

Università degli Studi di Modena e Reggio Emilia  
Esame di Tecnologie Web



Gardenia – Una App per il Green Sharing  
Documento Tecnico

Filip Czuba (Mat. 168966)  
Anno Accademico 2023-2024

# Introduzione a Gardenia

Gardenia è una web app che basa sull'idea di green sharing, ovvero l'atto di rendere accessibili ad altre persone tramite affitto degli spazi verdi di propria appartenenza.

Questo ha come obiettivo quello di dare a tutti la possibilità di fruire spazi verdi a costi accessibili e dà la possibilità ai proprietari di giardini, terrazzi o – più in generale – spazi all'aperto di recuperare le spese di gestione di tale proprietà, rendendo così tali luoghi più sostenibili sia economicamente che, rendendoli fruibili a più persone, ecologicamente.

L'idea per Gardenia nasce dalla volontà di realizzare un clone di celebri applicazioni quali *AirBnB* – il quale ha rivoluzionato il mondo degli affitti a breve termine – e *Affitto Giardino*, una piattaforma italiana che ha un obiettivo analogo a quello di *Gardenia*.

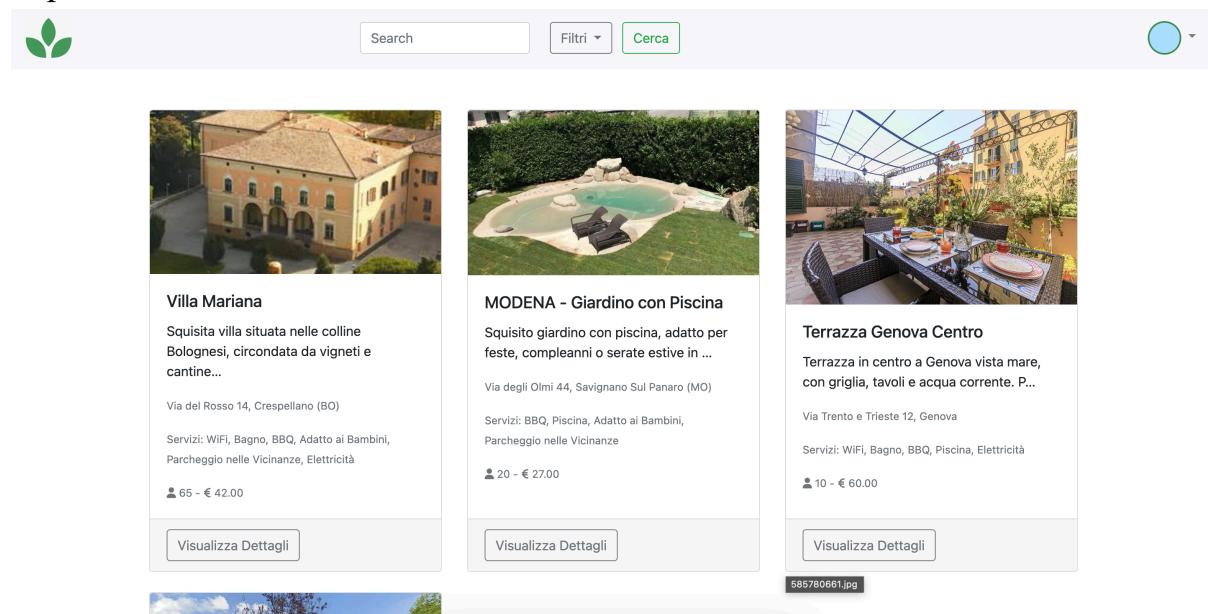


Fig.1: Screenshot della homepage di Gardenia.

# Tecnologie Utilizzate

---

Gardenia è sviluppata utilizzando un'architettura tecnologica moderna basata su Django, un framework web altamente scalabile, e integra diverse librerie e strumenti per garantire una piattaforma solida, efficiente e user-friendly. Di seguito vengono analizzate le principali tecnologie utilizzate e il loro contributo al funzionamento del progetto.

## Backend: Django Framework e SQLite3

---

Django rappresenta il fondamento dell'architettura di Gardenia, gestendo tutte le operazioni lato server. Grazie alla sua modularità, il progetto è suddiviso in app dedicate, ognuna delle quali si occupa di specifiche funzionalità:

- **accounts:** Gestione degli utenti, con un modello personalizzato *CustomUser* che consente di differenziare i ruoli tra *Landlord* (proprietario dello spazio) e *Renter* (affittuario). Questa app include funzionalità come registrazione, autenticazione e modifica del profilo.
- **posts:** Gestione degli annunci pubblicati dai *Landlord*. Permette di creare, modificare ed eliminare spazi verdi, oltre a gestire richieste di affitto da parte degli utenti.
- **reviews:** Implementa un sistema di recensioni, attraverso il quale i *Renter* possono valutare gli spazi affittati, fornendo un feedback visibile anche ai *Landlord*.

