

UNIVERSITÀ DI FEDERICO II

INGEGNERIA DEL SOFTWARE

---

# DietiEstates25

---

*Autori*

Antonio LEGNANTE

Vincenzo NOVIELLO

*Professori*

Prof. S. DI MARTINO

Prof. L.L.L. STARACE

Gennaio 2026



# Indice

<b>1</b>	<b>Analisi dei requisiti</b>	<b>2</b>
1.1	Glossario . . . . .	2
1.2	Casi d'uso . . . . .	3
1.2.1	Descrizione dei requisiti funzionali . . . . .	3
1.2.2	Diagramma dei casi d'uso . . . . .	4
1.3	Personas . . . . .	4
1.4	Requisiti non funzionali . . . . .	4
1.5	Diagrammi di cockburn . . . . .	4
1.6	Mock-up . . . . .	4
<b>2</b>	<b>Analisi di sistema</b>	<b>5</b>
2.1	Architettura di sistema . . . . .	5
2.2	Schema dei dati . . . . .	7
2.3	Design interfaccia utente - non ho capito - . . . . .	7
2.4	Diagramma delle classi . . . . .	7
2.5	Diagramma di sequenza . . . . .	7
<b>3</b>	<b>Testing</b>	<b>8</b>
3.1	Test del Controller . . . . .	8
3.2	Test del Service . . . . .	8
3.3	Test End-to-end . . . . .	8
3.4	Test Usabilita . . . . .	8

# Capitolo 1

## Analisi dei requisiti

### 1.1 Glossario

Di seguito sono elencati i concetti fondamentali del progetto

<b>Termine</b>	<b>Descrizione</b>
Agente Immobiliare	Professionista appartenente a un'agenzia immobiliare che utilizza la piattaforma per pubblicare annunci, gestire i contatti e comunicare con gli utenti.
Utente Registrato	Persona che crea un account sulla piattaforma e può consultare gli annunci, contattare gli agenti immobiliari e avviare conversazioni.
Agenzia Immobiliare	Organizzazione composta da uno o più agenti immobiliari, responsabile della gestione e pubblicazione degli immobili sulla piattaforma.
Immobile	Proprietà immobiliare pubblicata sulla piattaforma (appartamento, casa, locale commerciale, ecc.) con descrizione, immagini e caratteristiche tecniche.
Annuncio Immobiliare	Scheda informativa di un immobile pubblicata da un agente, contenente dettagli come prezzo, descrizione, foto e disponibilità.
Conversazione / Chat	Canale di comunicazione interna tra utente registrato e agente immobiliare, utilizzato per richieste di informazioni o appuntamenti.
Autenticazione	Processo tramite cui un utente o un agente immobiliare accede alla piattaforma utilizzando le proprie credenziali.
Registrazione	Procedura con cui un nuovo utente crea un account sulla piattaforma inserendo i propri dati personali.

## 1.2 Casi d'uso

### 1.2.1 Descrizione dei requisiti funzionali

#### **Requisito 1: Gestione Utenti e Autenticazione Attori coinvolti**

- Amministratore
- Agente immobiliare
- Utente

##### **Descrizione:**

Il sistema deve permettere la registrazione di nuovi utenti (clienti) e di agenti immobiliari. Tutti gli attori devono essere in grado di effettuare il login con credenziali sicure. L'Amministratore deve poter modificare le credenziali di accesso predefinite e gestire gli account degli agenti.

#### **Requisito 2: Inserimento di Inserzioni Immobiliari Attori coinvolti**

- Agente immobiliare

##### **Descrizione:**

Gli agenti immobiliari possono caricare nuove inserzioni di immobili, complete di dettagli quali foto, descrizione, prezzo, dimensioni, indirizzo, numero di stanze, classe energetica, ecc. Le inserzioni devono essere classificate per tipologia: vendita o affitto.

#### **Requisito 3: Ricerca Avanzata di Immobili Attori coinvolti**

- Utente
- Geopify

##### **Descrizione:**

Il sistema deve permettere la ricerca avanzata di immobili tramite filtri multipli: tipologia, prezzo, posizione geografica (con supporto mappa), numero di stanze, classe energetica, ecc. La ricerca deve essere efficiente e visualizzare i risultati anche tramite mappa interattiva.

#### **Requisito 4: Gestione delle Offerte sugli Immobili Attori coinvolti**

- Agente immobiliare
- Utente registrato

##### **Descrizione:**

Gli utenti registrati possono fare offerte per immobili specificando un prezzo. Gli agenti possono accettare, rifiutare o fare controproposte. Deve essere possibile visualizzare uno storico delle offerte sia per l'utente che per l'agente. Gli agenti possono inserire manualmente offerte ricevute esternamente al sistema.

