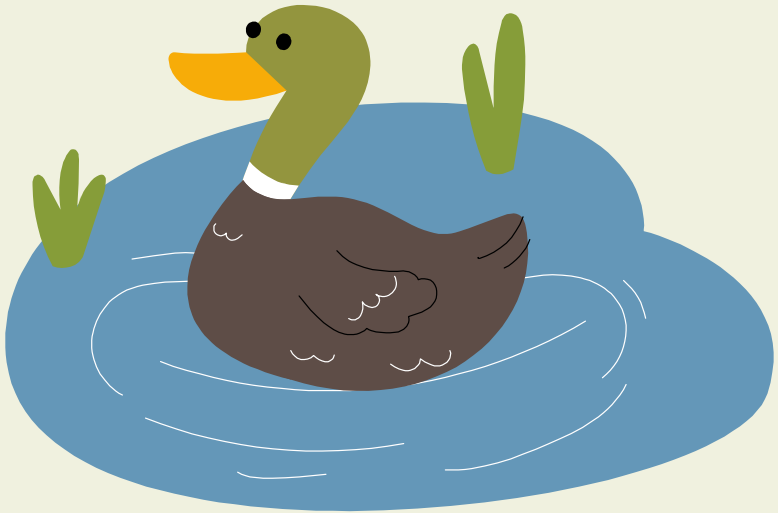
Descrizione ed esecuzione dei test unitari (TU)

04

Test Accettazione

Descrizione ed esecuzione dei test d'accetazione (TA)

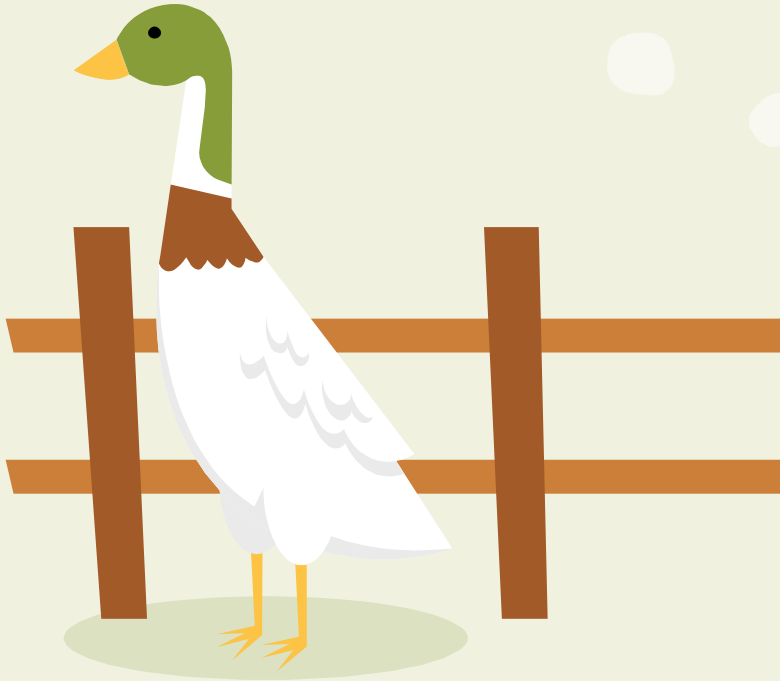
01 Requisiti



Requisiti e progetto del sistema

Lista dei requisiti (elaborati dai casi di uso)

- L'applicazione deve permettere di aggiungere un account per l'utente
- L'applicazione deve permettere di accedere al sistema mediante le proprie credenziali
- L'applicazione deve permettere di far visualizzare e selezionare agli utenti un elenco di animali da poter adottare
- L'applicazione deve permettere di far compilare un modulo per la richiesta di adozione



Lista dei requisiti (elaborati dalle specifiche)

- L'applicazione deve permettere agli admin di poter accedere mediante un account di tipo amministratore
- L'applicazione deve permettere agli admin di approvare o rifiutare le richieste, aggiornare lo stato di adozione
- L'applicazione deve permettere agli admin di aggiungere nuovi animali al rifugio, aggiornarne le informazioni personali, rimuoverli quando vengono adottati
- L'applicazione deve permettere di ricercare gli animali per specie, razza e età

