



DietiEstates25

GESTIONE SERVIZI IMMOBILIARI

Documentazione Ingegneria del Software
DietiEstates25

Gruppo INGSW2425_016

Florindo Zecconi (Matricola N86004544)

Raffaele Raia (Matricola N86004564)

Domenico Gagliotti (Matricola N86004536)

Anno 2024/2025

Indice

1	Introduzione	7
1.1	Gestione Documentazione	7
2	Documento dei requisiti	8
2.1	Glossario	8
2.2	Personas	9
2.2.1	Boris Yellnikoff - Amministratore con esperienza	10
2.2.2	Leandro Esposito - Amministratore junior	11
2.2.3	Vincent Vega - Agente immobiliare novello	12
2.2.4	Laura Bianchi - Agente immobiliare navigato	13
2.2.5	Mia e Sebastian - Giovane coppia sposata	14
2.2.6	Giovanni Rossi - Anziano	15
2.2.7	Francesco Totti - Pendolare	16
2.2.8	Analisi Personas	17
2.3	Casi d'uso - Use Case Diagram	19
2.3.1	Attori individuati	19
2.4	Mock-Up	21
2.4.1	Ricerca Immobile Per Città	21
2.4.2	Crea Agenzia	22
2.4.3	Gestisci Amministratori	23
2.4.4	Crea Amministratori	23
2.5	Tabelle Cockburn	25
2.5.1	Cockburn UC1 - Ricerca Immobile Per Città	25
2.5.2	Cockburn UC2 - Crea Agenzia	27
2.5.3	Cockburn UC3 - Gestione Amministratori	29
2.5.4	Cockburn UC4 - Crea Amministratore	30
2.6	Requisiti non-funzionali e di dominio	32
2.6.1	Requisiti non-funzionali	32
2.6.2	Requisiti di dominio	32
3	System Design	34
3.1	Architettura	34
3.1.1	Client-Server	34
3.1.2	N-Tier	35
3.2	Tecnologia	39
3.3	Persistenza dati	41
3.3.1	Class Diagram	41

3.3.2	Class Diagram Ristrutturato	42
3.3.3	Modello Logico	43
3.3.4	Schema Fisico	44
3.4	Class Diagram	49
3.4.1	AccessService	49
3.4.2	AdminManagementService	50
3.4.3	AdsEstateService	51
3.4.4	AgencyService	52
3.4.5	AgentManagementService	53
3.4.6	AppointmentService	54
3.4.7	DAO	55
3.4.8	DatabaseLib	56
3.4.9	DiEtiEstateExceptions	57
3.4.10	EmailSender	58
3.4.11	ManagementAccountService	59
3.4.12	Model	60
3.4.13	NotifyService	61
3.4.14	SearchService	62
3.4.15	Validator	63
3.5	Diagrammi di sequenza	64
3.5.1	Crea Agenzia	64
3.5.2	Ricerca Immobile Per Nome	65
3.5.3	Gestisci Admin	66
3.5.4	Aggiungi Admin	67
4	UI Design	68
4.1	Mockup ad alta fedeltá	68
4.2	Attributi di qualità	69
4.2.1	Learnability	69
4.2.2	Efficiency	69
4.2.3	Memorability	69
4.2.4	Gestione degli Errori	70
4.2.5	Satisfaction	70
4.3	Principi del design	71
4.4	Leggi	72
4.4.1	The Power Law of Practice	72
4.4.2	Hick's law	72
4.4.3	Fitt's law	72
4.5	Color Theory	74
4.5.1	Palette	74
4.5.2	Armonies	74
4.6	Tipografia	75
4.7	Gestalt theory of perception	75
4.7.1	Principio di similarità	75
4.7.2	Principio di prossimità	75
4.7.3	Principio di connessione	75
4.7.4	Principio della regione comune	75
4.7.5	Gerarchia	76
4.8	Linee guida e principi nella HCI	78
4.8.1	Euristiche di Usabilitá Nielsen-Molich	78

5 Software Design	80
5.1 Istruzioni utilizzo	80
5.1.1 Descrizione	80
5.1.2 Build delle immagini Docker	80
5.1.3 Variabili d'ambiente	80
5.1.4 Avvio dei servizi	80
5.1.5 Struttura principale	81
5.1.6 Qualità del codice	81
5.1.7 Versioning	82
6 Testing	83
6.1 Progettazione dei test	83
6.1.1 AddAdmin test	83
6.1.2 Test: <code>updateEstateAgent</code>	86
6.1.3 AddAgent test	87
6.1.4 <code>applyChangeAcquirente</code> test	90
6.2 Tecniche di valutazione dell'usabilità	92
6.2.1 Expert reviews / inspections	92
6.2.2 Progettazione e conduzione di un esperimento	93

Elenco delle figure

2.1 Personas - Amministratore con esperienza	10
2.2 Personas - Amministratore junior	11
2.3 Personas - Agente immobiliare novello	12
2.4 Personas - Agente immobiliare navigato	13
2.5 Personas - Giovane coppia sposata	14
2.6 Personas - Utente anziano	15
2.7 Personas - Pendolare	16
2.8 Use Case Diagram - DietiEstate25	19
2.9 Mock-Up Low Fidelity - Ricerca Immobile Per Nome	21
2.10 Mock-Up Low Fidelity - Crea Agenzia	22
2.11 Mock-Up Low Fidelity - Gestisci Amministratori	23
2.12 Mock-Up Low Fidelity - Crea Amministratori	24
3.1 Schema Non ristrutturato	41
3.2 Schema Ristrutturato	42
3.3 AccessService	49
3.4 AdminManagementService	50
3.5 AdsEstateService	51
3.6 AgencyService	52
3.7 AgentManagementService	53
3.8 AppointmentService	54
3.9 DAO	55
3.10 DatabaseLib	56
3.11 DiEtiEstateExceptions	57
3.12 EmailSender	58
3.13 ManagementAccountService	59
3.14 Model	60
3.15 NotifyService	61
3.16 SearchService	62
3.17 Validator	63
3.18 Sequence Diagram - Crea Agenzia	64
3.19 Sequence Diagram - Ricerca Immobile Per Nome	65
3.20 Sequence Diagram - Gestisci Admin	66
3.21 Sequence Diagram - Aggiungi Admin	67
4.1 MockUp Alta fedeltà	68
4.2 Color Palette	74

4.3	Color Armonies	74
4.4	Esempio Tipografia	75
4.5	Home page	76
5.1	Qualità del software misurata da SonarQube	81
5.2	Versioning - Statistiche	82
6.1	Semaforo – Classificazione dei risultati	94
6.2	Semaforo – Distribuzione dei risultati	95
6.3	Medie – Tempo, errori e richieste di aiuto	95
6.4	Menu di navigazione	96
6.5	Tasto per la ricerca	96
6.6	Tempo medio per utente	97
6.7	Media dei punteggi del questionario	98

Elenco delle tabelle

2.1	Cockburn UC1 - Ricerca Immobile Per Nome	25
2.2	Cockburn UC1 - Main Scenario	25
2.3	Cockburn UC1 - Extension A : cerca una città o comune inesistente	26
2.4	Cockburn UC1 - Extension B : clicca “Indietro” in #M4, #M5 o #M3	26
2.5	Cockburn UC1 - Extension C : clicca “Indietro” in #M6	26
2.6	Cockburn UC1 - Extension D : clicca “Indietro” in #M2	26
2.7	Cockburn UC2 - Crea Agenzia	27
2.8	Cockburn UC2 - Main Scenario	27
2.9	Cockburn UC2 - Extension A : Il nome dell'agenzia è già stato utilizzato	28
2.10	Cockburn UC2 - Extension B : la partita IVA è già stata registrata	28
2.11	Cockburn UC2 - Extension C : l'email è già stata utilizzata	28
2.12	Cockburn UC2 - Extension D : l'utente annulla la conferma della creazione dell'agenzia	28
2.13	Cockburn UC3 - Gestione Amministratori	29
2.14	Cockburn UC3 - Main Scenario	29
2.15	Cockburn UC3 - Extension A : clicca il tasto annulla	30
2.16	Cockburn UC3 - Extension B : clicca il tasto indietro	30
2.17	Cockburn UC3 - Crea Amministratore	30
2.18	Cockburn UC3 - Main Scenario	31
2.19	Cockburn UC4 - Extension A : clicca il tasto indietro	31
2.20	Cockburn UC4 - Extension B : Email già presente	31
2.21	Cockburn UC4 - Extension C : clicca il tasto annulla	31
3.1	Schema logico - fine	43
6.9	Valutazione qualitativa dell'interfaccia	93
6.10	Soggetti esperimento	93

Capitolo 1

Introduzione

DietiEstates25 è una piattaforma per la gestione di servizi immobiliari. Il sistema permette a più agenzie di pubblicare inserzioni immobiliari. Gli utenti possono quindi visualizzare le inserzioni, prenotare visite e fare offerte per acquistare/affittare un immobile.

1.1 Gestione Documentazione

Come richiesto la documentazione possiede quattro macro aree :

- **Documento dei Requisiti Software**
- **Documento di Design del sistema**
- **Artefatti Software e documentazione del processo di sviluppo.**
- **Testing e valutazione dell'usabilità**

Capitolo 2

Documento dei requisiti

2.1 Glossario

- **DietiEstates25** : DietiEstates25 è una piattaforma per la gestione di servizi immobiliari. Il sistema permette a più agenzie di pubblicare inserzioni immobiliari. Gli utenti possono quindi visualizzare le inserzioni, prenotare visite e fare offerte per acquistare/affittare un immobile.
- **Immobile** : Un bene fisico che non può essere spostato, come una casa, un appartamento o un ufficio. Gli immobili sono oggetto di transazioni immobiliari.
- **Servizi immobiliari** : Un insieme di servizi offerti da professionisti del settore immobiliare, tra cui gestione di affitti, gestione di vendite, oltre a assistenza legale e finanziaria per le transazioni.
- **Inserzioni immobiliari** : Annunci pubblicati su DietiEstates25 che descrivono un immobile disponibile per la vendita o l'affitto. Le inserzioni includono dettagli come la posizione, il prezzo e tutte le caratteristiche dell'immobile.
- **Acquistare immobile** : Il processo di acquisizione di un immobile, che comporta una visita programmata e una negoziazione del prezzo.
- **Affittare immobile** : L'atto di concedere in uso un immobile a un inquilino in cambio di un pagamento periodico, solitamente mensile. Comporta una visita programmata e una negoziazione del prezzo.
- **Agente immobiliare** : Un professionista del settore immobiliare che funge da intermediario tra acquirenti e venditori di immobili. Gli agenti immobiliari forniscono visite, assistono nelle negoziazioni e aiutano a facilitare il processo di acquisto o affitto. Essi sono gestiti da amministratori della agenzia.
- **Amministratore** : Un professionista responsabile della gestione di un'agenzia immobiliare. Questo ruolo può includere la supervisione degli agenti.
- **Utente** : Qualsiasi persona che interagisce con DietiEstates25, che può essere un acquirente o un inquilino. Gli utenti possono utilizzare l'applicazione per cercare immobili, gestire le proprie offerte immobiliari.
- **Categoria immobili** : Classificazioni utilizzate per raggruppare gli immobili in base a specifiche caratteristiche. Le categorie aiutano gli utenti a filtrare le ricerche e trovare più facilmente le proprietà che soddisfano le loro esigenze.

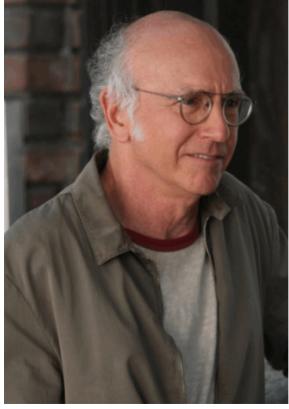
2.2 Personas

Conoscere l'utente gioca un ruolo importante nella definizione dei requisiti, permettendo di progettare interfacce utente efficaci e mirate. Abbiamo deciso di creare archetipi che contengano le seguenti informazioni:

- **Nome**
- **Info** : sesso, età, località e ruolo.
- **Skills** :
 - **Affinità tecnologica** : indica il livello di praticità e familiarità nell'utilizzo di strumenti informatici e tecnologici.
 - **Esperienza immobiliare** : indica il livello di conoscenza e competenza posseduto nell'ambito immobiliare.
- **Tecnologia** : indica quali sistemi o strumenti tecnologici sono più familiari e utilizzati dall'utente.
- **Interessi** : quali interessi hanno nella vita e quali sono legati al mondo immobiliare
- **Obiettivi** : obiettivi personali, cosa vogliono ottenere dalla vita e nel lavoro.
- **Motivazione** : quali sono le motivazioni che incentivano l'uso di un'applicazione come DietiEstates25
- **Frustazione** : quali sono le cose che potrebbero disincentivare l'uso di un'applicazione come DietiEstates25

Tutte queste caratteristiche ci aiuteranno a capire cosa potrebbe cercare un insieme di utenti da un'applicazione, come vorrebbero che fosse implementata, quali funzionalità desiderano e come si comporta il sistema. In sintesi, in mancanza della possibilità di intervistare gli stakeholder e condurre un'analisi etnografica, le personas diventano uno strumento prezioso.

2.2.1 Boris Yellnikoff - Amministratore con esperienza



NOME E COGNOME

Boris Yellnikoff

PERSONALITÀ

Pignolo E Diretto

BIO

Boris Yellnikoff è un gestore di agenzie immobiliari esperto e scettico. Con anni di esperienza nel settore immobiliare, Boris è noto per la sua abilità nel riconoscere opportunità e per la sua capacità di guidare il suo team attraverso le sfide del mercato. È un gestore pragmatico, che non si fa illusioni sulle dinamiche del settore, e la sua personalità schietta lo rende un leader rispettato ma spesso temuto. Nonostante il suo cinismo, Boris è profondamente impegnato nel garantire il miglior servizio possibile ai suoi clienti, analizzando ogni dettaglio con occhio critico e strategico.

INFO

Maschio 60 years

Italia, Milano

Amministratore di una agenzia d'immobilizzazione

Skills

Affinità tecnologica: 60

Esperienza immobiliare: 100

Tecnologia



Interessi

È appassionato di filosofia e ama approfondire argomenti esistenziali attraverso la lettura e le discussioni significative. Ha un forte interesse per la cultura e l'arte, partecipando a mostre e spettacoli teatrali, e apprezza la cucina gourmet, sia nella preparazione di piatti elaborati che nella scoperta di ristoranti raffinati. Nel suo lavoro, è affascinato dall'architettura e segue attentamente le tendenze del mercato immobiliare, con un occhio critico per le opportunità di investimento e i progetti di ristrutturazione. La sua sensibilità verso le questioni ambientali lo spinge a esplorare pratiche sostenibili nel settore immobiliare, cercando soluzioni innovative e eco-compatibili.

Obiettivi

Aspira a eccellere nel suo settore, puntando a diventare un leader rispettato per competenza e integrità. Desidera anche continuare a crescere intellettualmente, approfondendo la sua conoscenza di filosofia, arte e cultura. Boris è motivato a identificare opportunità di investimento redditizie, ottimizzando il suo portafoglio immobiliare. Inoltre, è impegnato nella promozione di pratiche ecologiche, cercando di contribuire a progetti sostenibili e a basso impatto ambientale. Infine, punta a costruire relazioni autentiche con clienti e colleghi, creando un ambiente di lavoro collaborativo e stimolante.

Motivazioni

Gestione centralizzata: L'app gli consentirebbe di avere una visione chiara delle attività e delle performance di tutti gli agenti, facilitando una gestione più organizzata.

Monitoraggio delle vendite: Potrebbe tenere traccia delle vendite in tempo reale, identificando rapidamente eventuali problemi o opportunità di miglioramento.

Comunicazione semplificata: L'app fornirebbe strumenti per comunicare facilmente con gli agenti, migliorando la collaborazione e riducendo i fraintendimenti.

Formazione e sviluppo: Boris potrebbe utilizzare l'app per fornire risorse formative e aggiornamenti, supportando la crescita professionale dei suoi agenti.

Analisi delle performance: L'app offrirebbe report e analisi dettagliate, permettendogli di valutare le prestazioni individuali e di gruppo, e di prendere decisioni informate su strategie di incentivazione e sviluppo.

Frustrazioni

Difficoltà nella gestione degli agenti: Se l'app non fornisce strumenti adeguati per monitorare e gestire le attività degli agenti, Boris potrebbe sentirsi frustrato nel non riuscire a tenere traccia delle loro performance e dei risultati di vendita.

Sicurezza e privacy: Preoccupazioni riguardo alla sicurezza dei dati sensibili dei clienti potrebbero rendere Boris riluttante a utilizzare l'app, temendo potenziali violazioni della privacy.

Interruzioni frequenti: Se l'app fosse soggetta a malfunzionamenti o crash, Boris si sentirebbe esasperato e poco fiducioso nella sua affidabilità, compromettendo la sua professionalità.

Mancanza di informazioni dettagliate e accurate: Desidera dati completi e affidabili per prendere decisioni informate. La superficialità delle informazioni o dati imprecisi non solo ostacolerebbe la sua capacità di valutare il mercato e le performance degli agenti, ma potrebbe anche compromettere la sua reputazione professionale.

Figura 2.1: Personas - Amministratore con esperienza

2.2.2 Leandro Esposito - Amministratore junior



Figura 2.2: Personas - Amministratore junior

2.2.3 Vincent Vega - Agente immobiliare novello



NOME E COGNOME

Vincent Vega

PERSONALITÀ

Spontaneo E Pratico

BIO

Vincent Vega è un nuovo volto nel mondo immobiliare, ma porta con sé una mentalità fresca e pronta all'azione. Dopo anni passati in altri settori, ha deciso di buttarsi nel campo immobiliare, dove la sua abilità nel gestire situazioni impreviste e la sua natura pratica lo rendono perfetto per il lavoro. Anche se agli inizi, Vincent non si perde in chiacchiere: sa ascoltare, adattarsi rapidamente e fare il possibile per trovare l'immobile giusto. Spontaneo e sempre pronto a risolvere i problemi sul campo, Vincent è l'agente che affronta tutto con un tocco di disinvolta e determinazione.

Interessi

Vincent Vega è affascinato da **proprietà vintage** e case dallo stile retrò, riflettendo il suo amore per l'estetica degli anni '50 e '60. Con un approccio pratico e orientato all'azione, è interessato a chiudere **trattative veloci** senza perdere la formalità. Vincent sfrutta il suo modo di fare disinvolto per creare conversazioni informali con i clienti, mettendoli a loro agio. Ha una preferenza per **quartieri vivaci** e culturalmente attivi, apprezzando l'atmosfera dinamica che li caratterizza, e ama scoprire dettagli unici nelle proprietà, rendendo le sue presentazioni immobiliari intriganti e personali.

Obiettivi

Vincent Vega ha l'obiettivo di costruire una clientela fedele, stabilendo relazioni solide per guadagnarsi la fiducia dei clienti. Vuole chiudere le vendite in modo rapido ed efficace, massimizzando le commissioni. Si impegna ad espandere la sua rete di contatti nel settore e ad acquisire conoscenze approfondite sulle tendenze del mercato immobiliare, in particolare riguardo a proprietà vintage e aree in via di sviluppo. Vincent mira anche a migliorare le sue abilità di negoziazione per ottenere le migliori offerte e a distinguersi nel settore creando un marchio personale riconoscibile. Infine, desidera fornire un servizio clienti eccellente, curando ogni dettaglio e mostrando attenzione alle esigenze specifiche di ciascun cliente.

Motivazioni

Gestione delle trattative: Un'area per monitorare offerte e controfferte, con possibilità di allegare documenti per tenere tutto organizzato.

Networking: Consente di connettersi con altri agenti e professionisti del settore per ampliare la rete di contatti.

Calendario integrato: Strumento per pianificare appuntamenti e visite, con promemoria automatica e sincronizzazione al calendario personale.

Recensioni e feedback: Sezione per raccogliere recensioni dai clienti e visualizzare il feedback, migliorando il servizio e la reputazione nel settore.

Frustrazioni

Difficoltà nel raccogliere feedback: Potrebbe avere problemi a ottenere recensioni o feedback dai clienti in modo semplice, ostacolando il miglioramento del suo servizio.

Problemi di sincronizzazione: Se l'app non si sincronizza bene con il calendario o altri strumenti che utilizza, potrebbe perdere appuntamenti importanti.

Lentezza dell'app: Se l'app è lenta nel caricamento o nelle operazioni, potrebbe perdere tempo prezioso e causare irritazione.