Per quanto riguarda DBMS invece viene lasciata l'impostazione di default proposta da Django, ovvero **SQLite 3**.

## Frontend

---

Il frontend di Gardenia è semplice e minimale: per l'implementazione sono state utilizzate principalmente tecnologie quali **CSS**, **HTML**, **JavaScript**, **Bootstrap Versione 4** e il **Django Template Engine**.

Bootstrap 4 è il framework front-end principale utilizzato per la progettazione dell'interfaccia di Gardenia. Esso consente di creare layout responsivi e

componenti UI predefiniti, garantendo una presentazione professionale e uniforme in tutte le pagine. Alcune caratteristiche distintive dell'implementazione di Bootstrap includono:

- **Griglie e Layout Fluidi:** La piattaforma utilizza il sistema grid di Bootstrap per organizzare i contenuti in colonne, adattandoli automaticamente alle diverse dimensioni dello schermo. Questo garantisce che Gardenia sia completamente mobile-friendly e accessibile su dispositivi di tutte le dimensioni.
- **Componenti Personalizzati:** Navbar, modali, pulsanti e card sono personalizzati con classi CSS aggiuntive per integrare elementi visivi unici, come colori verdi e icone intuitive, che richiamano il tema naturale della piattaforma.
- **Feedback Visivi:** Bootstrap è utilizzato per fornire feedback visivi chiari agli utenti, ad esempio evidenziando errori nei form o mostrando indicatori di successo per azioni completate, come la creazione di un annuncio o una recensione.

A Bootstrap vengono associati **Crispy Forms** e **Widget Tweaks** per rendere i form più user-friendly e aggiungere stili personalizzati.

Gardenia sfrutta il **Django Template Engine**, un potente strumento offerto dal framework Django, per gestire il rendering delle pagine web e l'interazione tra frontend e backend. Questo motore consente di generare pagine HTML dinamiche basandosi sui dati forniti dal backend, offrendo un'esperienza utente altamente personalizzata e fluida.

## Modelli e funzionalità

---

L'applicazione **Gardenia** utilizza un set di modelli Django ben strutturati per rappresentare le entità fondamentali del sistema, come utenti, annunci di proprietà, richieste di prenotazione e recensioni. Ogni modello è progettato per facilitare le operazioni specifiche e garantire un flusso dati coerente all'interno della piattaforma. Di seguito vengono descritti i principali modelli e le loro funzionalità.

## CustomUser

---

Il modello **CustomUser**, esteso da **AbstractUser**, è utilizzato per gestire gli utenti della piattaforma. La personalizzazione permette di utilizzare l'email come identificatore univoco e aggiungere campi specifici per supportare le funzionalità di Gardenia. Gli utenti sono divisi in quattro ruoli principali:

- **Landlord (Proprietario):** Può creare annunci di proprietà e gestire le richieste di prenotazione (approvazione/rifiuto). Ha accesso alle recensioni ricevute per i propri annunci.
- **Renter (Affittuario):** Può cercare annunci, inviare richieste di prenotazione e lasciare recensioni per le proprietà prenotate.
- **Admin (Amministratore):** Può gestire tutti gli utenti, annunci e recensioni attraverso il pannello di amministrazione di Django.
- **Utente non autenticato:** Può navigare nella piattaforma e visualizzare gli annunci, ma non può effettuare prenotazioni o lasciare recensioni.

### Campi principali:

- *user\_type*: Determina se l'utente è un landlord o un renter.
- *profile\_picture*: Permette agli utenti di caricare un'immagine profilo.
- *address* e *description*: Offrono dettagli optionali per arricchire il profilo utente.

### Funzionalità aggiuntive:

- Override dei metodi *save()* e *delete()* per gestire in modo sicuro l'upload e la rimozione delle immagini profilo.
- Gestione avanzata dei permessi con campi personalizzati per *groups* e *user\_permissions*.

## Posts

---

Il modello **Post** rappresenta un annuncio pubblicato da un proprietario. Questo modello include informazioni dettagliate sulla proprietà, come descrizione, indirizzo e opzioni disponibili (es. Wi-Fi, parcheggio, piscina).