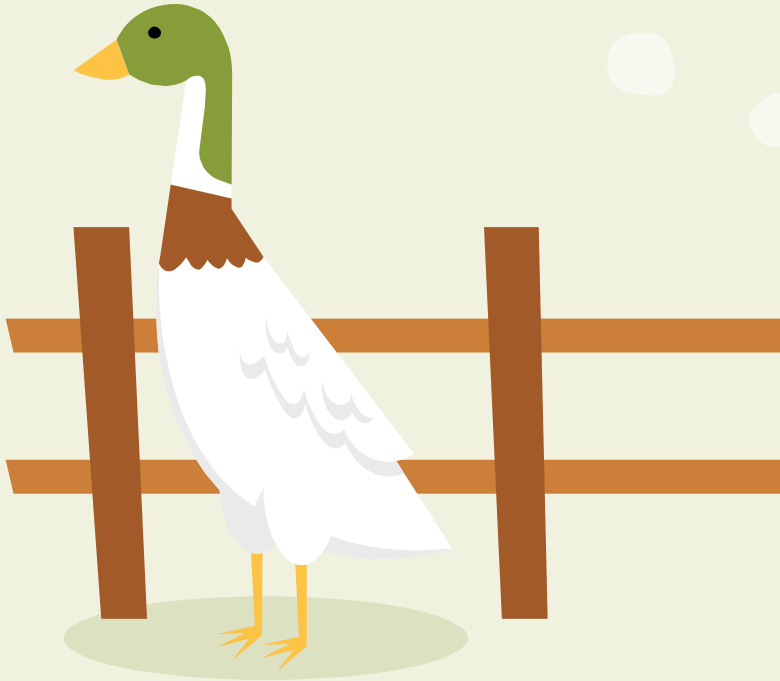


Diagramma delle Classi

All’interno del progetto sono state usate le seguenti classi:

- Utente -> l’utente che utilizza l’applicativo, può essere Admin o User semplice
- Animale -> è l’ospite del nostro rifugio
- Modulo_adozione -> è il modulo che l’utente invia per chiedere l’adozione di un animale
- Stato_adozione -> è un’ enumerazione che indica lo stato dell’animale nel rifugio