Funzionalità inutili o sovraccarico di opzioni: Troppo funzioni non necessarie o complicate potrebbero distrarlo, facendogli perdere il focus su ciò che è realmente importante per il suo lavoro.

Inaffidabilità delle notifiche: Se le notifiche arrivano in ritardo o non funzionano correttamente, potrebbe perdere opportunità di vendita.

Figura 2.3: Personas - Agente immobiliare novello

2.2.4 Laura Bianchi - Agente immobiliare navigato

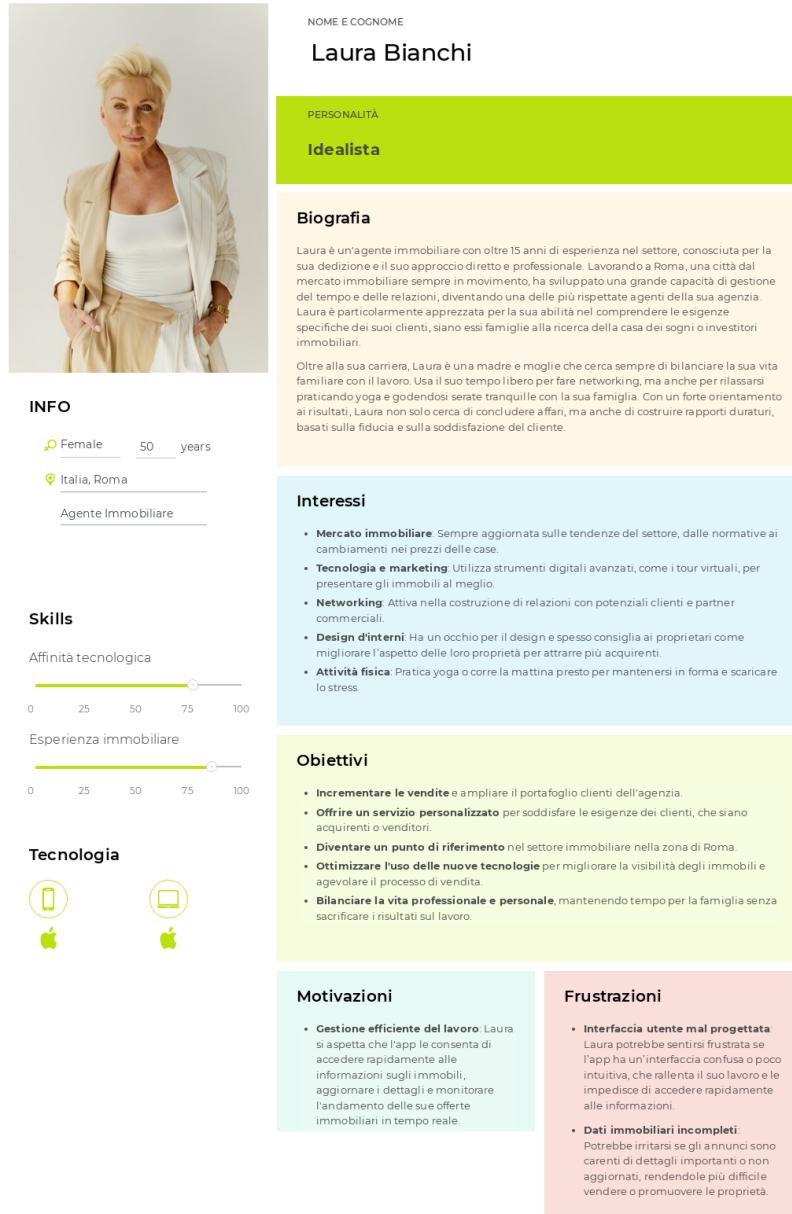


Figura 2.4: Personas - Agente immobiliare navigato

2.2.5 Mia e Sebastian - Giovane coppia sposata

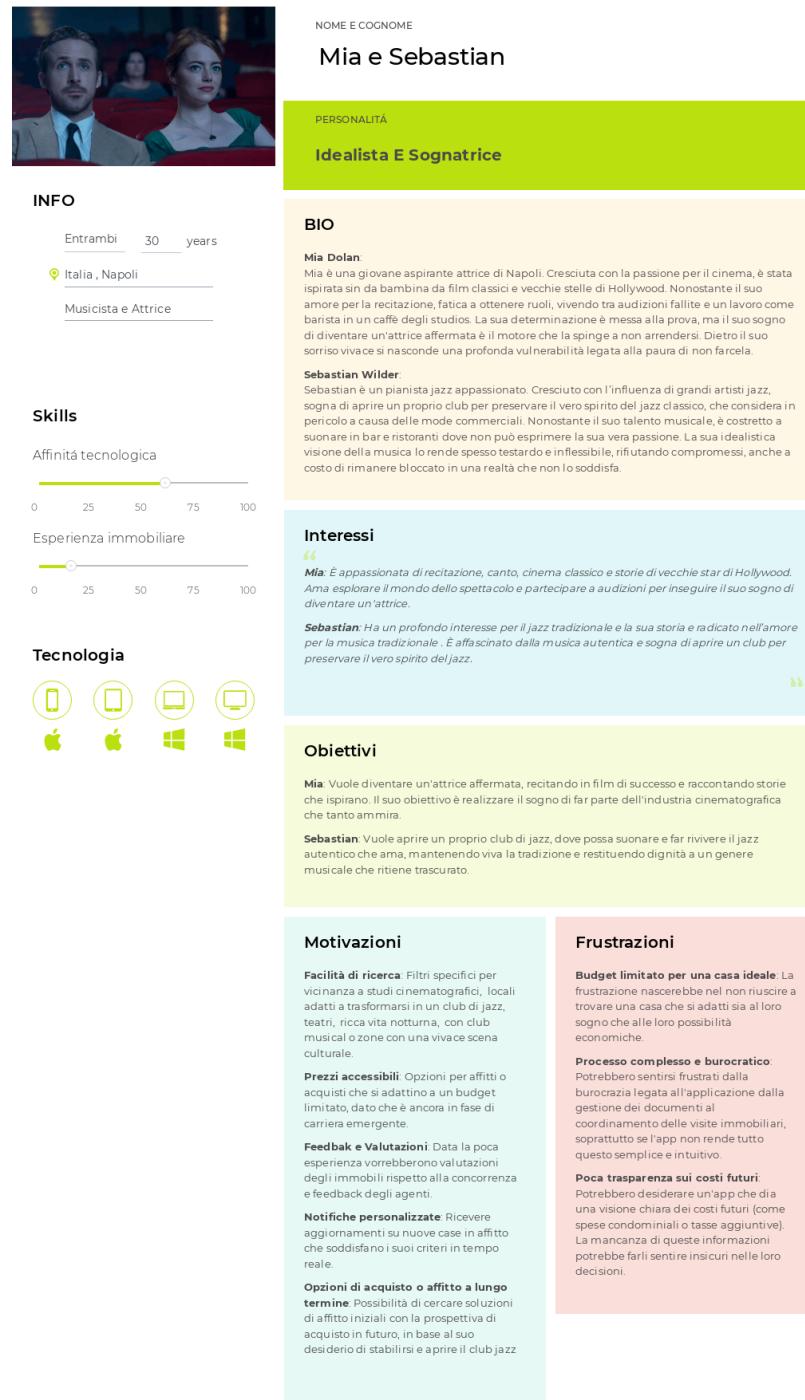


Figura 2.5: Personas - Giovane coppia sposata

2.2.6 Giovanni Rossi - Anziano

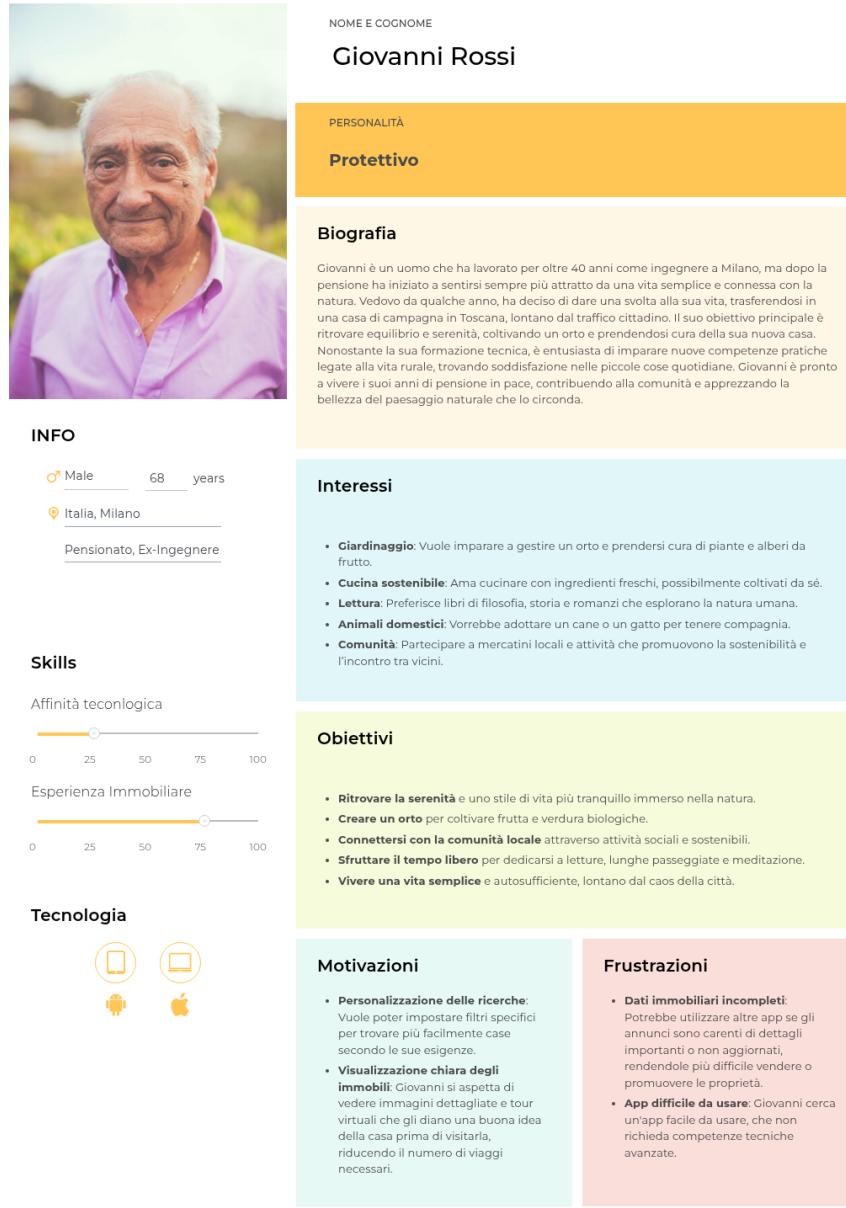


Figura 2.6: Personas - Utente anziano

2.2.7 Francesco Totti - Pendolare

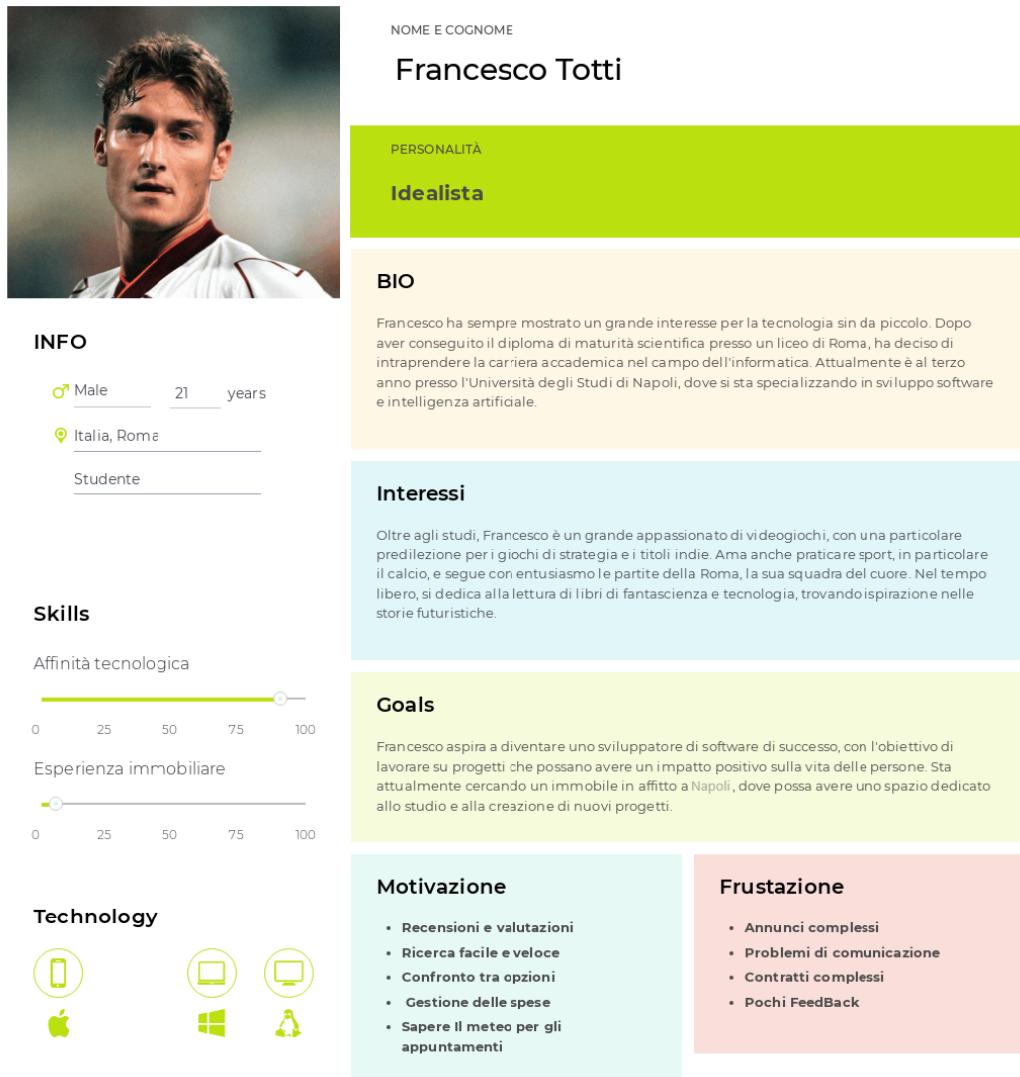


Figura 2.7: Personas - Pendolare

2.2.8 Analisi Personas

Amministratori in Generale

Da entrambe le personas si evince che desiderano avere il pieno controllo nella gestione dei loro agenti immobiliari: rimuoverli, aggiungerli e assegnare loro i lavori, avvalendosi dell'aiuto di amministratori delegati. Questi ultimi, a loro volta, possono essere gestiti direttamente dagli amministratori principali.

Boris Yellnikoff - Amministratore con esperienza

Dalla persona di Boris Yellnikoff si evince che ha bisogno di un'interfaccia poco meccanica. Essendo molto pignolo, richiede un'elevata usabilità. Diretto e pragmatico, manifesta l'esigenza di una gestione della comunicazione semplice, veloce e affidabile; per questo motivo, immaginiamo un sistema che garantisca email puntuali e prive di spam. Gestendo un'importante azienda, sicuramente non vuole compromettere la propria reputazione. Di conseguenza, utilizzerebbe uno strumento solo se lo ritiene veramente affidabile e sicuro, preferendo un software serio e privo di fronzoli. Si riscontra inoltre una chiara volontà di gestire l'agenzia con strumenti adeguati. Essendo molto fiscale, potrebbe richiedere anche analisi dettagliate delle vendite, report e statistiche sulle performance generali.

Leandro Esposito - Amministratore junior

Evinciamo dalla Biografia di Leandro che è un ragazzo giovane che non ha molta esperienza nel settore immobiliare. Essendo un ragazzo giovane punta a una chiara comunicazione, nella nostra applicazione ci deve essere un comparto email veloce e funzionale. Inoltre deve esserci un ottimo sistema di notifiche, quest'ultimo deve mantenere aggiornati gli utenti dell'applicazione, in oltre data l'inesperienza si deve avere delle notifiche mirate e decidere su cosa ricevere le varie notifiche. Leandro punta molto sull'eco-sostenibile quindi la nostra app potrebbe implementare delle funzionalità rispetto al settore eco-sostenibile. Inoltre evinciamo anche che Leandro voglia un ottimo supporto dai parte dei sviluppatori, quindi una predisposizione a cambiamenti e miglioramenti

Sintesi

- Per Leandro è importante:
 1. Funzionalità per l'eco-sostenibilità
 2. Comparto email
 3. Sistema gestione delle notifiche
 4. Supporto del applicazione e aggiornamenti veloci

Vincent Vega - Agente immobiliare novello

Considerando la sua inesperienza, è necessario creare un ambiente accogliente che non lo faccia sentire distante dalle sue funzioni, quindi dobbiamo fare attenzione a bilanciare la semplicità con la possibilità di gestire tutte le sue attività. Da quanto traspare, Vincent preferirebbe un calendario organizzato e preciso per le visite, in modo da evitare errori dovuti all'inesperienza. È importante che il sistema offra una modalità semplice per accettare o rifiutare un appuntamento richiesto. In particolare, Vincent sottolinea l'importanza fondamentale delle notifiche, che quindi dovranno avere un ruolo cruciale all'interno del prodotto software.

Laura Bianchi - Agente immobiliare navigato

Laura desidera un'applicazione che non solo semplifichi la gestione quotidiana del lavoro, ma che le offra anche un supporto concreto nel raggiungimento dei suoi obiettivi di crescita professionale. Per questo, l'interfaccia deve essere progettata in modo intuitivo, così da ridurre al minimo il tempo necessario

per apprendere le sue funzionalità. La piattaforma deve integrarsi perfettamente nel flusso di lavoro di Laura, garantendo rapidità e precisione, elementi cruciali per un'agente immobiliare che si destreggia tra numerosi impegni e clienti.

Un aspetto centrale è la capacità dell'applicazione di migliorare la presentazione degli immobili. Per Laura, che valorizza l'estetica e il design, la possibilità di caricare immagini di alta qualità e personalizzare gli annunci è fondamentale per attirare l'attenzione dei potenziali acquirenti. Allo stesso tempo, la gestione dei dati deve essere completa e accurata: ogni informazione sugli immobili deve essere accessibile in tempo reale, senza lasciare spazio a lacune che potrebbero compromettere il processo decisionale dei clienti.

Mia e Sebastian - Giovane coppia sposata

Quello che sicuramente emerge dalla coppia è la loro inesperienza, da cui si possono trarre la maggior parte delle considerazioni. Innanzitutto, immaginiamo che desiderino avere a disposizione tutti gli strumenti possibili per confrontare un immobile rispetto a un altro. Pertanto, potremmo pensare a ricerche personalizzabili, immaginando un filtro per ricerche mirate. Essendo giovani, dobbiamo considerare la loro inesperienza e la semplicità d'uso, ma al contempo offrire qualcosa di accattivante ed efficiente.

Giovanni Rossi - Anziano

Dalla persona di Giovanni Rossi si evince che il software di housing dovrebbe essere semplice e intuitivo, per adattarsi alle sue limitate competenze tecnologiche, con un'interfaccia chiara e compatibile con i SO più utilizzati come Android. La possibilità di personalizzare le ricerche, risponde al suo desiderio di trovare una casa che più corrisponde alle sue esigenze.

Inoltre gli annunci devono essere accompagnati da immagini dettagliate ed una descrizione precisa e accurata per risparmiare eventuali sorprese spiacevoli e perdite di tempo.

Francesco Totti - Pendolare

Dalla persona di Francesco Totti si evince che è uno studente universitario appassionato di tecnologia e sviluppo software, con uno stile di vita dinamico e in costante movimento. Per questo motivo, l'app deve essere perfettamente funzionante su dispositivi mobili e offrire un'esperienza fluida e veloce.

Data la sua scarsa esperienza nel settore immobiliare, è fondamentale che l'interfaccia sia semplice, con guide chiare e strumenti intuitivi. Il sistema deve facilitare la comparazione degli immobili, con recensioni, valutazioni e una gestione chiara delle spese. Le notifiche devono essere precise e puntuali, utili anche per conoscere il meteo prima di un appuntamento.

È importante che la piattaforma supporti una rapida ricerca e offra feedback concreti e leggibili. Francesco si scoraggia davanti a contenuti poco chiari o a contratti complicati, quindi è necessario evitare annunci e processi complessi.