### Campi principali:

- **title, description, address:** Forniscono le informazioni di base sull'annuncio.
- **landlord:** Associa l'annuncio al proprietario.
- **max\_people e price\_per\_person:** Definiscono le limitazioni e i costi relativi alla prenotazione.
- **tags (es. wifi, bathroom):** Forniscono un sistema di filtro per migliorare l'esperienza di ricerca degli utenti.

#### **Funzionalità aggiuntive:**

- **Verifica disponibilità:** Metodo *is\_available()* per controllare le date libere in base alle prenotazioni già approvate.
- **Gestione immagini:** Upload sicuro e strutturato delle immagini associate all'annuncio.
- **Eliminazione sicura:** Rimuove le immagini e le directory non utilizzate al momento della cancellazione.

## Reviews

---

Il modello **Review** consente agli affittuari di lasciare una recensione per una proprietà prenotata. Le recensioni aiutano gli altri utenti a valutare la qualità delle proprietà e i servizi offerti dai proprietari.

#### **Campi principali:**

- **renter e post:** Collegano la recensione all'utente che l'ha scritta e alla proprietà recensita.
- **rent\_request:** Associa la recensione a una richiesta di prenotazione specifica.
- **rating:** Valutazione numerica (1-5) della proprietà.
- **review\_text:** Testo descrittivo della recensione.

#### **Funzionalità aggiuntive:**

- **Vincoli:** Una recensione può essere creata solo se la richiesta di prenotazione è stata approvata.
- **Interfaccia intuitiva:** Gli utenti possono modificare o eliminare le proprie recensioni.

## RentRequest

---

Il modello **RentRequest** rappresenta una richiesta di prenotazione inviata da un affittuario per una proprietà specifica. Ogni richiesta passa attraverso tre stati principali: *pending* (in attesa), *approved* (approvata) o *rejected* (rifiutata).

### Campi principali:

- **post:** Associa la richiesta all'annuncio interessato.
- **renter:** Riferisce l'utente che ha effettuato la richiesta.
- **start\_date e end\_date:** Specificano le date della prenotazione.
- **status:** Stato corrente della richiesta.

### Funzionalità aggiuntive:

- Integrazione con il modello Post per garantire che le richieste non si sovrappongano.
- Collegamento diretto con il modello Review per permettere agli affittuari di recensire le proprietà solo dopo che la richiesta è stata approvata.

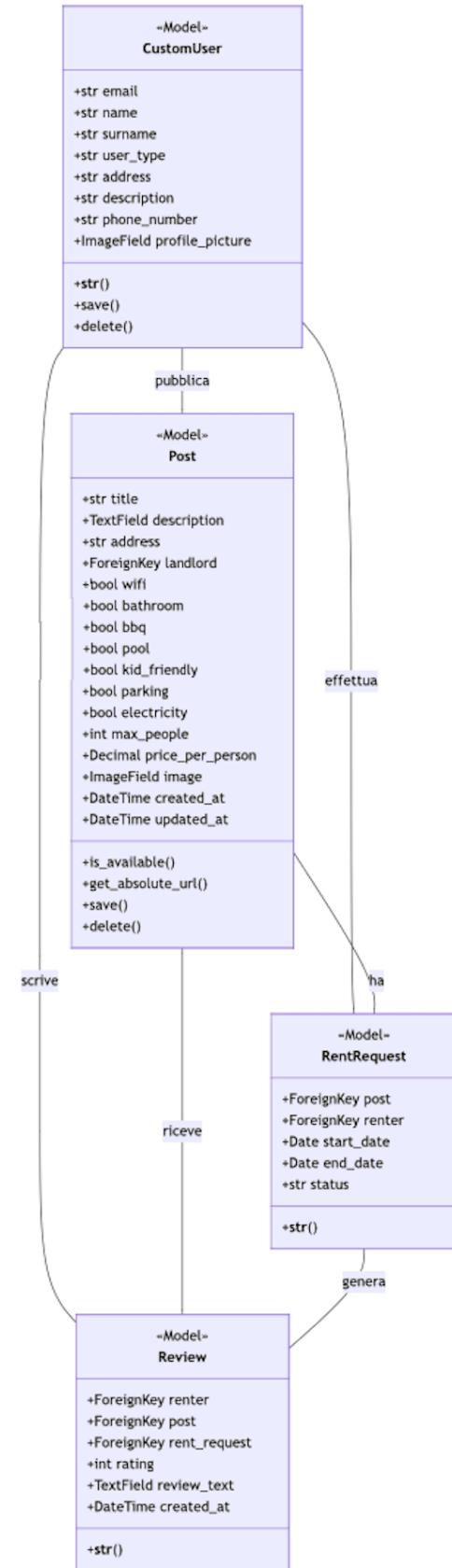


Fig 2: schema delle classi utilizzate in Gardenia.