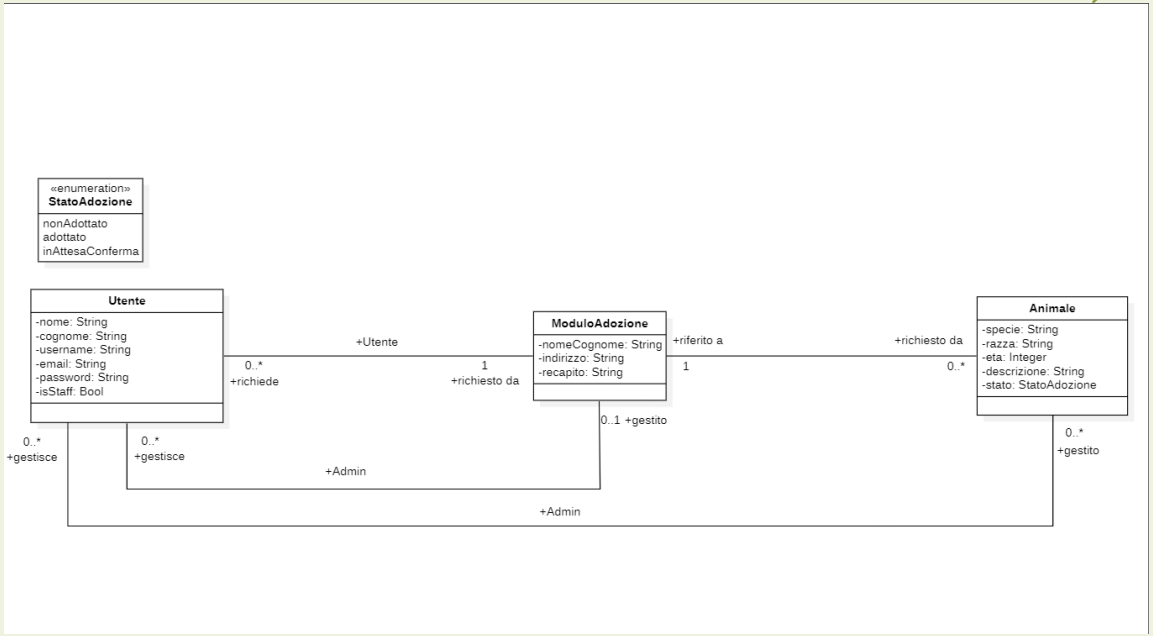


Diagramma di sequenza dei Moduli di adozione

Descrive la sequenza per adottare un animale del rifugio.

- L'utente seleziona un animale
- L'utente compila il modulo