Sintesi

- **Motivazioni:** Recensioni, gestione delle spese, facilità di confronto, meteo appuntamenti
- **Frustrazioni:** Annunci e contratti complessi, scarsa comunicazione, pochi feedback
- **Necessità principali:**
 1. Interfaccia semplice e mobile-friendly
 2. Sistema di notifiche intelligente
 3. Confronto opzioni immobiliari
 4. Supporto alla gestione economica

2.3 Casi d'uso - Use Case Diagram

Un passo fondamentale nell'Ingegneria dei Requisiti (RE) è la formalizzazione dei requisiti. Sebbene non esista un approccio universalmente migliore, la scelta della tecnica più adatta dipende dal contesto specifico del progetto. Nel caso di un'applicazione gestionale come questa, un metodo che garantisce un buon equilibrio tra costo e beneficio è certamente l'utilizzo dei diagrammi UML (Unified Modeling Language), in particolare degli Use Case Diagram. Questi strumenti consentono di rappresentare in modo chiaro e intuitivo le interazioni tra gli attori coinvolti e il sistema, facilitando la comprensione e la comunicazione dei requisiti tra le parti interessate.

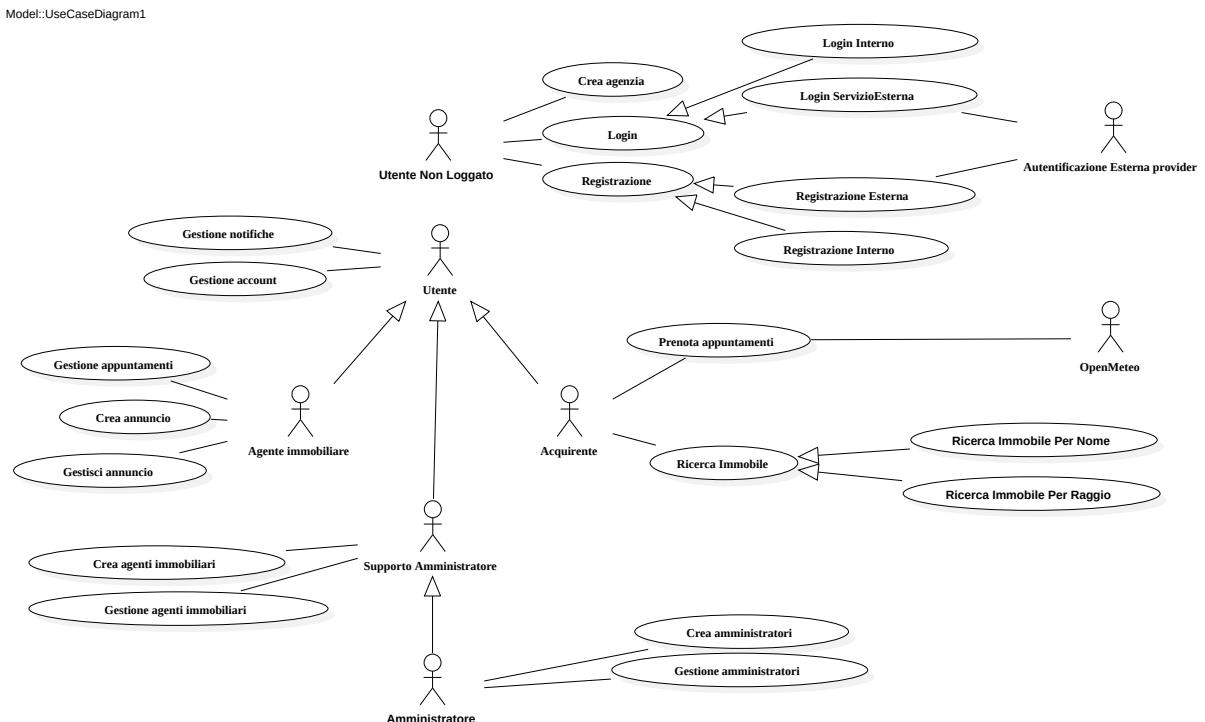


Figura 2.8: Use Case Diagram - DietiEstate25

2.3.1 Attori individuati

- **Utente non loggato**: rappresenta un individuo che non è ancora registrato nel sistema o non ha effettuato l'accesso. Questo scenario si verifica immediatamente all'apertura del software, quando non è possibile identificare l'utente né determinare le sue intenzioni. Di conseguenza, le uniche opzioni disponibili per l'utente in questo stato iniziale sono la registrazione di un nuovo account o il login, al fine di accedere alle funzionalità del sistema in base al proprio ruolo. Inoltre, viene fornita la possibilità di avviare il processo di creazione di una nuova agenzia immobiliare, per coloro che desiderano iniziare come nuovi amministratori. Queste scelte garantiscono che ogni utente venga indirizzato correttamente verso il percorso più adeguato alle sue necessità.

- **Utente** : rappresenta un qualsiasi individuo che ha effettuato con successo l'accesso al sistema. In base all'analisi dei requisiti derivati dalla traccia, tutti gli utenti, indipendentemente dal loro ruolo specifico, condividono alcune funzionalità di base. Tra queste, la possibilità di gestire il proprio account, inclusa la modifica di informazioni personali e delle credenziali, e l'accesso a un sistema di notifiche. Queste notifiche permettono agli utenti di rimanere aggiornati su eventi rilevanti o comunicazioni importanti all'interno della piattaforma.
- **Acquirente** : rappresenta un potenziale utilizzatore del software, il cui obiettivo principale è cercare immobili di interesse. Questo attore utilizza il sistema per filtrare e individuare proprietà che soddisfano i suoi criteri, come posizione, prezzo, e caratteristiche specifiche. Una volta identificati gli immobili di interesse, l'acquirente può prenotare visite direttamente tramite il software, coordinandosi con l'agente immobiliare assegnato. La pianificazione degli appuntamenti tiene conto sia della disponibilità dell'agente sia di eventuali fattori esterni, come le condizioni climatiche, che potrebbero influenzare l'efficacia della visita. In questo modo, il sistema facilita l'interazione tra acquirenti e agenti, ottimizzando il processo di ricerca e prenotazione.
- **Agente Immobiliare** : responsabile della creazione degli annunci immobiliari, operando sotto la supervisione e le direttive della propria agenzia immobiliare. Gli annunci, una volta pubblicati, possono essere modificati o aggiornati in qualsiasi momento dall'agente. Inoltre, sulla base della conoscenza del dominio, l'agente immobiliare svolge il ruolo di intermediario tra l'acquirente e l'agenzia. Questo implica che l'agente deve essere in grado di gestire richieste di appuntamenti provenienti dagli acquirenti interessati.
- **Supporto Amministratore** : rappresenta una figura di assistenza all'interno della gerarchia gestionale dell'agenzia immobiliare, operando come aiutante degli amministratori. Questo attore ha il compito specifico di definire e gestire gli agenti immobiliari all'interno del sistema.
- **Amministratore** : possiede un ruolo di massimo controllo e autorità all'interno del sistema. Oltre a poter svolgere tutte le funzioni assegnate al Supporto Amministratore, l'Amministratore ha la capacità esclusiva di gestire la creazione di altri amministratori e supporti amministratori e gestione del reparto amministrazione. La scelta di consentire la presenza di più amministratori anziché limitarsi a uno solo è motivata da esigenze operative e organizzative. In particolare, si considera che, nel contesto di un'agenzia X, gli amministratori rappresentino i proprietari dell'azienda. In caso di società con più soci, è fondamentale garantire a ciascuno di essi gli stessi privilegi e poteri decisionali.

2.4 Mock-Up

Passiamo ora alla presentazione dei mockup a bassa fedeltà (low-fidelity), che rappresentano una prima bozza del design e dell'interfaccia del sistema. Questi mockup sono utili per esplorare rapidamente idee, testare i flussi principali e raccogliere feedback preliminari prima di procedere con le versioni più dettagliate e finalizzate. Use Case scelti :

- Ricerca Immobile
- Crea Agenzia

2.4.1 Ricerca Immobile Per Cittá

Il primo mockup selezionato rappresenta il processo di ricerca di un immobile. Questo flusso inizia con la fase di individuazione basata sulla localizzazione, che può essere effettuata attraverso una ricerca testuale o tramite la definizione di un raggio geografico. Successivamente, è possibile affinare i risultati applicando filtri personalizzati in base a parametri specifici, come il prezzo, la tipologia dell'immobile o la metratura desiderata. Infine, il sistema visualizza i risultati in modo chiaro e organizzato, consentendo all'utente di esplorare le opzioni disponibili in maniera intuitiva.

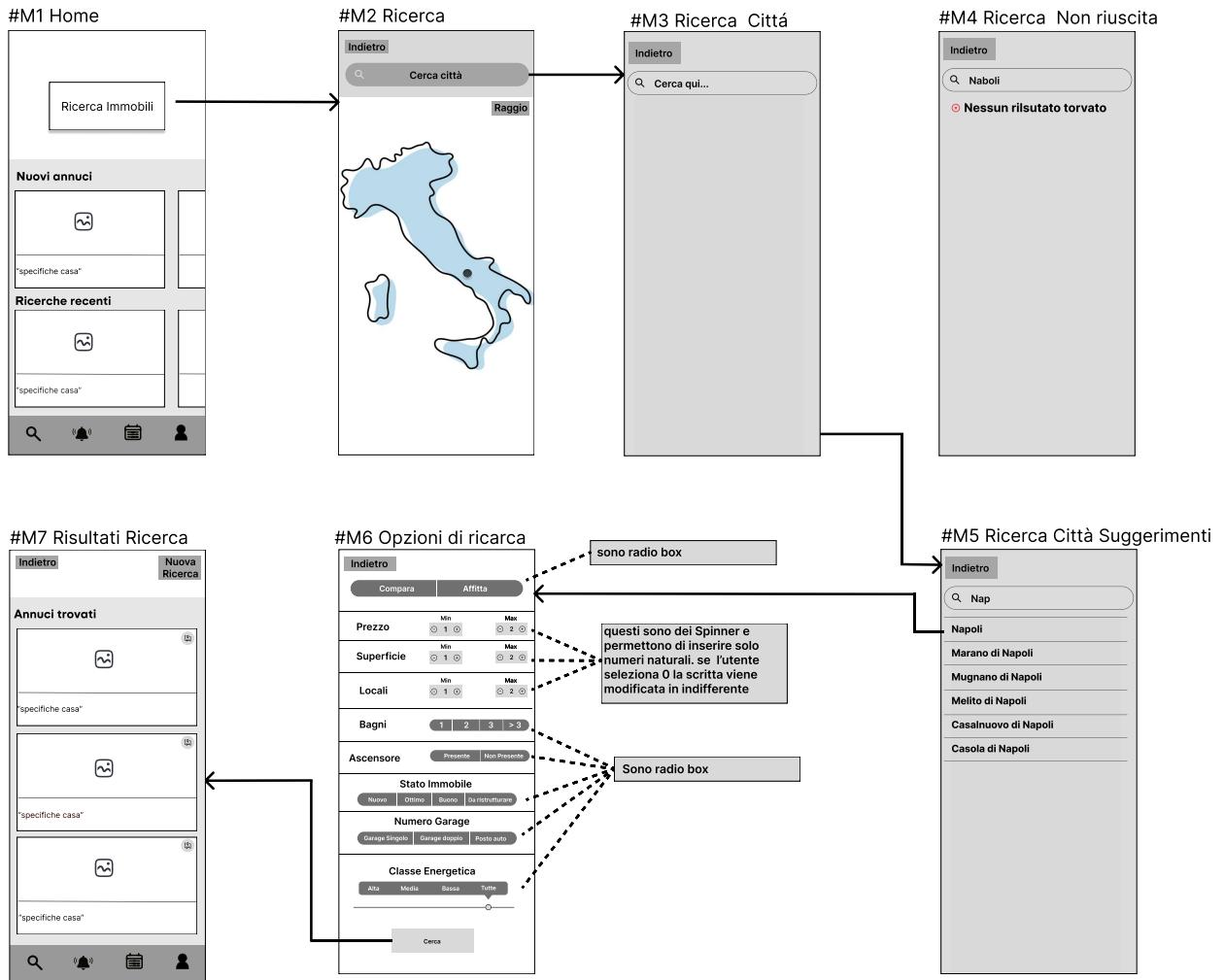


Figura 2.9: Mock-Up Low Fidelity - Ricerca Immobile Per Nome

2.4.2 Crea Agenzia

Il secondo mockup selezionato rappresenta il flusso di creazione di un'agenzia immobiliare. Il processo inizia dalla schermata principale del sistema, da cui l'utente accede alla sezione dedicata alla creazione dell'agenzia. Qui, viene presentato un modulo da compilare con tutti i dati richiesti, come il nome dell'agenzia, la partita IVA e i contatti. Una volta inserite correttamente tutte le informazioni, il sistema verifica i dati e, in caso di esito positivo, consente la creazione dell'agenzia. Al termine del processo, vengono fornite le credenziali di accesso temporanee e una conferma visiva del successo dell'operazione.

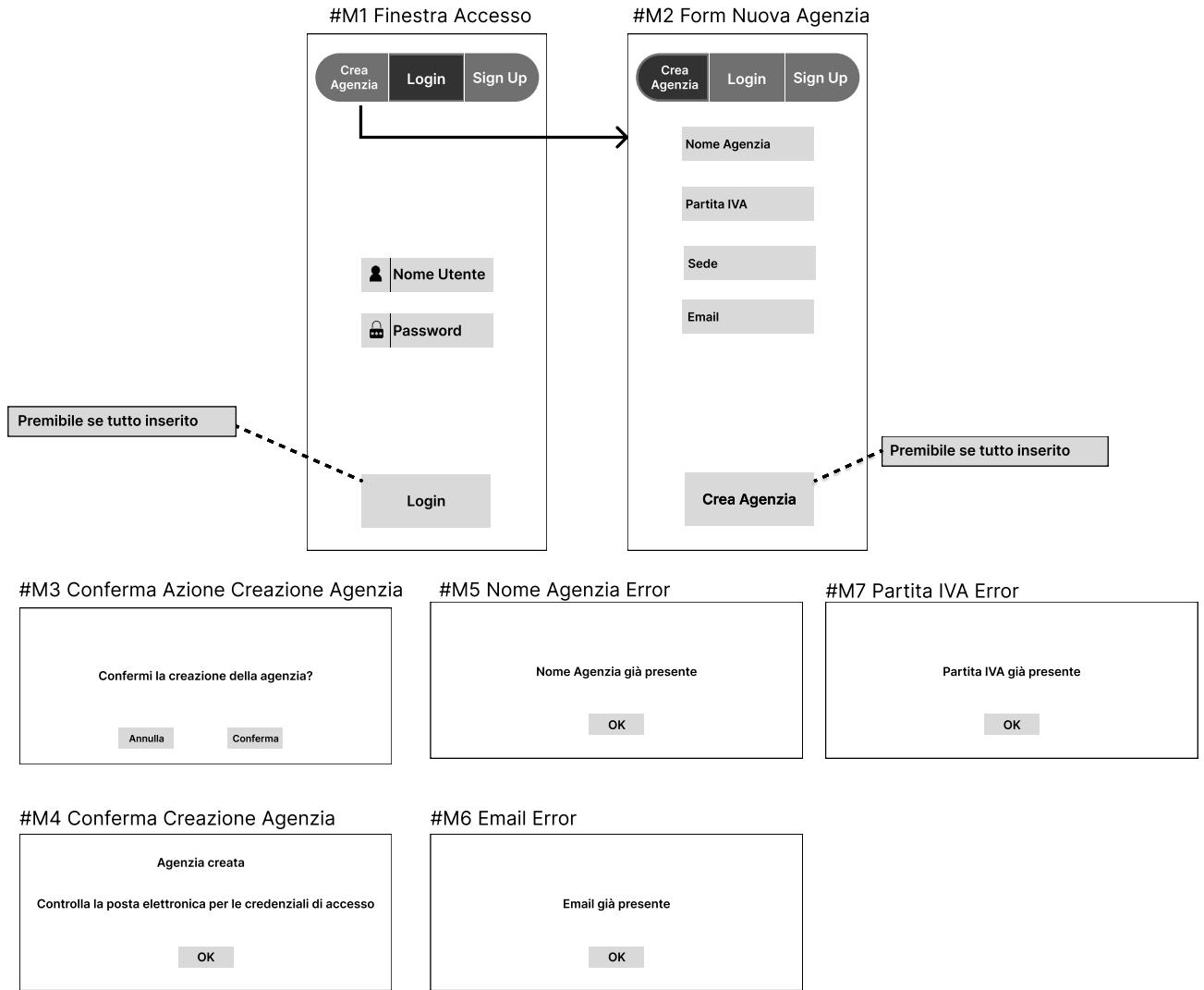


Figura 2.10: Mock-Up Low Fidelity - Crea Agenzia

2.4.3 Gestisci Amministratori

Esso rappresenta il mock-up a bassa fedeltà del flusso di gestione degli amministratori. Il processo è articolato in quattro schermate principali:

- M1 – Home Admin : punto di accesso principale da cui è possibile navigare alla sezione “Gestione Amministratori”.
- M2 – Gestione Admin : elenco degli amministratori registrati, con possibilità di rimuovere/promuovere/regredire uno specifico utente tramite un menu contestuale.
- M3 – Conferma Azione : finestra di dialogo che chiede conferma dell’azione di rimozione.
- M4 – Azione Completata: messaggio di feedback che notifica il successo dell’operazione.

Questo mock-up illustra in modo semplice e intuitivo il flusso di gestione degli utenti con ruolo amministrativo.

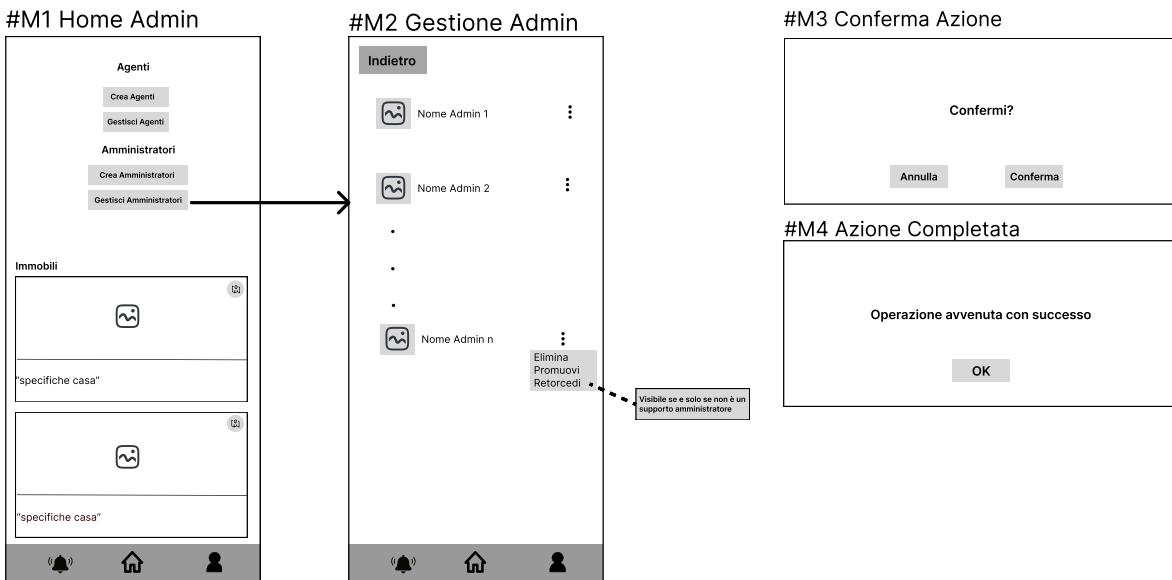


Figura 2.11: Mock-Up Low Fidelity - Gestisci Amministratori

2.4.4 Crea Amministratori

Essa è il mock-up low fidelity del flusso di creazione di un nuovo amministratore nell’applicazione. Il processo inizia dalla schermata M1 Home, da cui l’utente può accedere alla sezione ”Crea Amministratori”. La schermata M2 Crea Amministratore consente l’inserimento dell’indirizzo email del nuovo utente e l’opzione per assegnargli il ruolo di supporto amministratore.

Il flusso include:

- M3 Email Error: messaggio di errore in caso l'email sia già registrata.
- M4 Azione Completata: conferma visiva del successo dell’operazione.
- M5 Conferma Azione: dialogo di conferma prima di procedere con la creazione.

Il mock-up guida l’utente attraverso un’interazione semplice e lineare, con validazioni e messaggi di feedback chiari.