## Funzionalità principali

Gardenia è dotato di diverse funzionalità che permettono alla web app di essere fruита in modo efficace dagli utenti.

La funzione di ricerca nella HomeView rappresenta un elemento cruciale dell'applicazione, implementando un sistema di filtri dinamici e altamente flessibile che consente agli utenti di trovare proprietà con un livello di precisione straordinario. Il metodo `get_queryset()` è il cuore pulsante di questo meccanismo, dove vengono gestite le diverse dimensioni di ricerca e filtraggio.

**Architettura della Ricerca Testuale** La ricerca testuale utilizza il metodo `Q` di Django, che permette query complesse e flessibili. Quando un utente inserisce un termine di ricerca nel campo '`q`', il sistema esegue una ricerca case-insensitive che abbraccia simultaneamente tre campi:

- **title** (titolo della proprietà)
- **description** (descrizione dettagliata)
- **address** (indirizzo)

Questo approccio consente ricerche estremamente intuitive: un utente che cerca "mare" potrebbe trovare proprietà con descrizioni che menzionano vista mare, vicinanza al mare, o localizzate in zone marittime.

**Filtri Numerici e Quantitativi:** Due filtri numerici permettono una selezione precisa basata su parametri quantitativi come il prezzo massimo e la capienza minima.

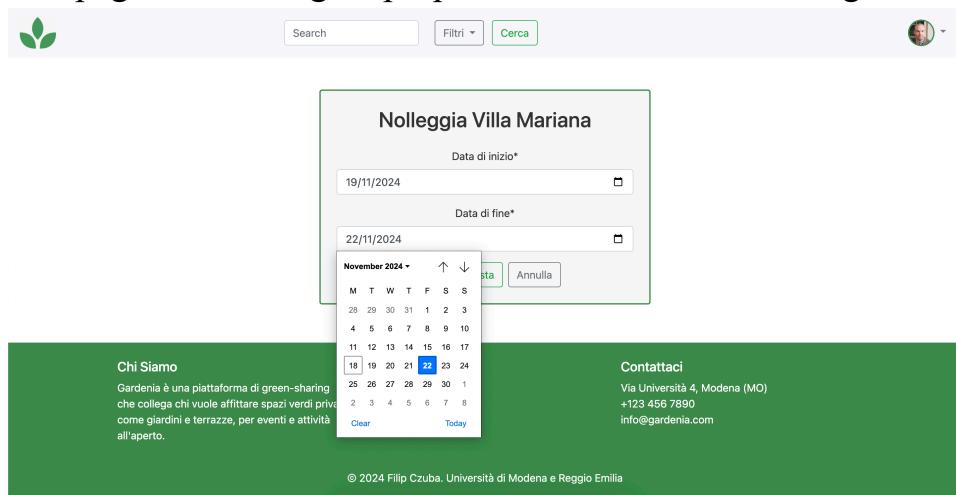
**Filtri per Tag:** Un sistema di selezione di servizi tramite i tag. Viene utilizzata una logica OR che consente selezioni multiple e inclusive.

Fig. 3: Ricerca del termine “Modena” con diversi filtri applicati.

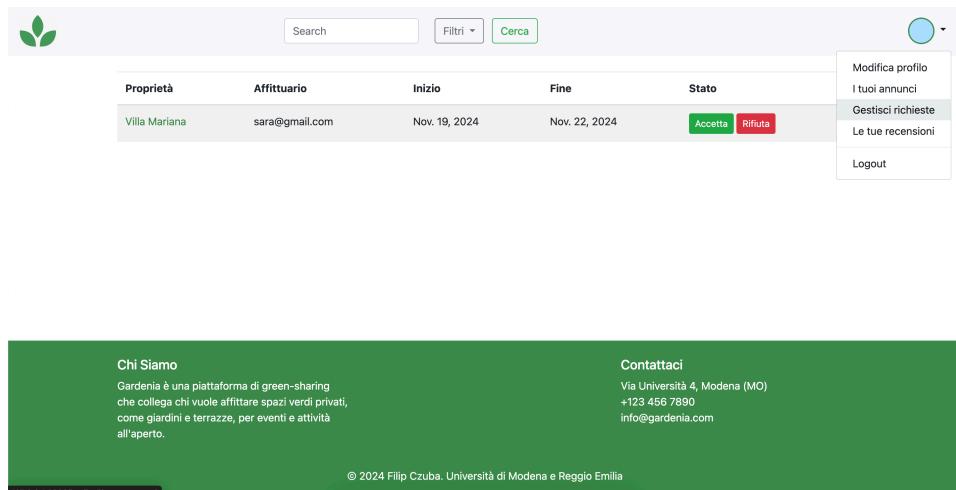
Agli utenti affittuari sono accessibili sistemi di prenotazione e recensione delle proprietà: dopo che un affittuario ha mandato ad un proprietario una richiesta di affitto, il proprietario può accettare o rifiutare la proposta.