- Il modulo viene inviato ad un amministratore
- Un amministratore può decidere se accettare o rifiutare la richiesta
- Viene aggiornato lo stato dell'animale

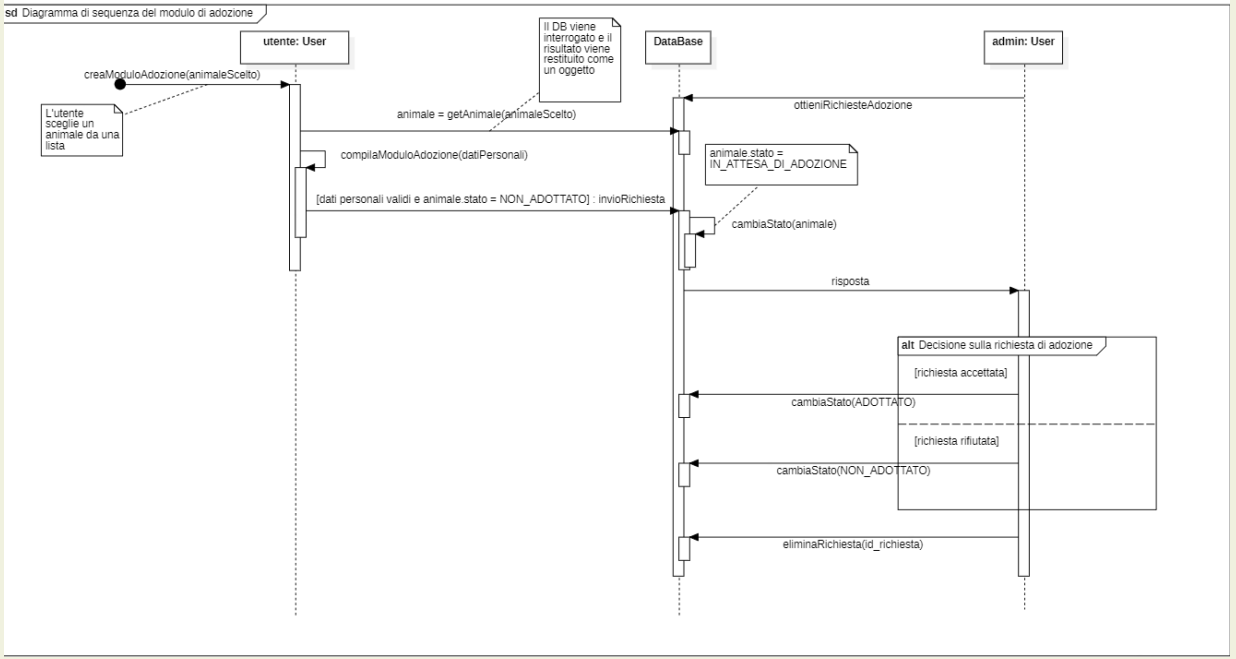
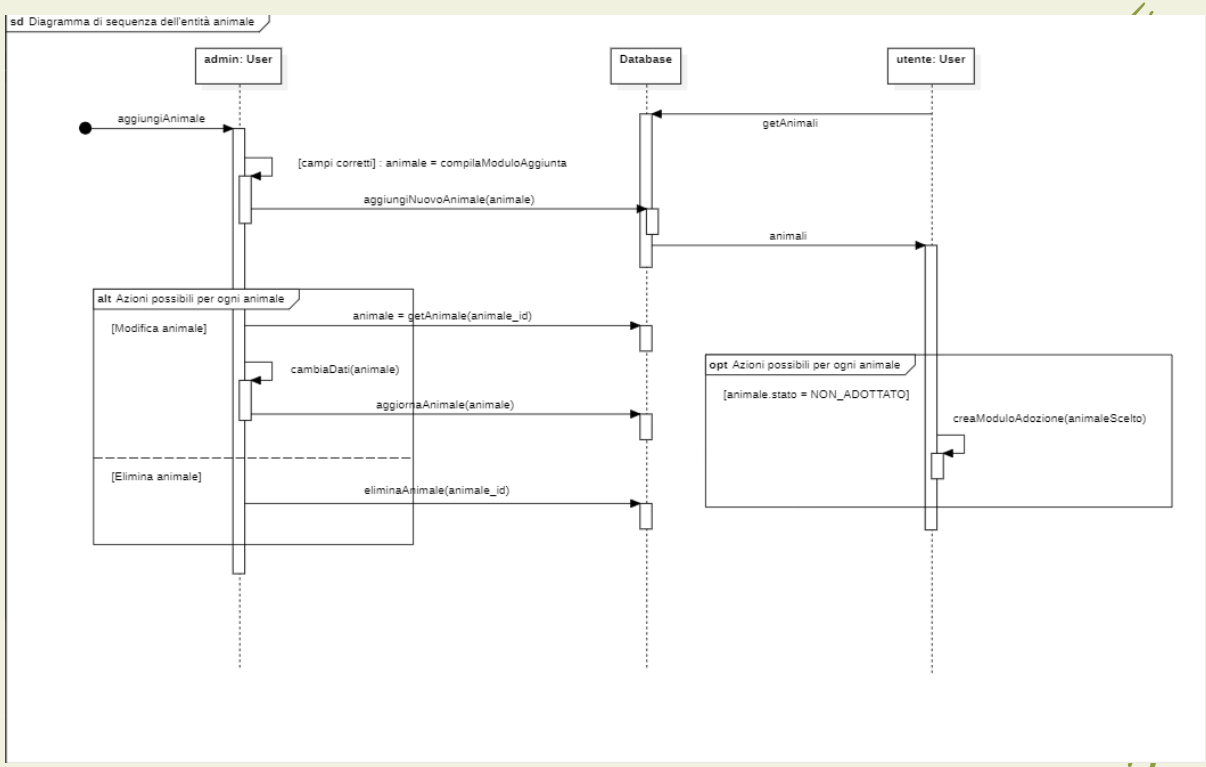