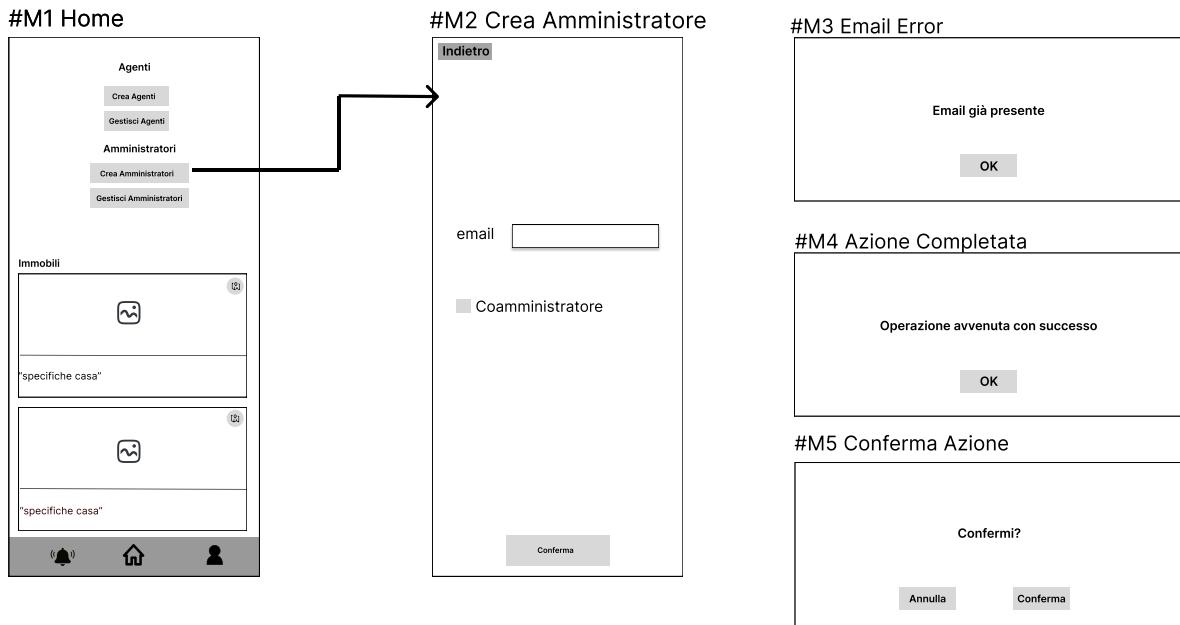


Figura 2.12: Mock-Up Low Fidelity - Crea Amministratori

2.5 Tabelle Cockburn

In questa sezione della documentazione, espliceremo i nostri use case selezionati utilizzando il formato della fully dressed description, conformemente allo standard definito dalla tabella di Cockburn.

2.5.1 Cockburn UC1 - Ricerca Immobile Per Città

Elemento	Descrizione
Use Case 1	Ricerca Immobile per città
Goal in Context	Permettere all'Acquirente di poter cercare immobili filtrando la ricerca
Preconditions	Acquirente deve essere autenticato
Success End Condition	Il sistema salva in un log la ricerca dell'Acquirente
Failed End Condition	Il sistema non tiene traccia della ricerca dell'Acquirente
Primary Actor	Acquirente
Trigger	Preme “Ricerca Immobili” nel #M1 Home

Tabella 2.1: Cockburn UC1 - Ricerca Immobile Per Nome

Step	Acquirente	Sistema
1	Trigger	
2		Mostra #M2
3	Clicca il pulsante ”Cerca città”	
4		Mostra #M3
5	Compila form	
6		Mostra #M5 con città suggerite
7	Clicca il risultato	
8		Mostra #M6
9	Seleziona filtri e clicca “Cerca”	
10		Mostra #M7 e termina Use Case

Tabella 2.2: Cockburn UC1 - Main Scenario

Extension A: Cerca una città o comune inesistente

Step	Acquirente	Sistema
6a		Mostra #M4
7a	Ricompila	
8a		Ritorna a Step 6 del main scenario

Tabella 2.3: Cockburn UC1 - Extension A : cerca una città o comune inesistente

Extension B: Clicca “Indietro” in #M4, #M5 o #M3

Step	Acquirente	Sistema
5/6/7b	Clicca “Indietro”	
6/7/8b		Ritorna a Step 2 del main scenario

Tabella 2.4: Cockburn UC1 - Extension B : clicca “Indietro” in #M4, #M5 o #M3

Extension C: Clicca “Indietro” in #M6

Step	Acquirente	Sistema
9c	Clicca “Indietro”	
10c		Ritorna a Step 4 del main scenario

Tabella 2.5: Cockburn UC1 - Extension C : clicca “Indietro” in #M6

Extension D: Clicca “Indietro” in #M2

Step	Acquirente	Sistema
3d	Clicca “Indietro”	
4d		Mostra #M1 e termina UC

Tabella 2.6: Cockburn UC1 - Extension D : clicca “Indietro” in #M2

2.5.2 Cockburn UC2 - Crea Agenzia

Elemento	Descrizione
Use Case 2	Crea Agenzia
Goal in Context	Permettere la creazione di una nuova agenzia nel sistema.
Preconditions	Nessuna.
Success End Condition	Il sistema tiene traccia della nuova agenzia e invia le credenziali temporanee via email.
Failed End Condition	Il sistema non tiene traccia della nuova agenzia; nessuna agenzia viene creata; l'utente non riceve credenziali via email.
Primary Actor	Utente non registrato.
Trigger	L'utente non registrato clicca su "Crea Agenzia" in #M1.

Tabella 2.7: Cockburn UC2 - Crea Agenzia

Main Scenario

Step	Utente non registrato	Sistema
1	Trigger	
2		Mostra #M2.
3	Compila il form	
4	Clicca il pulsante "Crea Agenzia"	
5		Mostra #M3.
6	Clicca conferma	
7		Invia email con le credenziali.
8		Mostra #M4.
9	Premere "OK"	
10		Mostra #M1 e termina Use Case.

Tabella 2.8: Cockburn UC2 - Main Scenario

Extensions

Extension A: Il nome dell'agenzia è già stato utilizzato

Step	Utente non registrato	Sistema
5a		Mostra #M5.
6a	L'utente clicca "OK"	
7a		Ritorna a Step 2 del Main Scenario.

Tabella 2.9: Cockburn UC2 - Extension A : Il nome dell'agenzia è già stato utilizzato

Extension B: La partita IVA è già stata registrata

Step	Utente non registrato	Sistema
5b		Mostra #M7.
6b	L'utente clicca "OK"	
7b		Ritorna a Step 2 del Main Scenario.

Tabella 2.10: Cockburn UC2 - Extension B : la partita IVA è già stata registrata

Extension C: L'email è già stata utilizzata

Step	Utente non registrato	Sistema
5c		Mostra #M7.
6c	L'utente clicca "OK"	
7c		Ritorna a Step 2 del Main Scenario.

Tabella 2.11: Cockburn UC2 - Extension C : l'email è già stata utilizzata

Extension D: L'utente annulla la conferma della creazione dell'agenzia

Step	Utente non registrato	Sistema
6d	L'utente clicca "Annulla"	
7d		Ritorna a Step 2 del Main Scenario.

Tabella 2.12: Cockburn UC2 - Extension D : l'utente annulla la conferma della creazione dell'agenzia

2.5.3 Cockburn UC3 - Gestione Amministratori

Elemento	Descrizione
Use Case 3	Gestione Amministratori
Goal in Context	Permettere agli amministratori di gestire gli utenti amministrativi
Preconditions	Amministratore deve essere autenticato
Success End Condition	il co-amministratore viene aggiornato
Failed End Condition	Il sistema non tiene traccia delle modifiche apportate al co-amministratore
Primary Actor	Amministratore
Trigger	Preme “Gestisce Amministratori” nel #M1 Home admin

Tabella 2.13: Cockburn UC3 - Gestione Amministratori

Step	Amministratore	Sistema
1	Trigger	
2		Mostra #M2.
3	clicca l’icona del menu affianco al amministratore che si vuole modificare	
4	Clicca il pulsante ”Elimina/promuovi/retrocedi” nel menu a tendina	
5		Mostra #M3.
6	Clicca conferma	
7		aggiorna l’amministratore.
8		Mostra #M4.
9	Premere “OK”	
10		Mostra #M1 e termina Use Case.

Tabella 2.14: Cockburn UC3 - Main Scenario

Extensions

Extension A: Clicca Annulla

Step	Utente non registrato	Sistema
6a	L'utente clicca "Annulla"	
7a		Ritorna a Step 2 del Main Scenario.

Tabella 2.15: Cockburn UC3 - Extension A : clicca il tasto annulla

Extension B: Clicca Indietro

Step	Utente non registrato	Sistema
3b	L'utente clicca "Indietro".	
4b		Mostra #M1 e termina Use Case.

Tabella 2.16: Cockburn UC3 - Extension B : clicca il tasto indietro

2.5.4 Cockburn UC4 - Crea Amministratore

Elemento	Descrizione
Use Case 4	Crea Amministratore
Goal in Context	Permettere agli amministratori di creare altri amministratori
Preconditions	Amministratore deve essere autenticato
Success End Condition	il sistema tiene traccia del nuovo Amministratore
Failed End Condition	Il sistema non tiene traccia del nuovo Amministratore
Primary Actor	Amministratore
Trigger	Preme "Crea Amministratori" nel #M1 Home admin

Tabella 2.17: Cockburn UC3 - Crea Amministratore

Step	Amministratore	Sistema
1	Trigger	
2		Mostra #M2.
3	compila il form	
4	Clicca conferma	

Step	Amministratore	Sistema
5		Mostra #M5
6	Clicca Conferma	
7		Mostra #M4
8	Clicca "OK" e termina Use Case.	

Tabella 2.18: Cockburn UC3 - Main Scenario

Extensions

Extension A: Clicca Indietro

Step	Amministratore	Sistema
3a	L'utente clicca "Indietro".	
4a		Mostra #M1 e termina Use Case.

Tabella 2.19: Cockburn UC4 - Extension A : clicca il tasto indietro

Extension B: Email già presente

Step	Amministratore	Sistema
5b		Mostra #M3
6b	Clicca "OK"	
7b		Ritorna a Step 2 del Main Scenario

Tabella 2.20: Cockburn UC4 - Extension B : Email già presente

Extension C: Clicca Annulla

Step	Utente non registrato	Sistema
6c	L'utente clicca "Annulla"	
7c		Ritorna a Step 2 del Main Scenario.

Tabella 2.21: Cockburn UC4 - Extension C : clicca il tasto annulla

2.6 Requisiti non-funzionali e di dominio

Abbiamo raggiunto una fase del progetto in cui ci concentriamo sull'identificazione dei requisiti non funzionali.

2.6.1 Requisiti non-funzionali

- **Prestazioni :**

- Il sistema deve rispondere con un tempo massimo di 10 secondi (perché la performance è necessaria per un'ottima fruibilità del sistema)
- Il sistema deve supportare almeno 100 utenti simultaneamente senza degrado significativo delle prestazioni.

- **Scalabilità :**

- Ogni componente del sistema deve essere progettato per scalare orizzontalmente o (disgiunzione inclusiva) verticalmente. (perché il sistema deve supportare eventuali carichi maggiori)

- **Affidabilità :**

- Il sistema deve avere un'affidabilità del 99.99% ~ ovvero 3 giorni 15 ore di down-time durante tutto l'anno (perché più agenzie possono basare il loro business sul nostro sistema)

- **Usabilità :**

- Il sistema deve avere un tasso di errore di usabilità di 10 errori all'ora massimo (perché è richiesto un uso intuitivo, rapido e piacevole dell'interfaccia)

- **Sicurezza :**

- Il sistema deve garantire la crittografia dei dati sensibili (perché i dati sensibili degli utilizzatori devono essere protetti da malintenzionati)

- **Interoperabilità:**

- Il sistema deve essere in grado di integrarsi con tecnologie esterne per garantire i servizi.

2.6.2 Requisiti di dominio

- **Gestione delle Inserzioni:**

- Le inserzioni immobiliari devono includere dettagli completi come foto, descrizione, prezzo, superficie, posizione geografica, locali, classe energetica, e servizi aggiuntivi (ascensore, garage, ecc.).
- Gli immobili devono essere classificati in categorie principali ("vendita" o "affitto"), con possibile espansione futura per "affitti brevi" o "case vacanze".

- **Posizione Geografica:**

- Ogni immobile deve essere associato a una posizione esatta sulla mappa interattiva.
- La ricerca deve includere un filtro geografico (per comune, città o raggio specificato su mappa).

- **Gestione degli Utenti:**

- Esistono tre tipi di utenti principali: amministratori (agenzia), agenti immobiliari e clienti.
 - I profili degli agenti devono includere informazioni professionali, foto.

- **Ricerca Immobili:**

- Gli utenti possono filtrare gli immobili usando le caratteristiche di quest'ultimi.

- **Prenotazione delle Visite:**

- Gli utenti possono prenotare visite a immobili disponibili.
 - Gli agenti immobiliari possono confermare o rifiutare prenotazioni, con possibilità di notifiche e gestione visuale tramite calendario.

- **Notifiche e Preferenze:**

- Gli utenti possono ricevere notifiche per eventi specifici (es. nuove inserzioni, conferma/ri-fiuto prenotazioni).
 - Le notifiche devono essere configurabili dall'utente (attivazione/disattivazione per ogni categorie).

- **Previsioni Meteo:**

- Durante la prenotazione di una visita, il sistema deve mostrare le previsioni meteo per aiutare nella pianificazione.

- **Cronologia delle Ricerche:**

- Ogni utente deve poter accedere alla cronologia delle proprie ricerche per rieseguire rapidamente query precedenti.

Capitolo 3

System Design

In questa sezione viene descritta l'architettura del sistema, evidenziando le principali componenti, le loro interazioni e le scelte progettuali effettuate. Verranno analizzate le tecnologie più adatte alle nostre esigenze e formalizzati il Class Diagram e il Sequence Diagram.

3.1 Architettura

Passiamo dunque alla descrizione dell'architettura passo dopo passo, elencando le scelte effettuate, i vantaggi e gli svantaggi, evidenziando i punti di forza e le caratteristiche su cui dobbiamo ancora lavorare.

3.1.1 Client-Server

Abbiamo bisogno :

- **Applicazione di rete** : ci serve una struttura che permetta ai differenti moduli di comunicare fra di loro tramite internet, quindi un sistema che permetta la distributività.
- **Sicurezza** : come specificato nel requisito non funzionale in [2.6.1](#) ci dice che i vari dati sensibili che navigano e risiedono tra i moduli siano protetti da malintenzionati.(esclude peer-to-peer).
- **Scalabilità**: come specificato nel requisito non funzionale in [2.6.1](#) abbiamo bisogno di una struttura che permetta di bilanciare il numero di risorse allocate in base al numero delle richieste ricevute, per supportare eventuali picchi di richieste.
- **Prestazioni** : come specificato nel requisito non funzionale in [2.6.1](#) abbiamo bisogno di una architettura che permetta una gestione semplice delle prestazioni (esclude peer-to-peer).

Abbiamo scelto l'architettura client-server, poiché è concepita per la realizzazione di applicazioni di rete. Inoltre, la teoria ci indica che questa architettura offre vantaggi in termini di sicurezza, scalabilità e prestazioni.

3.1.2 N-Tier

Abbiamo bisogno :

- **Modularità** : necessitiamo di una architettura che ci permetta di scindere vari moduli rendendoli indipendenti.
- **Facilità nella manutenzione** : vogliamo che un qualsiasi cambio di necessità non richieda la completa riorganizzazione di tutti i moduli del sistema (esclude architettura a repository).
- **Mascheramento**: vogliamo che i differenti moduli possano comunicare solo con l'interfaccia dei moduli richiesti (esclude MVC).

L'architettura a N-Tier è un architettura molto adatta al nostro dominio e ci permette di avere queste caratteristiche, dividendo in 4 livelli fondamentali il nostro sistema.

Livello Presentazione

Questo livello si occupa di gestire l'interazione tra l'utente finale e il sistema. Il livello di presentazione (front-end) utilizza le API messe a disposizione dal backend per permettere all'utente di interagire con le funzionalità offerte dal sistema in modo semplice e intuitivo. Ha il compito di visualizzare i dati ricevuti dal backend e di raccogliere gli input dell'utente, inviandoli poi al livello Business tramite un punto di accesso centrale per i moduli per l'elaborazione.

Livello API Gateway

Questo livello ha il compito di instradare le richieste verso il modulo corretto del livello business. Grazie a questo meccanismo, tutti i moduli possono essere raggiunti tramite un'unica porta su un solo host, semplificando la comunicazione e l'accesso ai servizi. Il gateway funge da punto di ingresso centralizzato, migliorando la gestione delle richieste e l'organizzazione dell'architettura; inoltre si assicura che gli utenti siano autenticati tramite JWT.

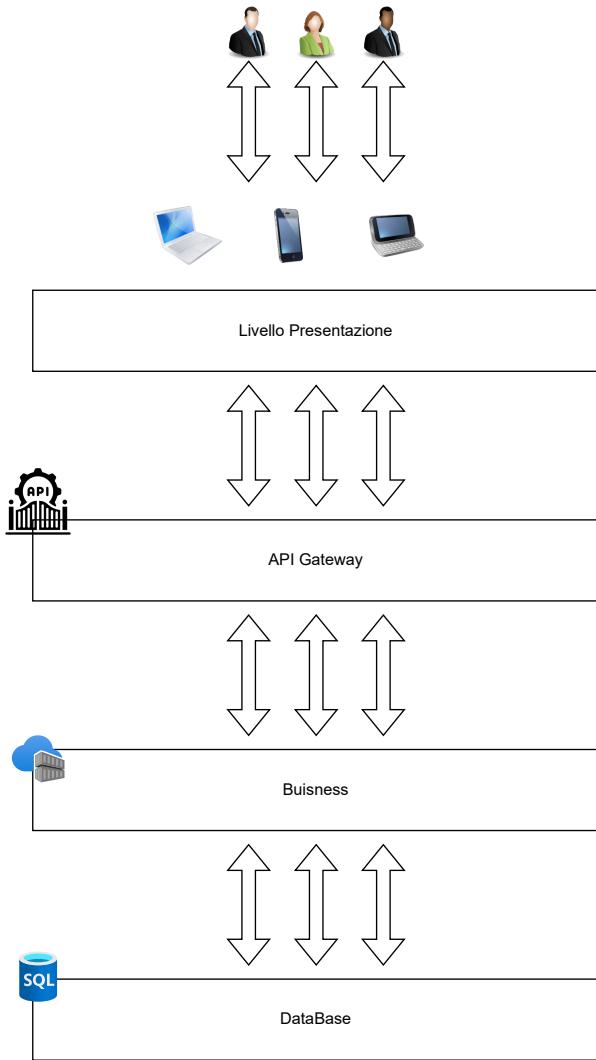
Livello Business

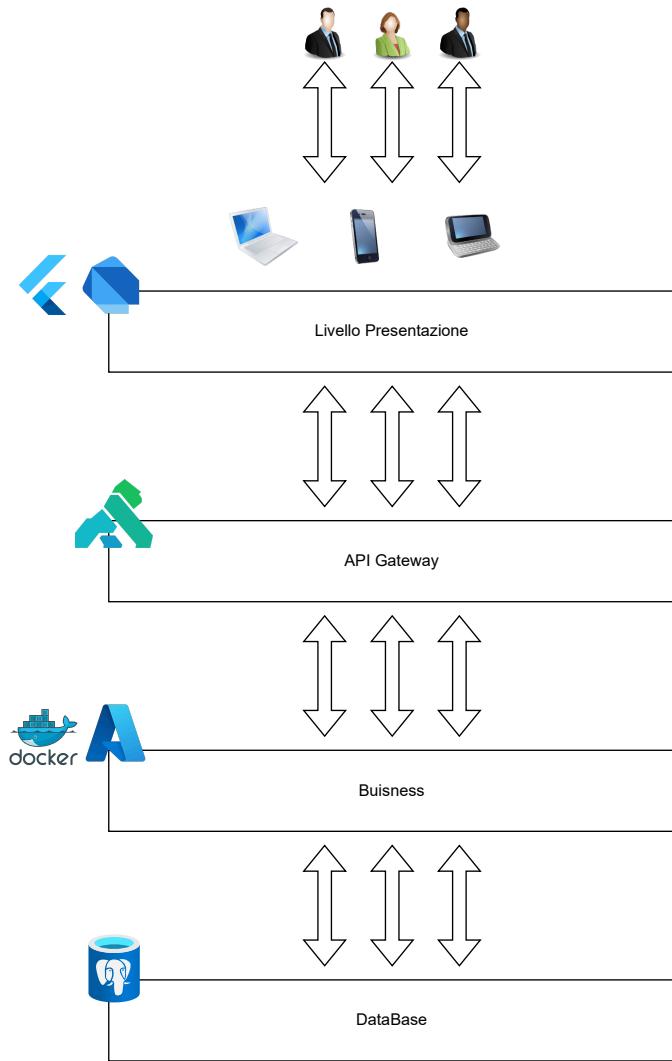
Questo livello rappresenta il cuore logico dell'applicazione. È suddiviso in diversi moduli, ciascuno con una responsabilità specifica (ad esempio: un modulo per la gestione dell'autenticazione, un modulo per le notifiche, ecc.). Una caratteristica fondamentale di questo livello è che ogni modulo è autosufficiente. I vantaggi di questo approccio sono:

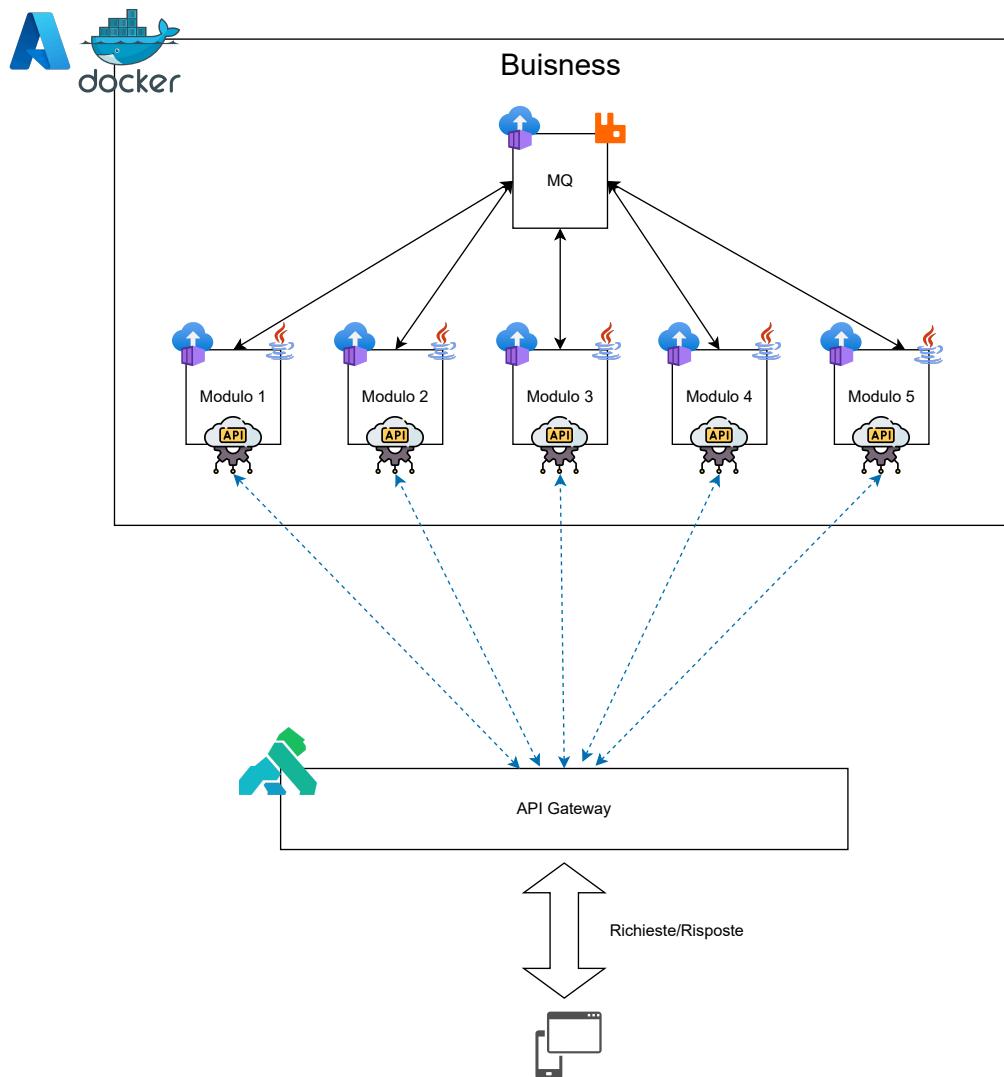
- I vari moduli sono indipendenti, ovvero progettati per funzionare autonomamente rispetto agli altri. Questo approccio garantisce una maggiore **modularità**, una migliore **manutenibilità** e una più semplice **scalabilità del sistema**.
- Ogni modulo è eseguito all'interno di un container, il che consente un controllo preciso sulle risorse assegnate. È possibile definire dei limiti minimi e massimi di risorse per ciascun modulo, assicurando così un **livello minimo di qualità del servizio (minimo QoS)** e prevenendo che un singolo modulo **consumi eccessive risorse** a discapito degli altri.

Livello Persistenza dati

Questo livello si occuperà di conservare i dati e preservarli.







3.2 Tecnologia

- **Logica di presentazione :**

- **Flutter/Dart:** Abbiamo scelto questa tecnologia per sviluppare il livello di presentazione dell'applicazione. In particolare, utilizzeremo il framework Flutter, basato sul linguaggio Dart, poiché consente di scrivere una singola codebase che può essere compilata per molteplici piattaforme. Questa caratteristica ci permette di ottimizzare il processo di sviluppo e di massimizzare l'efficienza nella gestione del livello di presentazione su diversi dispositivi.

- **Persistenza dati :**

- **PostgreSQL:** Abbiamo scelto PostgreSQL come DBMS per il nostro progetto, in quanto si basa su un sistema relazionale che si adatta perfettamente al nostro dominio, il quale si presta al modello a tabelle. La scelta è motivata anche dalla disponibilità di licenze gratuite, a differenza del competitor Oracle, e dall'esperienza già consolidata del team con questa tecnologia.

- **Logica di business :**

- **Java:** Java è un linguaggio orientato agli oggetti estremamente standardizzato e consolidato. Inoltre, gode di un ampio supporto e offre una vasta quantità di materiale open source disponibile, rendendolo una scelta affidabile e versatile.

- **Servizi cloud :**

- **AWS/Azure:** Il Cloud Computing rappresenta un modello di distribuzione delle risorse informatiche che consente l'accesso a servizi tramite internet. Abbiamo scelto il cloud per due caratteristiche principali: la scalabilità, che permette di incrementare le performance in proporzione al numero di risorse allocate, e l'elasticità, che consente di distribuire le risorse in base al carico di lavoro.

- **Comunicazione e sicurezza :**

- **HTTPS:** Abbiamo implementato HTTPS per garantire la sicurezza delle comunicazioni tra client e server, proteggendo i dati da intercettazioni e manomissioni. Inoltre, utilizzeremo HTTPS per la creazione delle API Restful, poiché è uno dei protocolli più utilizzati per la comunicazione web, ampiamente supportato e standardizzato.
 - **JWT (JSON Web Token):** Utilizzato per gestire le sessioni utente in modo sicuro e senza stato, garantendo che i dati non vengano manomessi durante la trasmissione.
 - **Kong:** È un API Gateway open-source utilizzato per il bilanciamento del carico, l'accesso unificato e la gestione della sicurezza (JWT).

- **Gestore delle dipendenze :**

- **Gradle e Maven:** Queste tecnologie ci permettono di gestire con facilità le dipendenze del nostro software, assicurando una build automatica corretta e completa di tutte le dipendenze necessarie. Gradle sarà utilizzato per il livello di presentazione (quindi per le app sviluppate con Flutter), mentre Maven sarà impiegato per la parte di software lato Java.

- **Versionamento del codice :**

- **git :** Abbiamo scelto Git come sistema di versionamento per la sua affidabilità e diffusione nell'industria. Grazie a Git, possiamo gestire il lavoro in team in modo collaborativo, creare branch per sperimentazioni o nuove funzionalità, e tracciare facilmente ogni modifica al codice. Inoltre, la sua compatibilità con piattaforme come GitHub e GitLab semplifica l'integrazione con il nostro flusso di lavoro.

- **Containerizzazione :**

- **Docker:** Docker sarà utilizzato per la distribuzione efficiente dei microservizi, in quanto consente di creare ambienti virtualizzati leggeri e indipendenti per ciascun microservizio. Inoltre, Docker offre numerosi vantaggi, tra cui: indipendenza dall'hardware, facilità di memorizzazione, isolamento dei servizi e gestione semplificata dei container.

3.3 Persistenza dati

La sezione "Persistenza dei dati su un database" descrive il modello di persistenza adottato nel progetto, basato sui diagrammi delle classi e sul loro successivo miglioramento. Si approfondisce come le entità e le loro relazioni siano rappresentate nel database, analizzando l'evoluzione della struttura del modello per ottimizzare la gestione e il recupero dei dati. Vengono inoltre esplorate le scelte architetturali che assicurano integrità, coerenza e performance nel sistema.

3.3.1 Class Diagram

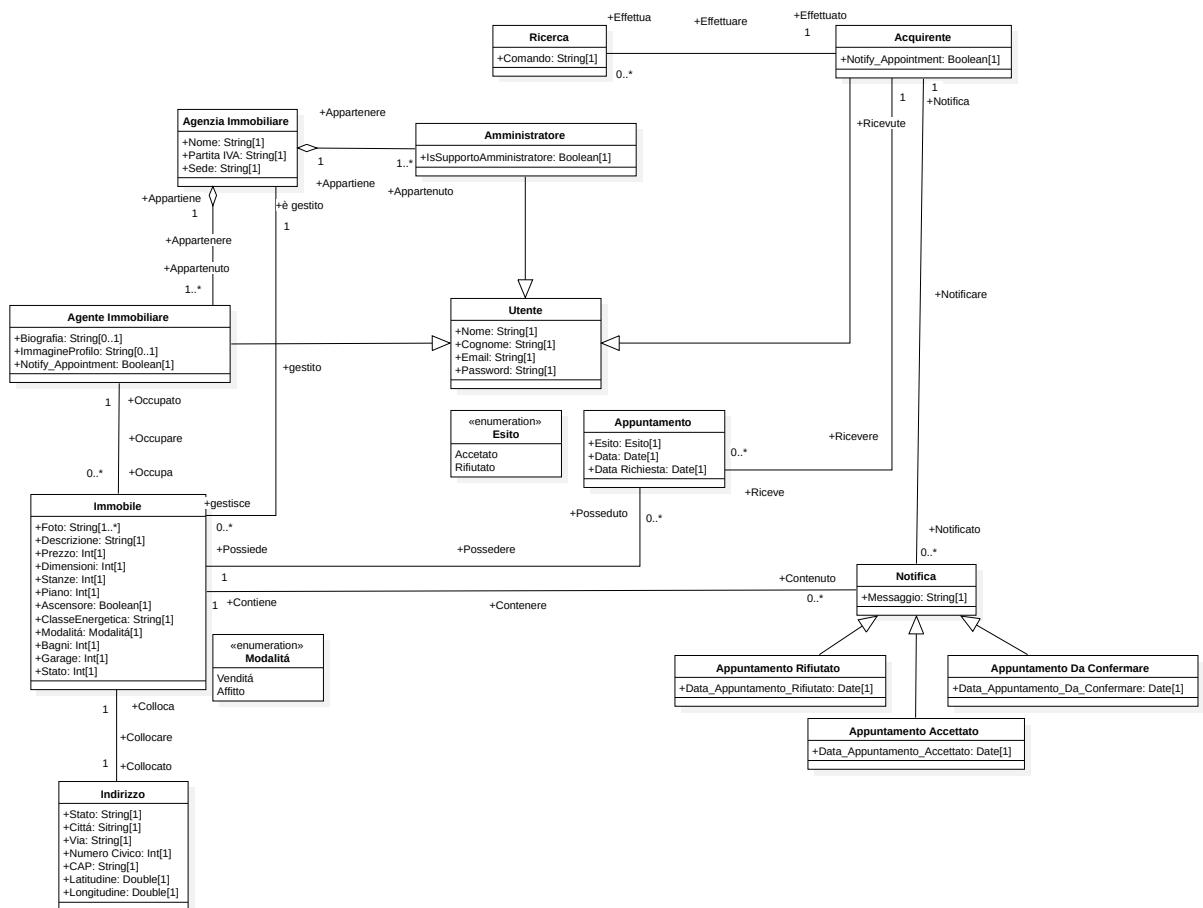


Figura 3.1: Schema Non ristrutturato

3.3.2 Class Diagram Ristrutturato

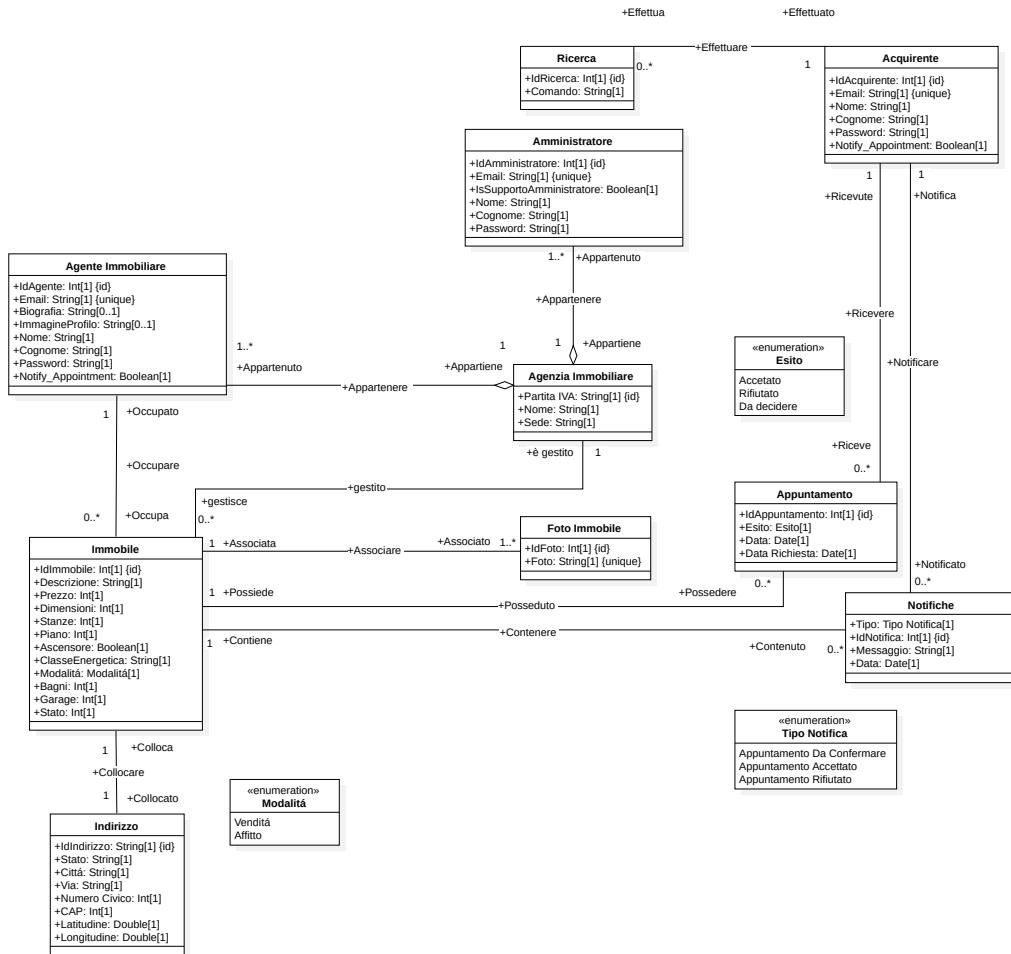


Figura 3.2: Schema Ristrutturato

3.3.3 Modello Logico

Infine possiamo raggiungere la parte finale di questa progettazione, ovvero la parte dello schema logico.
Indicazioni :

- Gli attributi sottolineati sono chiavi primarie.
 - Gli attributi doppiamente sottolineati sono chiavi esterne.
 - Gli attributi con * possono essere NULL
-

Agenzia Immobiliare (Partita IVA, Nome, Sede)

Agente Immobiliare (IdAgente, Email, Biografia*, ImmagineProfilo, Nome, Cognome, Password, Notify Appointment, Partita IVA)

Amministratore (IdAmministratore, Email, IsSupportoAmministratore, Nome, Cognome, Password, Partita IVA)

Immobile (IdImmobile, Descrizione, Prezzo, Dimensioni, Stanze, Piano, Ascensore, ClasseEnergetica, Modalità, Stato, Bagni, Garage, idAgente, idIndirizzo, Partita IVA)

Foto Immobile (IdFoto, Foto, IdImmobile)

Acquirente (IdAcquirente, Email, Nome, Cognome, Password, Notify Appointment)

Appuntamento (IdAppuntamento, Esito, Data, Data Richiesta, idAcquirente, idImmobile)

Notifica (IdNotifica, Tipo Notifica, Messaggio, Data, Data Richiesta, idImmobile, idAcquirente)

Indirizzo (IdIndirizzo, Stato, Città, Quartiere, Via, NumeroCivico, CAP, Latitudine, Longitudine)

Ricerca (IdRicerca, Comando, idAcquirente)

Tabella 3.1: Schema logico - fine

3.3.4 Schema Fisico

```

1 CREATE TYPE modalita AS ENUM (
2     'Vendita',
3     'Affitto'
4 );
5
6
7 CREATE TYPE notificatipo AS ENUM (
8     'Appuntamento Accettato',
9     'Appuntamento Rifiutato',
10    'Appuntamento Da Confermare',
11 );
12
13 CREATE TYPE stato AS ENUM (
14     'Nuovo',
15     'Ottimo',
16     'Buono',
17     'Da ristrutturare'
18 );
19
20 CREATE TABLE acquirente (
21     idacquirente integer NOT NULL,
22     email character varying(100) NOT NULL,
23     nome character varying(50) NOT NULL,
24     cognome character varying(50) NOT NULL,
25     password character varying(255) NOT NULL,
26     idpushnotify character varying(255),
27     notify_appointment boolean NOT NULL,
28     notify_new_estate boolean NOT NULL,
29     change_price_notify boolean NOT NULL
30 );
31
32 ALTER TABLE ONLY acquirente
33     ADD CONSTRAINT acquirente_email_key UNIQUE (email);
34
35 ALTER TABLE ONLY acquirente
36     ADD CONSTRAINT acquirente_pkey PRIMARY KEY (idacquirente);
37
38
39 CREATE TABLE agenteimmobiliare (
40     idagente integer NOT NULL,
41     email character varying(100) NOT NULL,
42     biografia text,
43     immagineprofilo text NOT NULL,
44     nome character varying(50),
45     cognome character varying(50),
46     password character varying(255) NOT NULL,
47     partitaiva character varying(20),
48     idpushnotify character varying(100),
49     notify_appointment boolean NOT NULL
50 );
51
52 ALTER TABLE ONLY agenteimmobiliare
53     ADD CONSTRAINT agenteimmobiliare_email_key UNIQUE (email);
54

```

```

55 ALTER TABLE ONLY agenteimmobiliare
56   ADD CONSTRAINT agenteimmobiliare_pkey PRIMARY KEY (idagente);
57
58 ALTER TABLE ONLY agenziaimmobiliare
59   ADD CONSTRAINT agenziaimmobiliare_nome_key UNIQUE (nome);
60
61 ALTER TABLE ONLY agenziaimmobiliare
62   ADD CONSTRAINT agenziaimmobiliare_pkey PRIMARY KEY (partitaiva);
63
64 ALTER TABLE ONLY agenteimmobiliare
65   ADD CONSTRAINT agenteimmobiliare_partitaiva_fkey FOREIGN KEY (partitaiva) REFERENCES
66   agenziaimmobiliare(partitaiva) ON DELETE CASCADE;
67
68 CREATE TABLE agenziaimmobiliare (
69   partitaiva character varying(20) NOT NULL,
70   nome character varying(100) NOT NULL,
71   sede character varying(100) NOT NULL
72 );
73
74 ALTER TABLE ONLY agenteimmobiliare
75   ADD CONSTRAINT agenziaimmobiliare_pkey PRIMARY KEY (partitaiva);
76
77 ALTER TABLE ONLY agenziaimmobiliare
78   ADD CONSTRAINT agenziaimmobiliare_nome_key UNIQUE (nome);
79
80
81 CREATE TABLE amministratore (
82   idamministratore integer NOT NULL,
83   email character varying(100) NOT NULL,
84   issupportoamministratore boolean NOT NULL,
85   nome character varying(50) NOT NULL,
86   cognome character varying(50) NOT NULL,
87   password character varying(255) NOT NULL,
88   partitaiva character varying(20)
89 );
90
91 ALTER TABLE ONLY amministratore
92   ADD CONSTRAINT amministratore_email_key UNIQUE (email);
93
94 ALTER TABLE ONLY amministratore
95   ADD CONSTRAINT amministratore_pkey PRIMARY KEY (idamministratore);
96
97 ALTER TABLE ONLY amministratore
98   ADD CONSTRAINT amministratore_partitaiva_fkey FOREIGN KEY (partitaiva) REFERENCES
99   agenziaimmobiliare(partitaiva) ON DELETE CASCADE;
100
101 CREATE TABLE appuntamento (
102   idappuntamento integer NOT NULL,
103   esito character varying(20) NOT NULL,
104   data date NOT NULL,
105   idacquirente integer NOT NULL,
106   idimmobile integer NOT NULL,
107   datarichiesta date
108 );