**Requisito 5: Integrazione con Servizi Esterni (Geoapify) Attori coinvolti**

- Geopify

**Descrizione:**

All'atto della creazione di un'inserzione immobiliare, il sistema deve interrogare il servizio esterno Geoapify per verificare la presenza di scuole, parchi o trasporto pubblico nelle vicinanze dell'immobile. Se presenti, verranno mostrati appositi indicatori ("Vicino a scuole", ecc.).

**1.2.2 Diagramma dei casi d'uso**

**1.3 Personas**

**1.4 Requisiti non funzionali**

**1.5 Diagrammi di cockburn**

**1.6 Mock-up**

## Capitolo 2

# Analisi di sistema

### 2.1 Architettura di sistema

**Architettura complessiva** L'applicazione adotta un'architettura client-server organizzata secondo il modello a 3 livelli (3-tier), strutturata come Single Page Application (SPA) e progettata seguendo il pattern MVC. Al primo accesso il server invia al client l'intera applicazione Web (HTML, CSS, JavaScript). Da quel momento l'applicazione viene eseguita interamente nel browser, e il server entra in gioco soltanto per fornire o aggiornare i dati tramite API RESTful, che restituiscono JSON. Il rendering dell'interfaccia utente avviene lato client (Client-Side Rendering).

**Architettura del Server (Backend)** Il lato server adotta un'architettura ispirata al pattern MVC, ma limitata alle componenti Model e Controller, poiché la View è delegata al client.

**Model** Il model comprende:

- Entità: rappresentano le strutture dati persistenti (immobili, utenti, chat, ecc...)
- Logica di business (servizi): implementano le regole applicative (inserimento di un immobile, autenticazione, ricerca immobili, apertura chat. . .).

Le entità costituiscono il nucleo del livello dati. I servizi fungono da ponte tra i Controller e il database, incapsulando la logica applicativa.

**Controller** Il Controller:

- riceve le richieste HTTP dai client (routing),
- le interpreta
- invoca la logica dei servizi

- restituisce risposte JSON

Non produce HTML (la View è lato client). Il backend espone esclusivamente servizi RESTful.

**Sistema di persistenza dei dati** Il sistema di persistenza dei dati è costituito:

- Sistema di persistenza dei dati testuali: gestito tramite un ORM (Object–Relational Mapping), che assicura una netta separazione tra la logica applicativa e il database relazionale. L'ORM consente di modellare le entità come oggetti, gestire le relazioni in modo trasparente e semplificare le operazioni CRUD.
- Sistema di persistenza delle immagini gestito tramite un apposito Database. In questo modo si garantiscono migliori prestazioni, minor carico sul database e una maggiore leggerezza complessiva del sistema. Nel database viene mantenuto solo il riferimento (path o URL) all'immagine.

**Sistema di autenticazione e autorizzazione** L'applicazione adotta un meccanismo stateless di autenticazione e autorizzazione basato su JWT (JSON Web Token). Al momento del login, il server genera un token firmato contenente le informazioni essenziali sull'utente, come identificatore, ruolo e tempo di scadenza. Il client memorizza il token localmente e lo include in tutte le successive richieste HTTP verso le API RESTful, tramite l'header Authorization. Il server verifica la validità e l'integrità del token ad ogni richiesta, senza mantenere sessioni lato backend. Questo approccio garantisce: sicurezza, scalabilità e flessibilità.

**Architettura del client** Il client costituisce la View dell'applicazione e gestisce:

- il rendering dell'interfaccia (Client-Side Rendering),
- la gestione dello stato,
- le interazioni dell'utente,
- le chiamate alle API REST del backend.

**Sintesi dei 3 livelli (3-tier)**

1. Presentation Tier (Client / Browser)
  - View
  - SPA renderizzata lato client
  - Gestione dell'interfaccia e delle interazioni
  - Memorizzazione e utilizzo dei token JWT
2. Application Tier (Server / Logica di business)
  - Controller

- Servizi (Business Logic)
- Gestione sicurezza, verifica dei token JWT ad ogni richiesta

### 3. Data Tier (Database)

- Dati testuali
- Dati multimediali (immagini)
- Operazioni CRUD sui dati

#### **Ciclo delle richieste**

1. L'utente invia una richiesta HTTP/HTTPS (GET o POST)
2. DispatcherServlet intercetta la richiesta e la inoltra al controller appropriato. (Front Controller design pattern)
3. il controller elabora la logica, interagisce con il model e prepara i dati
4. il controller restituisce i dati in formato JSON

## **2.2 Schema dei dati**

## **2.3 Design interfaccia utente - non ho capito -**

## **2.4 Diagramma delle classi**

## **2.5 Diagramma di sequenza**

## Capitolo 3

# Testing

3.1 Test del Controller

3.2 Test del Service

3.3 Test End-to-end

3.4 Test Usabilita