Diagramma di sequenza di gestione Animali

Descrive la sequenza per gestire gli animali del rifugio.

- L'utente sceglie se modifcare, aggiungere o eliminare un animale
- Compila il relativo form
- Vengono aggiornati o aggiunti i dati sull'animale



02

Codice

Descrizione del codice

Struttura del codice

Il codice è diviso tra i seguenti file:

- Model.py -> contiene la struttura del modello
- Views.py -> svolge la funzione di controller
- Urls.py -> collega views e template
- Forms.py -> contiene il form per la registrazione
- Filters.py -> definisce i form e i filtri
- Templates/* -> sono le pagine HTML che costituiscono I template

UNO SGUARDO APPROFONDITO ALLE VIEWS

Le views hanno la funzione di controller dell'applicativo e il loro funzionamento è caratterizzato da:

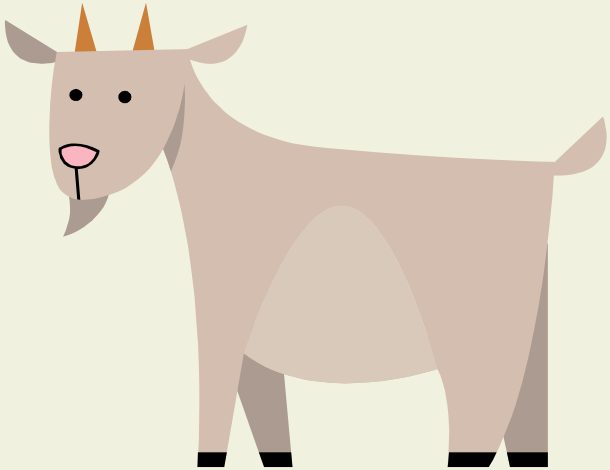
- Il passaggio dei dati tra le varie pagine avviene principalmente tramite richieste HTTP con mettodo POST
- I dati passati ai template sono contenuti nel context
- Vengono effetuati controlli sulla validità dei campi inseriti nei form e sui dati passati
- Vengono effetuati controlli aggiuntivi se per accedere alla pagina è necessario aver effettuato l'accesso o essere amministratore
- Permettono l'interazione con il Database

```python
@login_required(login_url='login')
def gestione_animali(request):

    #controllo se admin
    if(not(request.user.is_staff)):
        return redirect('home')

    lista_animali = Animale.objects.order_by("specie")
    lista_filtrata = AnimaleAdminFilter(request.GET, queryset=lista_animali)
    template = loader.get_template("rifugioAnimali/gestione_animali.html")
    context = {
        "lista_animali" : lista_animali,
        "lista_filtrata" : lista_filtrata,
    }
    return HttpResponse(template.render(context,request))
```

Esempio di views: view della pagina gestione_animali

03

Test Unitari (TU)



Descrizione ed
esecuzione di test unitari

Struttura dei test

Sono test Whitebox che testano la correttezza del codice.

Per effettuare l test viene usato un DB temporaneo e viene usata la libreria unittest

Sono stati effettuati 100 TU divisi tra:

- Test sul model -> verifica la correttezza del modello, controllando se sono stati rispettati tutti i vincoli
- Test sulle views -> verificano l'accesso alla pagina, i reindirizzamenti, il passaggio dei dati e la corretta interazione con il DB

```python
class TestInvioAggiungiAnimaleViewTestCase(TestCase):
    def setUp(self):
        self.user = User.objects.create_user(username='user', password='user')
        self.admin = User.objects.create_superuser(username='admin', password='admin',)

    def test_invio_aggiungi_animale_view(self):
        self.client.login(username='user', password='user')
        response = self.client.post(reverse('invio_aggiungi_animale'), {
            'specie' : 'cane',
            'razza' : 'pastore tedesco',
            'eta' : 5,
            'descrizione' : 'cane di 5 anni',
        })
        self.assertEqual(response.status_code, 302)
```

04

Test accettazione

Descrizione ed esecuzione
di test di accettazione
(TA)

Struttura dei test

I test di accettazione sono stati fatti utilizzando Selenium.
è stato utilizzato un database temporaneo tramite l'utilizzo della libreria unittest

Sono stati fatti 17 AT divisi nel seguente modo:

- TestAccettazioneLogin-> test per il login dell'utente (sia utente normale che admin)
- TestAccettazioneRegistrazione-> suite di test per la registrazione dell'utente
- TestAccettazioneRichiestaAnimale-> suite di test per la pagina di visualizzazione animale e richiesta adozione
- TestAccettazioneGestioneRichiesta-> suite di test per la pagina di visualizzazione animali e gestione delle richieste di adozione
- TestAccettazioneGestioneAnimali-> suite di test per la pagina di gestione animali

```python
def test_visualizzazione_animale_adozione(self):
    login("user","Ciaociao1!",self.driver,self.live_server_url)
    time.sleep(5)
    #premiAdotta
    self.driver.find_element("id","adotta_101").click()
    time.sleep(2)

def test_richiesta_adozione_animale(self):
    login("user","Ciaociao1!",self.driver,self.live_server_url)

    #premiAdotta
    self.driver.find_element("id","adotta_101").click()
    time.sleep(5)
    #compilazione modulo adozione
    nomeBox = self.driver.find_element("id","adotta_nome")
    nomeBox.send_keys("Mario")

    indirizzoBox = self.driver.find_element("id","adotta_indirizzo")
    indirizzoBox.send_keys("via da qui 45")

    recapitoBox = self.driver.find_element("id","adotta_recapito")
    recapitoBox.send_keys("1234567890")

    time.sleep(2)

    adotta = self.driver.find_element("id","adotta")
    adotta.send_keys(Keys.ENTER)

    time.sleep(2)
```


Grazie!