```

```

109
110 ALTER TABLE ONLY appuntamento
111   ADD CONSTRAINT appuntamento_pkey PRIMARY KEY (idappuntamento);
112
113 ALTER TABLE ONLY appuntamento
114   ADD CONSTRAINT appuntamento_idacquirente_fkey FOREIGN KEY (idacquirente) REFERENCES
115     acquirente(idacquirente) ON DELETE CASCADE;
116
117 ALTER TABLE ONLY appuntamento
118   ADD CONSTRAINT appuntamento_idimmobile_fkey FOREIGN KEY (idimmobile) REFERENCES
119     immobile(idimmobile) ON DELETE CASCADE;
120
121 CREATE TABLE feedback (
122   idfeedback integer NOT NULL,
123   numerostelle integer NOT NULL,
124   messaggio text NOT NULL,
125   idagente integer NOT NULL,
126   idacquirente integer NOT NULL,
127   CONSTRAINT feedback_numerostelle_check CHECK (((numerostelle >= 0) AND (numerostelle <=
128     5)))
129 );
130
131 ALTER TABLE ONLY feedback
132   ADD CONSTRAINT feedback_pkey PRIMARY KEY (idfeedback);
133
134 ALTER TABLE ONLY feedback
135   ADD CONSTRAINT feedback_idacquirente_fkey FOREIGN KEY (idacquirente) REFERENCES
136     acquirente(idacquirente);
137
138 ALTER TABLE ONLY feedback
139   ADD CONSTRAINT feedback_idagente_fkey FOREIGN KEY (idagente) REFERENCES
140     agenteimmobiliare(idagente);
141
142 CREATE TABLE fotoimmobile (
143   idfoto integer NOT NULL,
144   foto text NOT NULL,
145   idimmobile integer
146 );
147
148 ALTER TABLE ONLY fotoimmobile
149   ADD CONSTRAINT fotoimmobile_pkey PRIMARY KEY (idfoto);
150
151 ALTER TABLE ONLY fotoimmobile
152   ADD CONSTRAINT fotoimmobile_idimmobile_fkey FOREIGN KEY (idimmobile) REFERENCES
153     immobile(idimmobile) ON DELETE CASCADE;
154
155 CREATE TABLE immobile (
156   idimmobile integer NOT NULL,
157   idagente integer NOT NULL,
158   idindirizzo integer NOT NULL,
     partitaiva character varying(20) NOT NULL,

```

```

159     descrizione text NOT NULL,
160     prezzo numeric(10,2) NOT NULL,
161     dimensioni integer NOT NULL,
162     stanze integer NOT NULL,
163     piano integer NOT NULL,
164     bagni integer NOT NULL,
165     garage integer NOT NULL,
166     ascensore boolean NOT NULL,
167     classeenergetica character varying(10) NOT NULL,
168     modalita modalita NOT NULL,
169     stato stato NOT NULL,
170     CONSTRAINT isvaliddimensioni CHECK ((dimensioni > 0)),
171     CONSTRAINT isvalidpiano CHECK ((piano >= 0)),
172     CONSTRAINT isvalidprezzo CHECK ((prezzo > (0)::numeric)),
173     CONSTRAINT isvalidstanze CHECK ((stanze > 0))
174 );
175
176 ALTER TABLE ONLY immobile
177   ADD CONSTRAINT immobile_idagente_fkey FOREIGN KEY (idagente) REFERENCES
178     agenteimmobiliare(idagente);
179
180 ALTER TABLE ONLY immobile
181   ADD CONSTRAINT immobile_idagenzia_fkey FOREIGN KEY (partitaiva) REFERENCES
182     agenziaimmobiliare(partitaiva) ON DELETE CASCADE;
183
184 ALTER TABLE ONLY immobile
185   ADD CONSTRAINT immobile_idindirizzo_fkey FOREIGN KEY (idindirizzo) REFERENCES
186     indirizzo(idindirizzo);
187
188 ALTER TABLE ONLY immobile
189   ADD CONSTRAINT immobile_pkey PRIMARY KEY (idimmobile);
190
191 CREATE TABLE indirizzo (
192     idindirizzo integer NOT NULL,
193     stato character varying(50) NOT NULL,
194     citta character varying(50) NOT NULL,
195     via character varying(100) NOT NULL,
196     numerocivico character varying(255) NOT NULL,
197     cap integer NOT NULL,
198     quartiere character varying(20),
199     latitudine numeric(9,6),
200     longitudine numeric(9,6),
201     CONSTRAINT chk_latitude CHECK (((latitudine >= ('-90'::integer)::numeric) AND (latitudine
202       <= (90)::numeric))),
203     CONSTRAINT chk_longitude CHECK (((longitudine >= ('-180'::integer)::numeric) AND
204       (longitudine <= (180)::numeric))),
205     CONSTRAINT indirizzo_latitudine_check CHECK (((latitudine >= ('-90'::integer)::numeric)
206       AND (latitudine <= (90)::numeric))),
207     CONSTRAINT indirizzo_longitudine_check CHECK (((longitudine >=
208       ('-180'::integer)::numeric) AND (longitudine <= (180)::numeric)))
209 );
210
211 ALTER TABLE ONLY indirizzo

```

```
208     ADD CONSTRAINT indirizzo_pkey PRIMARY KEY (idindirizzo);  
209  
210  
211 CREATE TABLE notifica (  
212     idnotifica integer NOT NULL,  
213     tiponotifica notificatipo NOT NULL,  
214     messaggio text NOT NULL,  
215     data date,  
216     idacquirente integer NOT NULL,  
217     idimmobile integer NOT NULL,  
218     dataricezione date NOT NULL  
219 );  
220  
221 ALTER TABLE ONLY notifica  
222     ADD CONSTRAINT notifica_pkey PRIMARY KEY (idnotifica);  
223  
224 ALTER TABLE ONLY notifica  
225     ADD CONSTRAINT notifica_idacquirente_fkey FOREIGN KEY (idacquirente) REFERENCES  
226         acquirente(idacquirente) ON DELETE CASCADE;  
227  
228 ALTER TABLE ONLY notifica  
229     ADD CONSTRAINT notifica_idimmobile_fkey FOREIGN KEY (idimmobile) REFERENCES  
230         immobile(idimmobile) ON DELETE CASCADE;  
231  
232  
233 CREATE TABLE ricerca (  
234     idricerca integer NOT NULL,  
235     comando text NOT NULL,  
236     idacquirente integer NOT NULL  
237 );  
238  
239 ALTER TABLE ONLY ricerca  
240     ADD CONSTRAINT ricerca_pkey PRIMARY KEY (idricerca);  
241  
242 ALTER TABLE ONLY ricerca  
243     ADD CONSTRAINT ricerca_idacquirente_fkey FOREIGN KEY (idacquirente) REFERENCES  
244         acquirente(idacquirente) ON DELETE CASCADE;
```