Agli affittuari viene impedito di creare prenotazioni che partono o terminano prima della data attuale o in un periodo temporale che si andrebbe a sovrapporre a quello di un'altra prenotazione già accettata relativa alla stessa proprietà.

All'utente, inoltre, è reso possibile recensire le proprietà se e solo se la richiesta di affitto di tale proprietà è stata precedentemente accettata. Le recensioni sono visibili sulle pagine delle singole proprietà e sono visibili a tutti gli utenti.



*Fig. 4: Schermata di prenotazione di una proprietà da parte di un affittuario.*



*Fig. 5: Pannello di Controllo del Proprietario.*

*Fig. 6: Recensioni nella pagina della proprietà.*

## Testing

---

Nel sistema sono state implementate tre principali suite di test per verificare il corretto funzionamento delle componenti fondamentali dell'applicazione. Questi test garantiscono la robustezza e l'affidabilità del sistema attraverso una copertura completa delle funzionalità core.

La suite PostAndBookingTest si occupa di verificare l'intero flusso di gestione degli annunci e delle prenotazioni. In particolare, viene testata la capacità dei proprietari (landlord) di creare e gestire annunci immobiliari, nonché la possibilità per gli affittuari (renter) di effettuare richieste di prenotazione. La suite verifica inoltre il processo di approvazione o rifiuto delle richieste da parte dei proprietari, assicurando che tutte le transizioni di stato avvengano correttamente e che i dati vengano persistiti nel database in modo appropriato.

La suite ReviewTest è dedicata alla verifica del sistema di feedback e recensioni. Questi test sono cruciali per garantire l'integrità del sistema di valutazione, assicurando che solo gli affittuari con prenotazioni effettivamente approvate possano lasciare recensioni. La suite verifica anche la possibilità di lasciare recensioni multiple per diverse prenotazioni dello stesso alloggio, controllando la corretta associazione tra recensioni, prenotazioni e utenti, nonché la validità dei dati inseriti come testo e punteggio.

La suite AccountsTest si concentra sulla verifica dei processi di gestione degli account utente. Vengono testati i flussi di registrazione per entrambe le tipologie di utenti (proprietari e affittuari), i processi di autenticazione (login/logout) e la gestione del profilo utente. Particolare attenzione è stata posta alla verifica della modifica dei dati profilo, inclusa la gestione dell'upload di immagini profilo, e alla corretta implementazione delle restrizioni di accesso per le funzionalità protette. La suite include inoltre test per la validazione dei dati durante la registrazione e per la gestione appropriata dei redirect per gli utenti già autenticati.

I test implementati seguono le best practice di Django e utilizzano il framework di testing integrato, garantendo una copertura completa delle funzionalità critiche del sistema. Ogni suite è stata progettata per essere indipendente e autocontenuta, con un proprio metodo setUp che prepara l'ambiente di test necessario, assicurando così la ripetibilità e l'affidabilità dei test stessi.

## Conclusioni e Futuro di Gardenia

---

Il progetto Gardenia ha permesso di esplorare e implementare le principali funzionalità di un sistema di gestione affitti, dimostrando l'efficacia del framework Django nella costruzione di applicazioni web complesse. L'architettura sviluppata ha posto particolare attenzione alla gestione delle autorizzazioni utente, implementando un sistema robusto di distinzione tra landlord e renter, e ha dimostrato l'efficacia di funzionalità avanzate come il motore di ricerca con filtri multipli e il sistema di gestione delle prenotazioni con verifica delle sovrapposizioni temporali.

Per futuri sviluppi del progetto, sarebbe interessante implementare funzionalità aggiuntive come un sistema di geolocalizzazione per la ricerca delle proprietà, un calendario dinamico per la gestione delle disponibilità e un sistema di messaggistica interno per la comunicazione tra utenti. Dal punto di vista tecnico, l'implementazione di un'API REST permetterebbe di espandere il progetto verso una possibile applicazione mobile, mentre l'ottimizzazione delle query di database potrebbe migliorare ulteriormente le performance del sistema di ricerca.