3.4 Class Diagram

3.4.1 AccessService

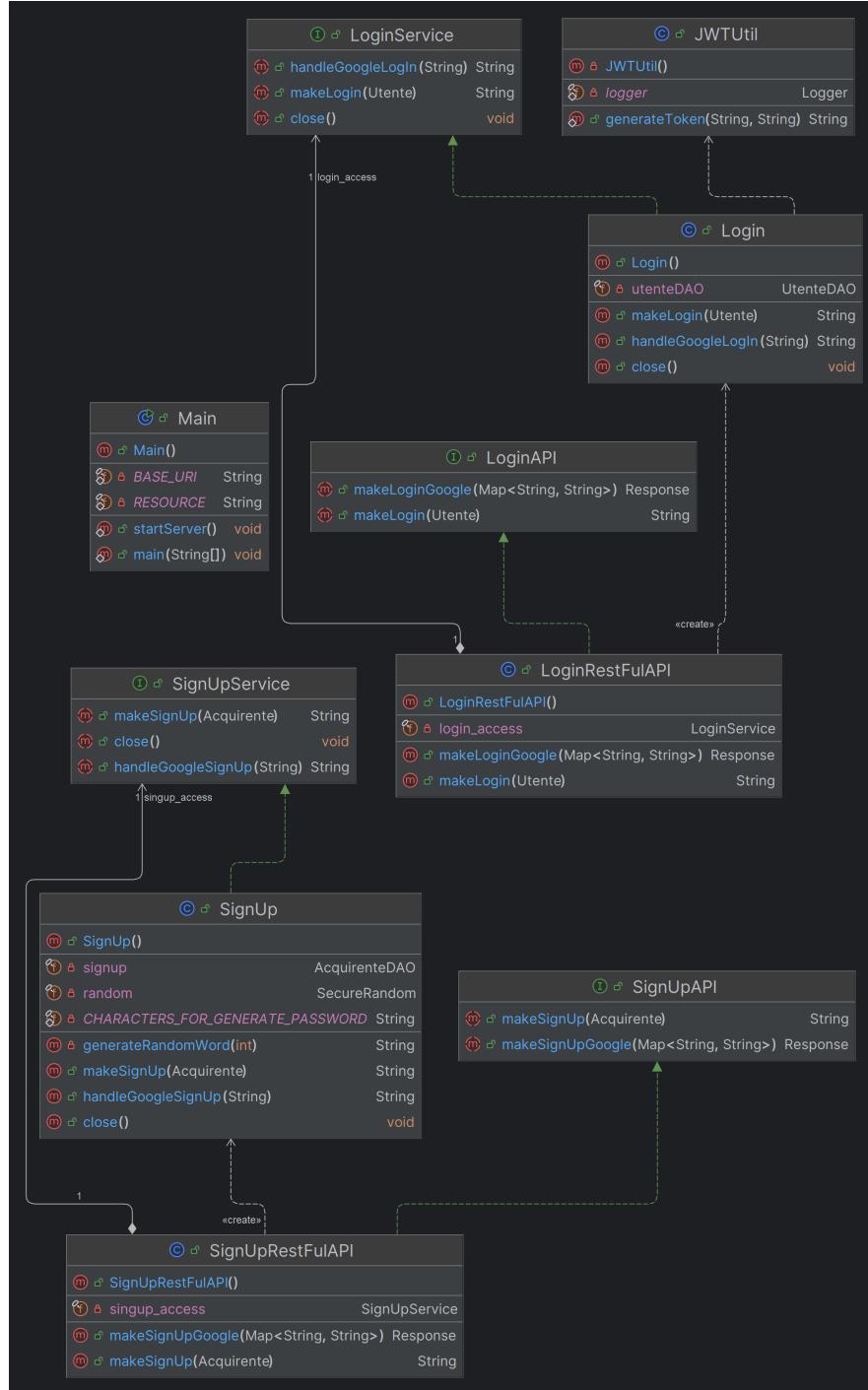


Figura 3.3: AccessService

3.4.2 AdminManagementService

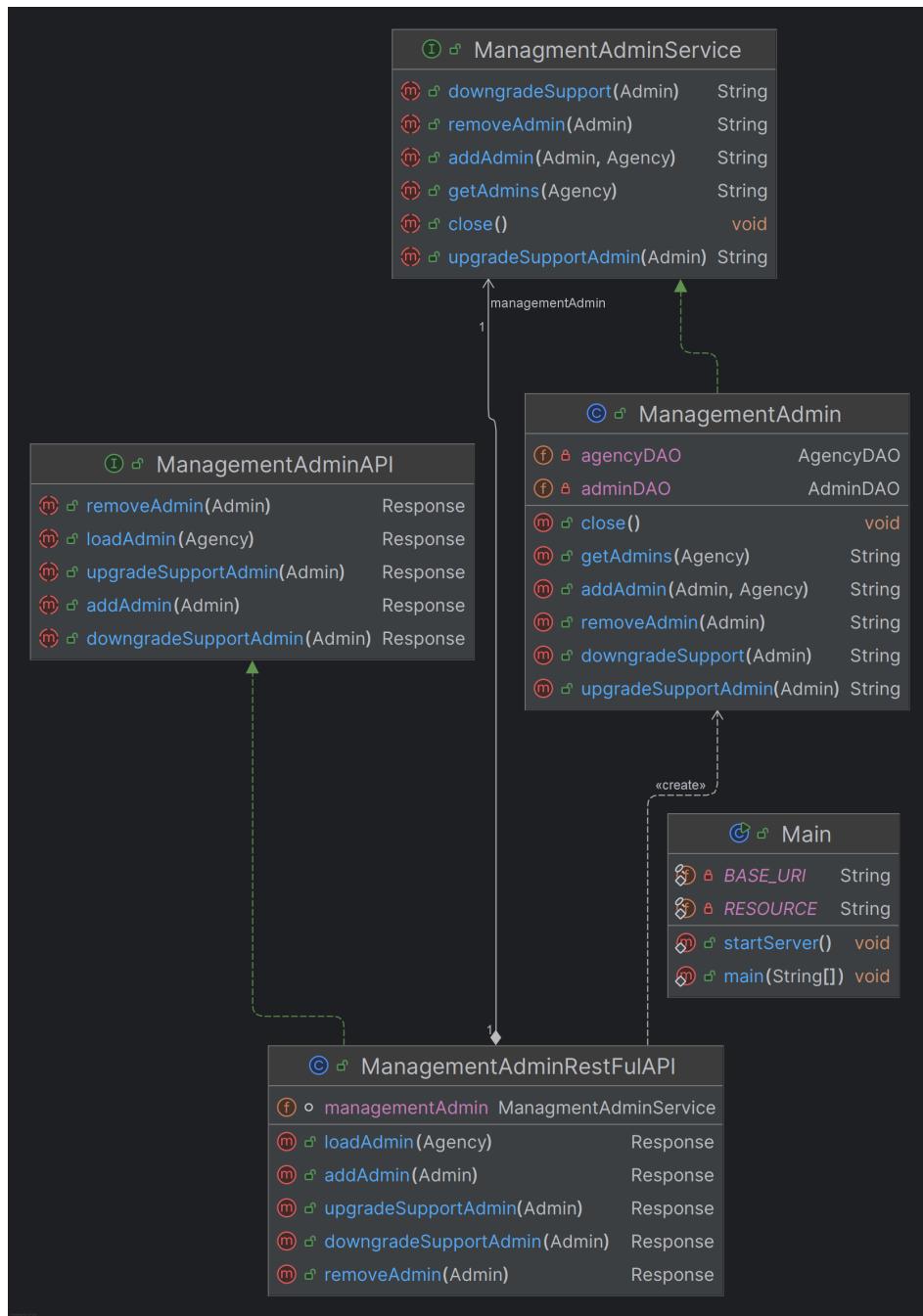


Figura 3.4: AdminManagementService

3.4.3 AdsEstateService

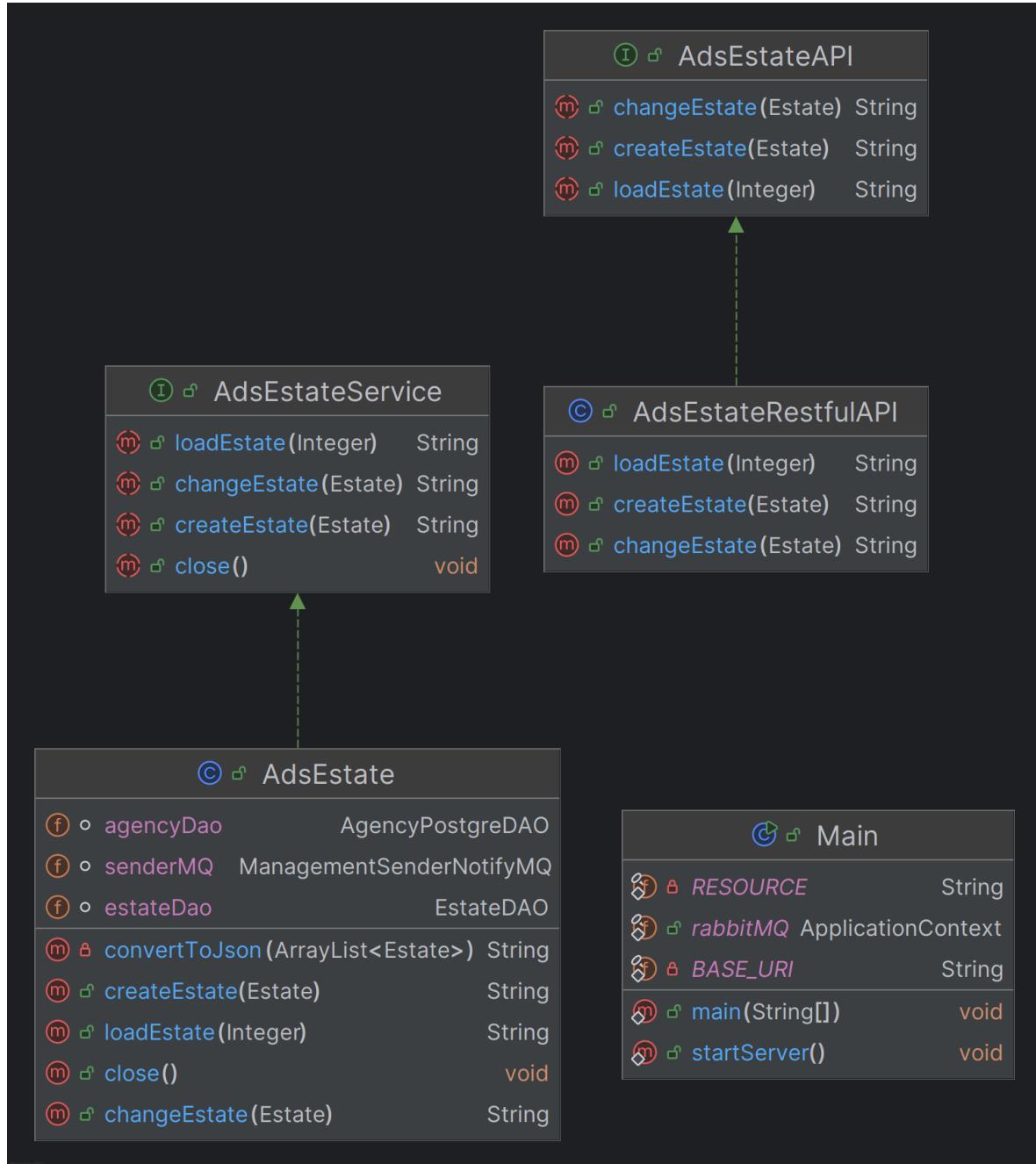


Figura 3.5: AdsEstateService

3.4.4 AgencyService

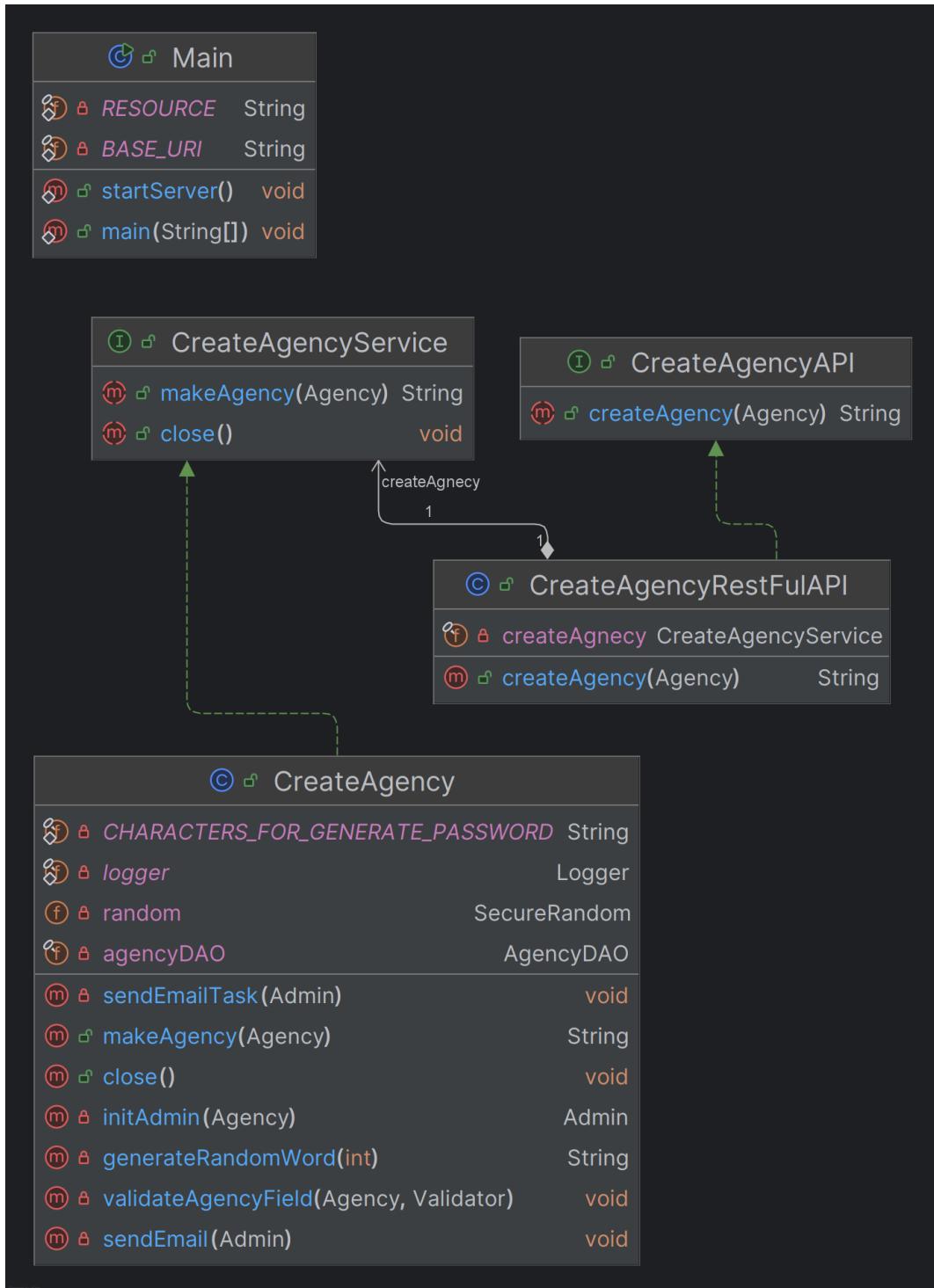


Figura 3.6: AgencyService

3.4.5 AgentManagementService

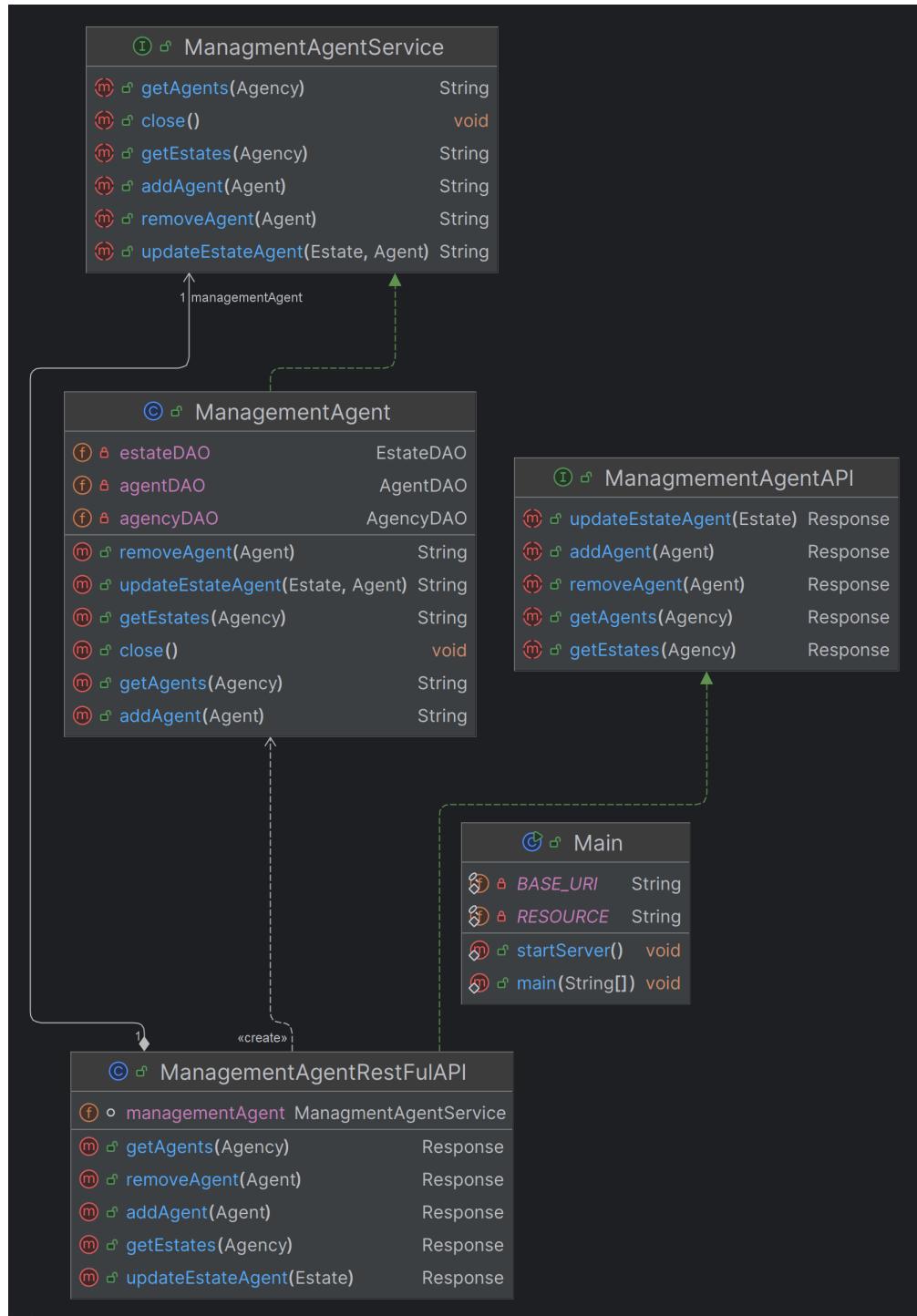


Figura 3.7: AgentManagementService

3.4.6 AppointmentService



Figura 3.8: AppointmentService

3.4.7 DAO

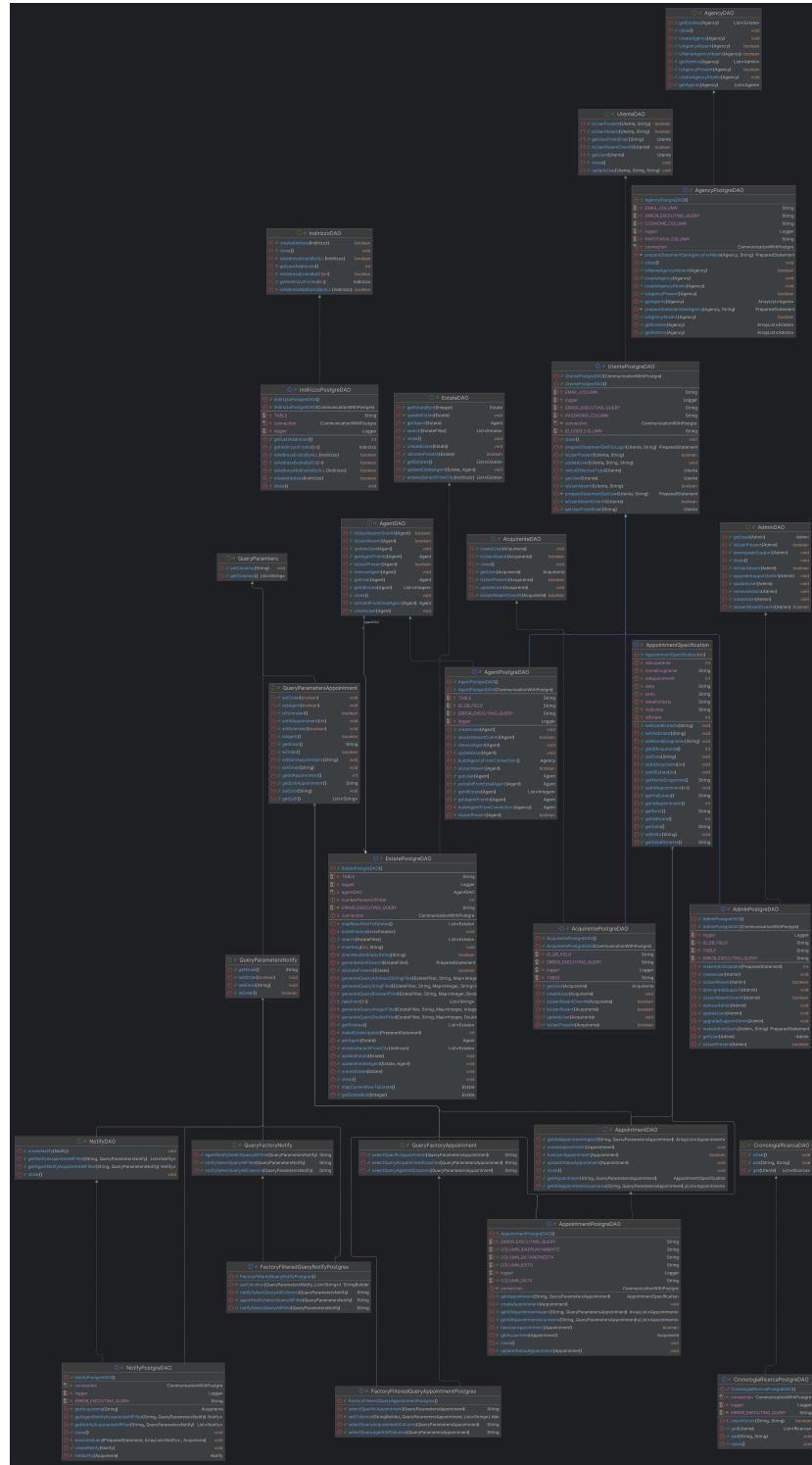


Figura 3.9: DAO

3.4.8 DatabaseLib

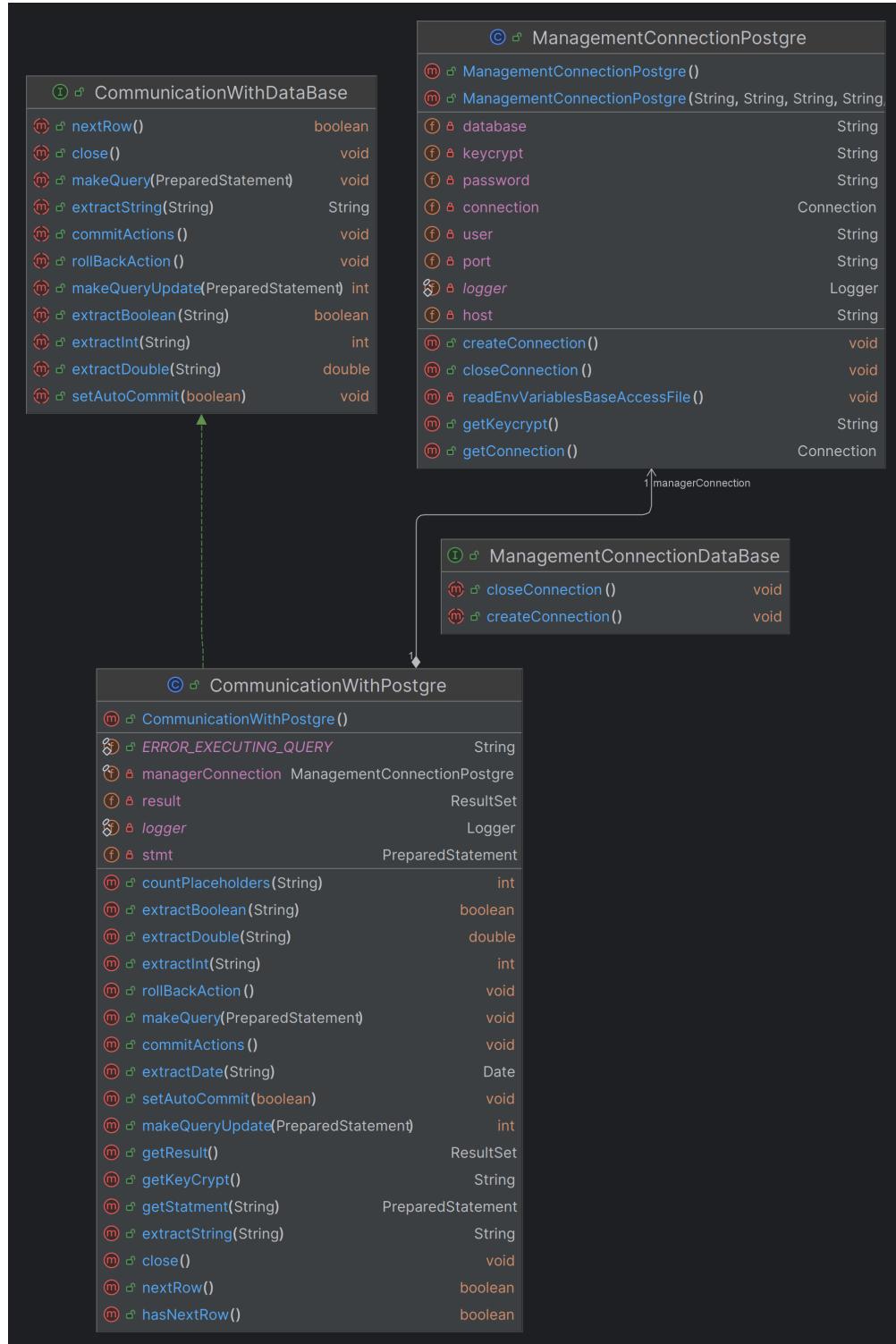


Figura 3.10: DatabaseLib

3.4.9 DiEtiEstateExceptions



Figura 3.11: DiEtiEstateExceptions

3.4.10 EmailSender

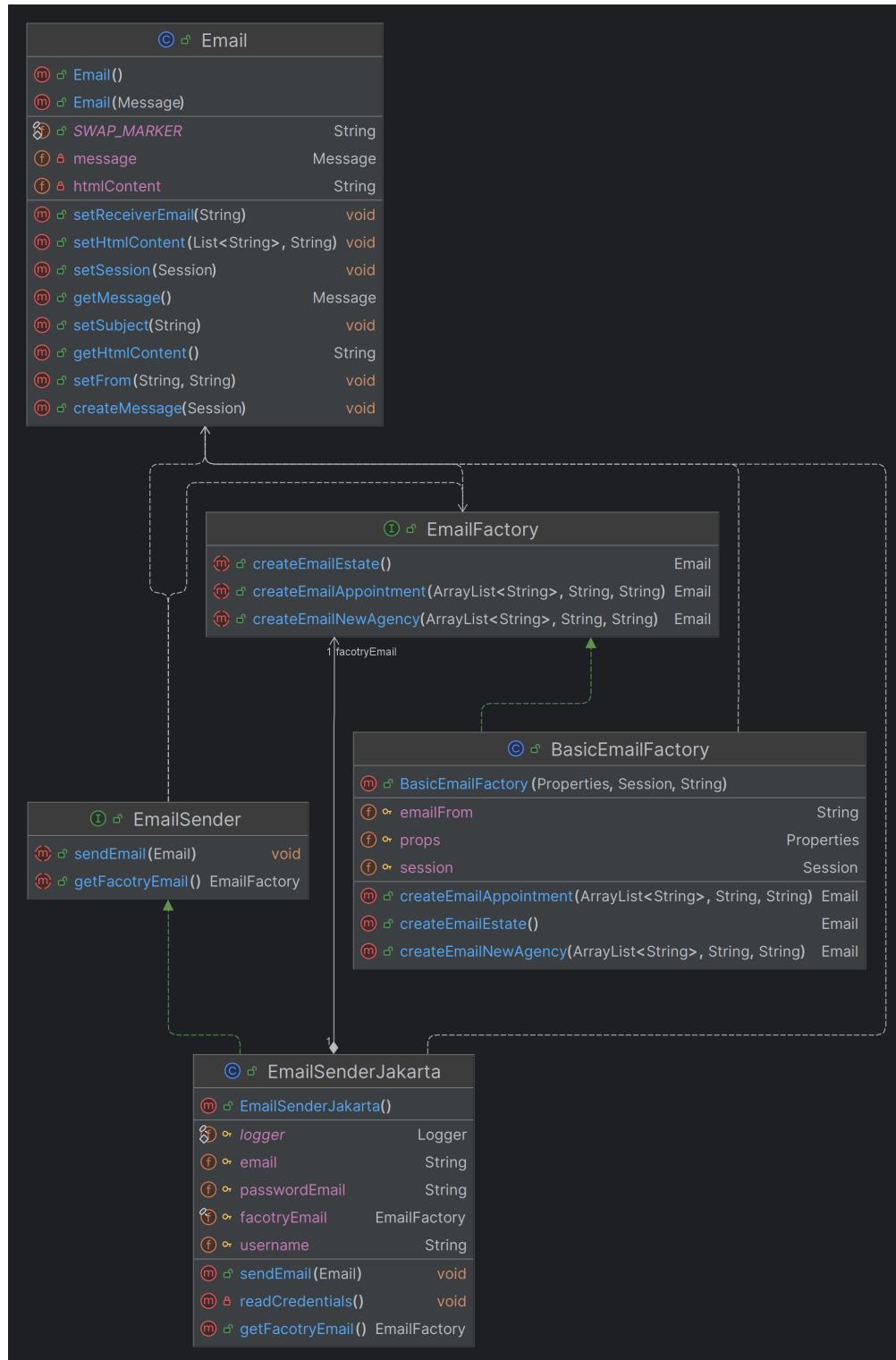


Figura 3.12: EmailSender

3.4.11 ManagementAccountService

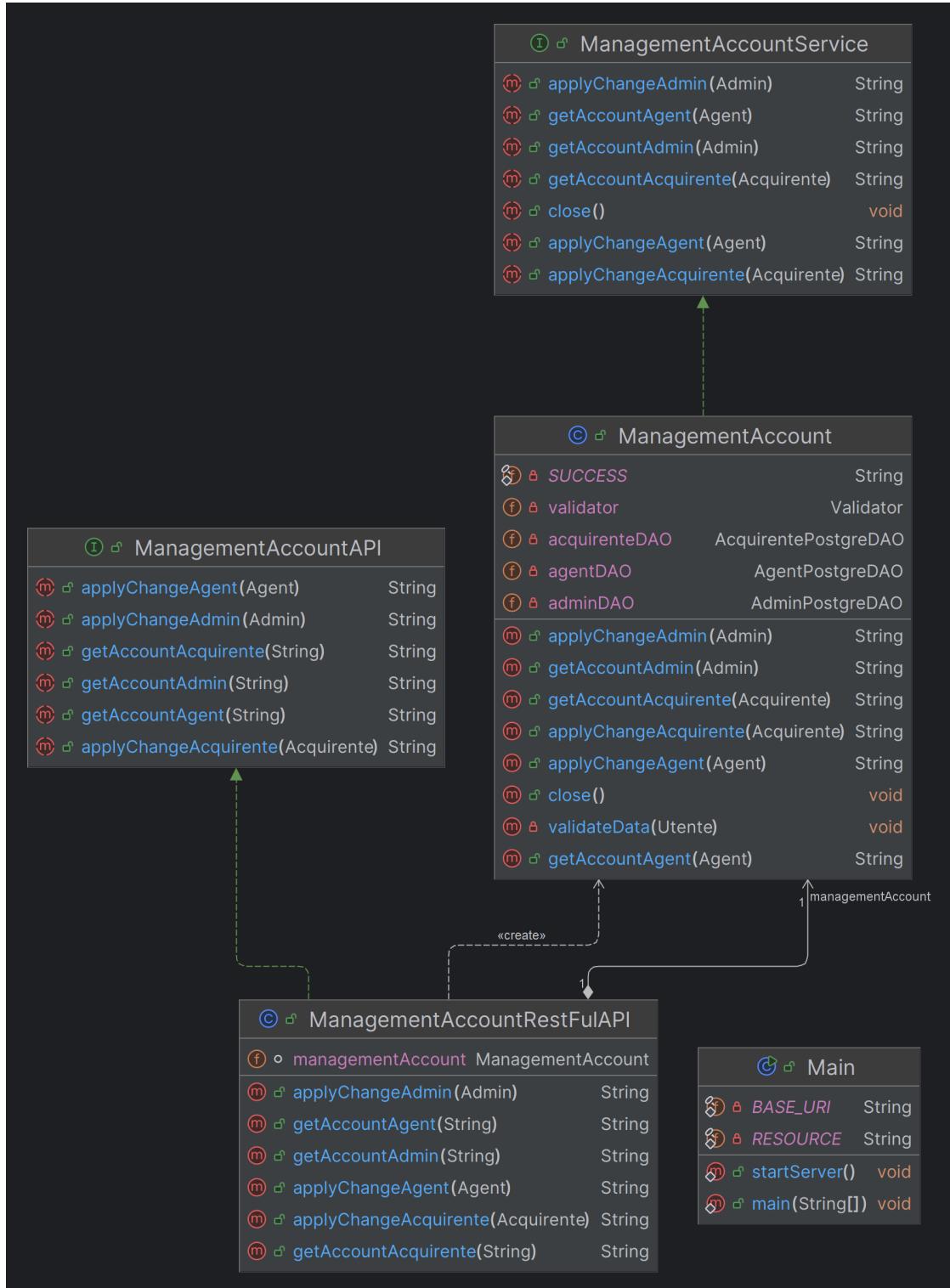


Figura 3.13: ManagementAccountService

3.4.12 Model



Figura 3.14: Model

3.4.13 NotifyService

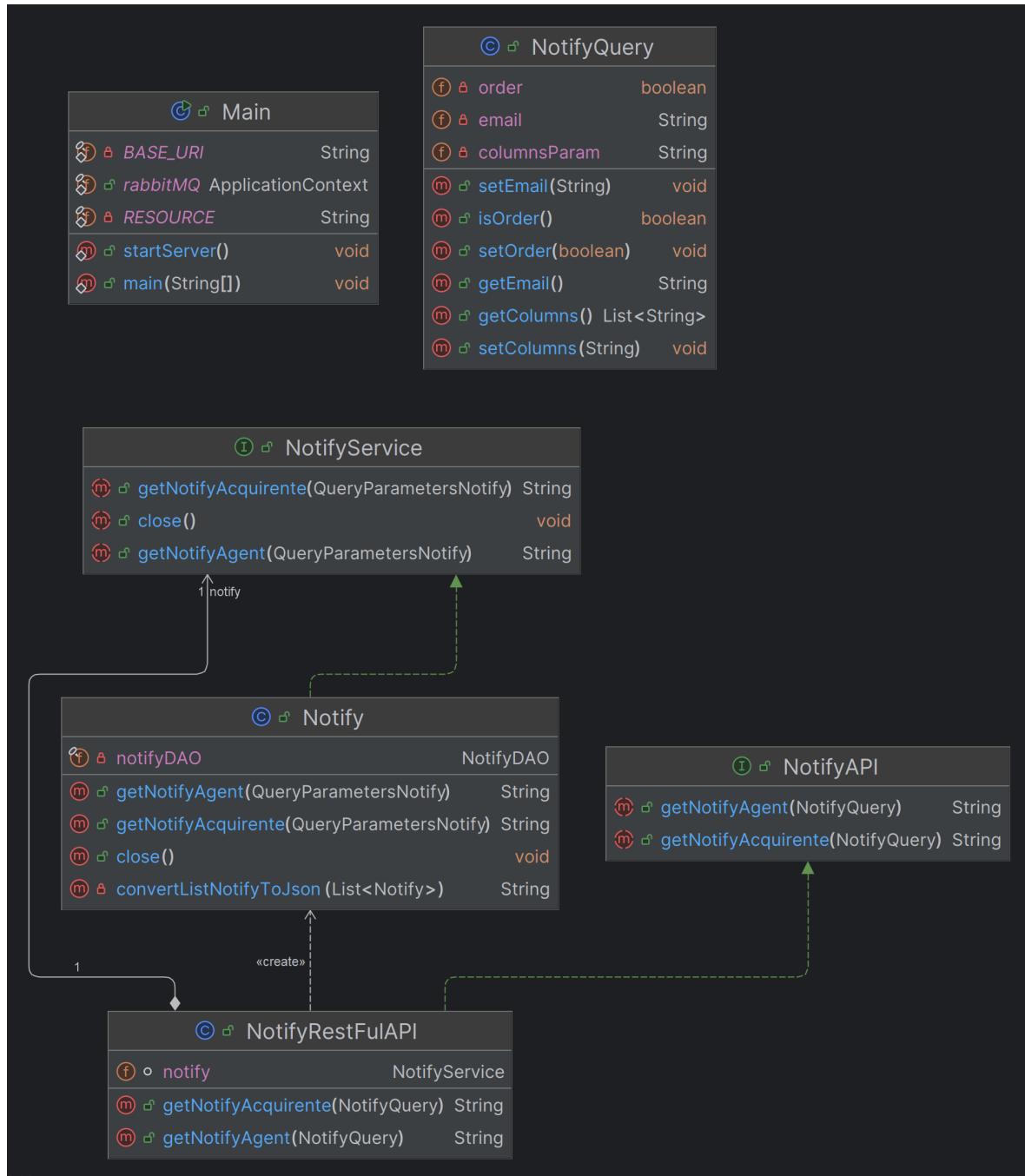


Figura 3.15: NotifyService

3.4.14 SearchService

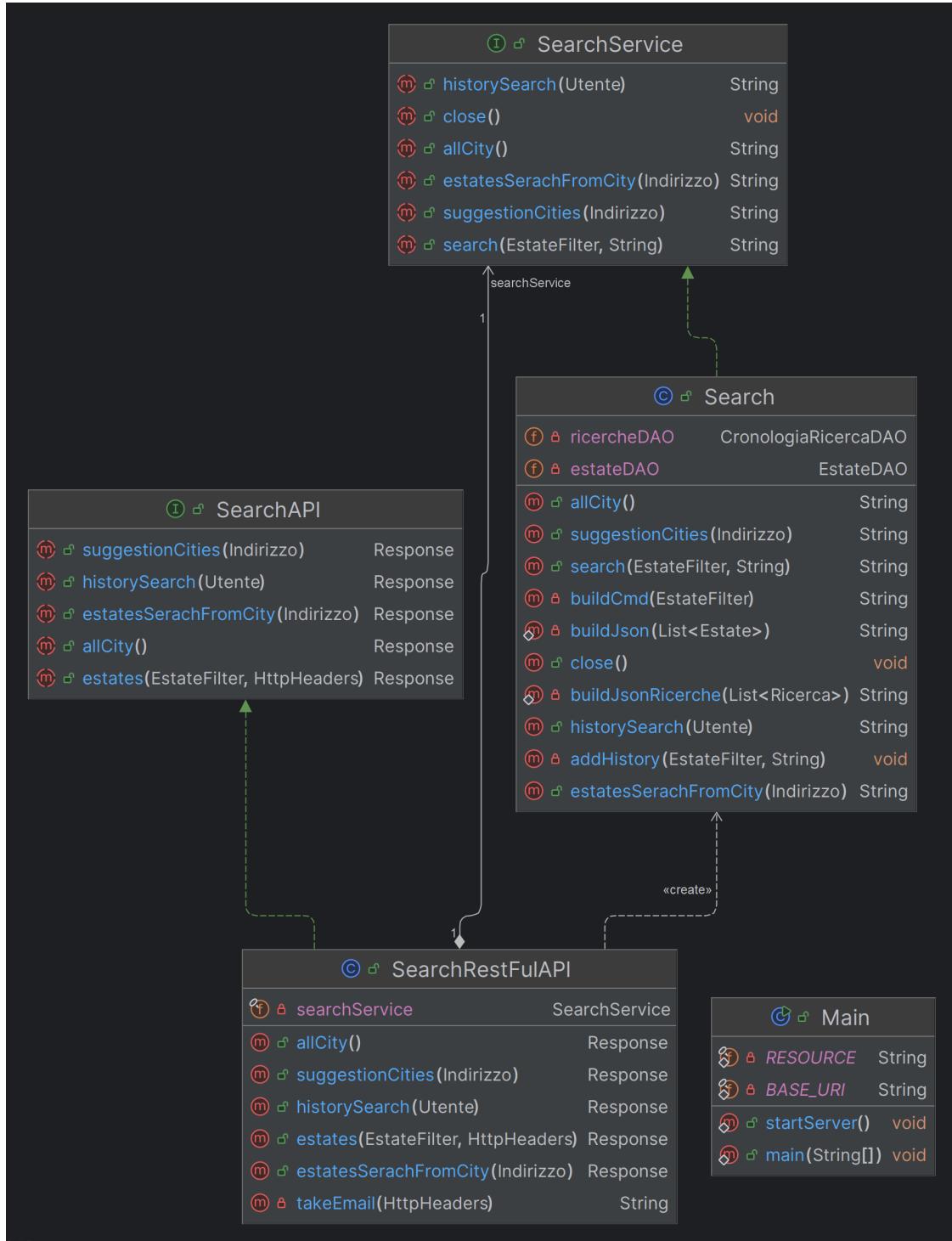


Figura 3.16: SearchService

3.4.15 Validator

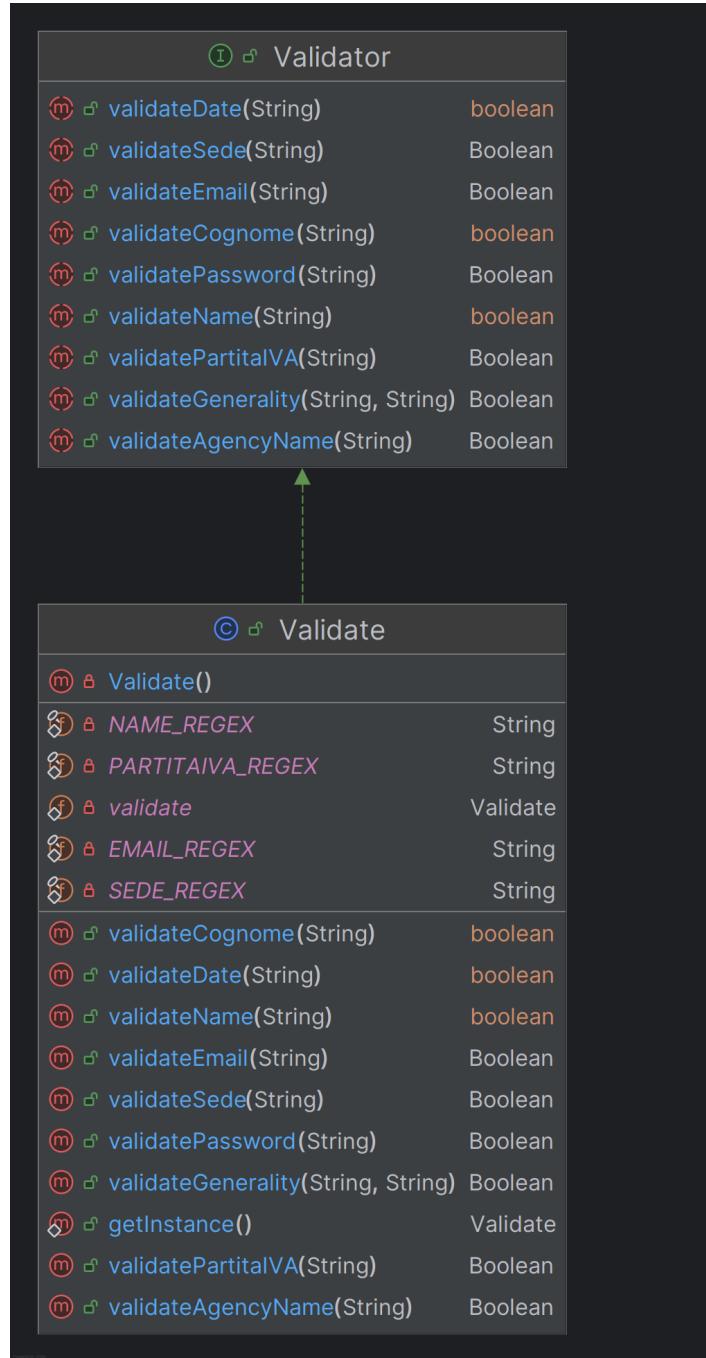


Figura 3.17: Validator

3.5 Diagrammi di sequenza

Un diagramma di sequenza (sequence diagram) è uno strumento della UML utilizzato per rappresentare le interazioni tra oggetti o attori in un sistema nel tempo. Mostra come i messaggi vengono scambiati tra le entità e l'ordine in cui avvengono. Ogni oggetto è rappresentato da una linea verticale, e le interazioni tra gli oggetti sono indicate con frecce orizzontali, che rappresentano chiamate o risposte. Il diagramma evidenzia così il flusso temporale delle operazioni.

3.5.1 Crea Agenzia

Il diagramma di sequenza "Crea Agenzia" descrive il processo attraverso il quale un sistema permette la registrazione di una nuova agenzia, mediante l'inserimento dei valori necessari.

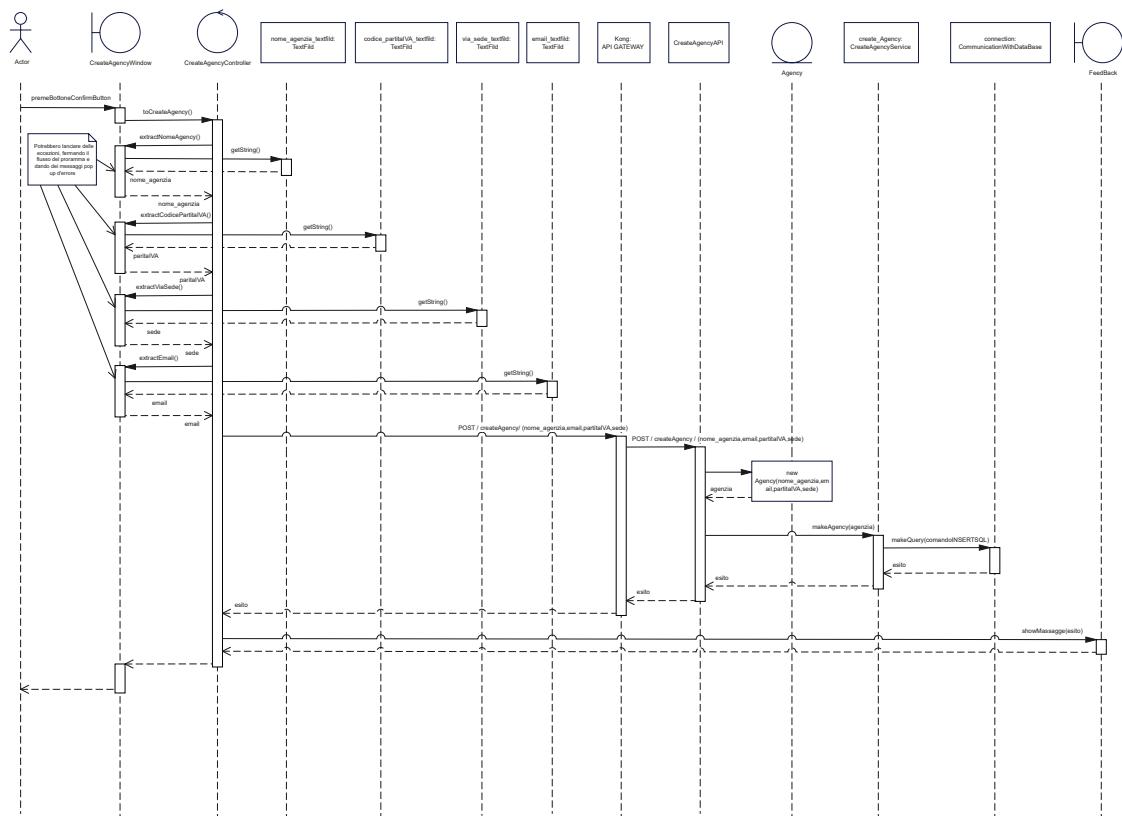


Figura 3.18: Sequence Diagram - Crea Agenzia

3.5.2 Ricerca Immobile Per Nome

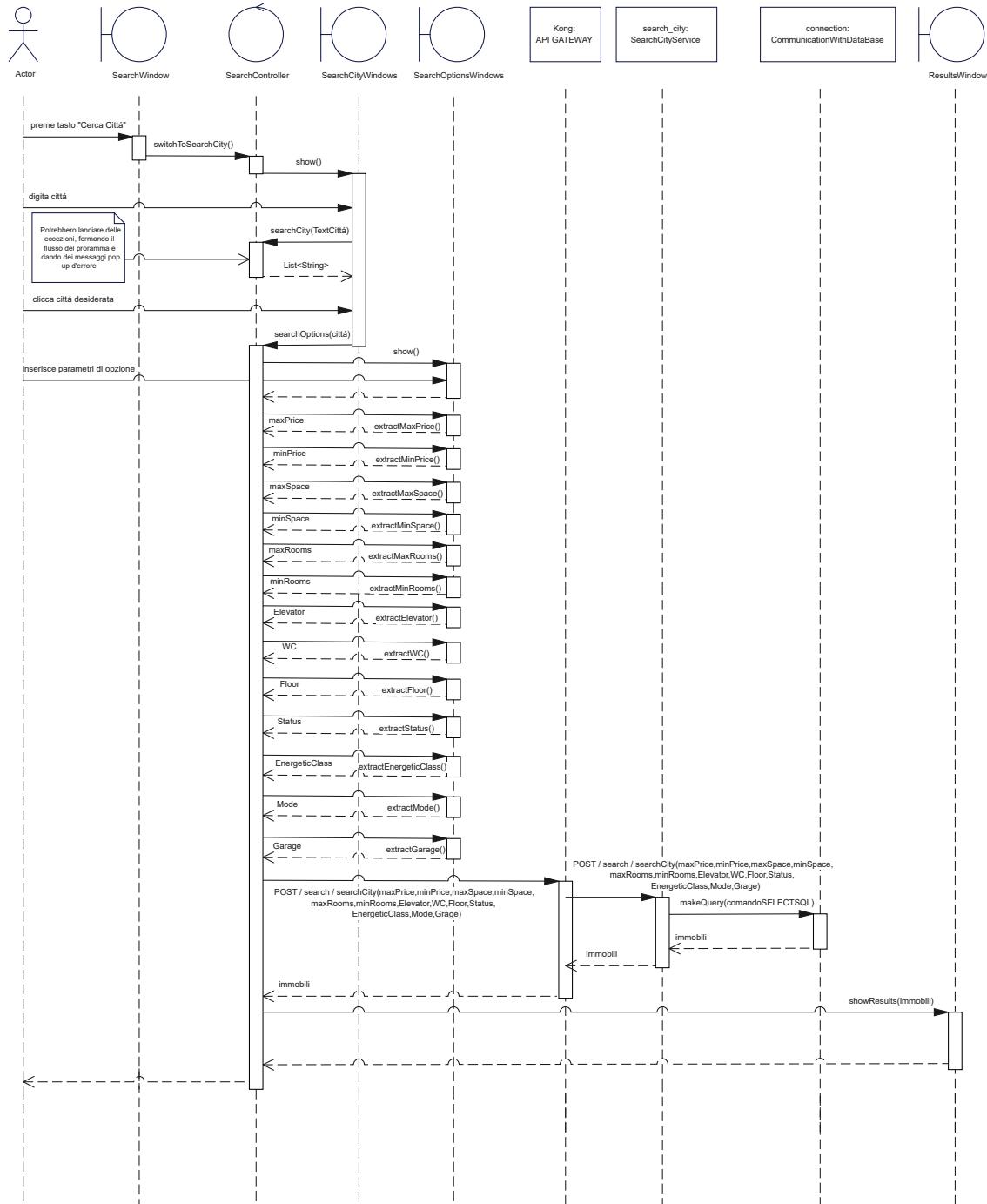


Figura 3.19: Sequence Diagram - Ricerca Immobile Per Nome

3.5.3 Gestisci Admin

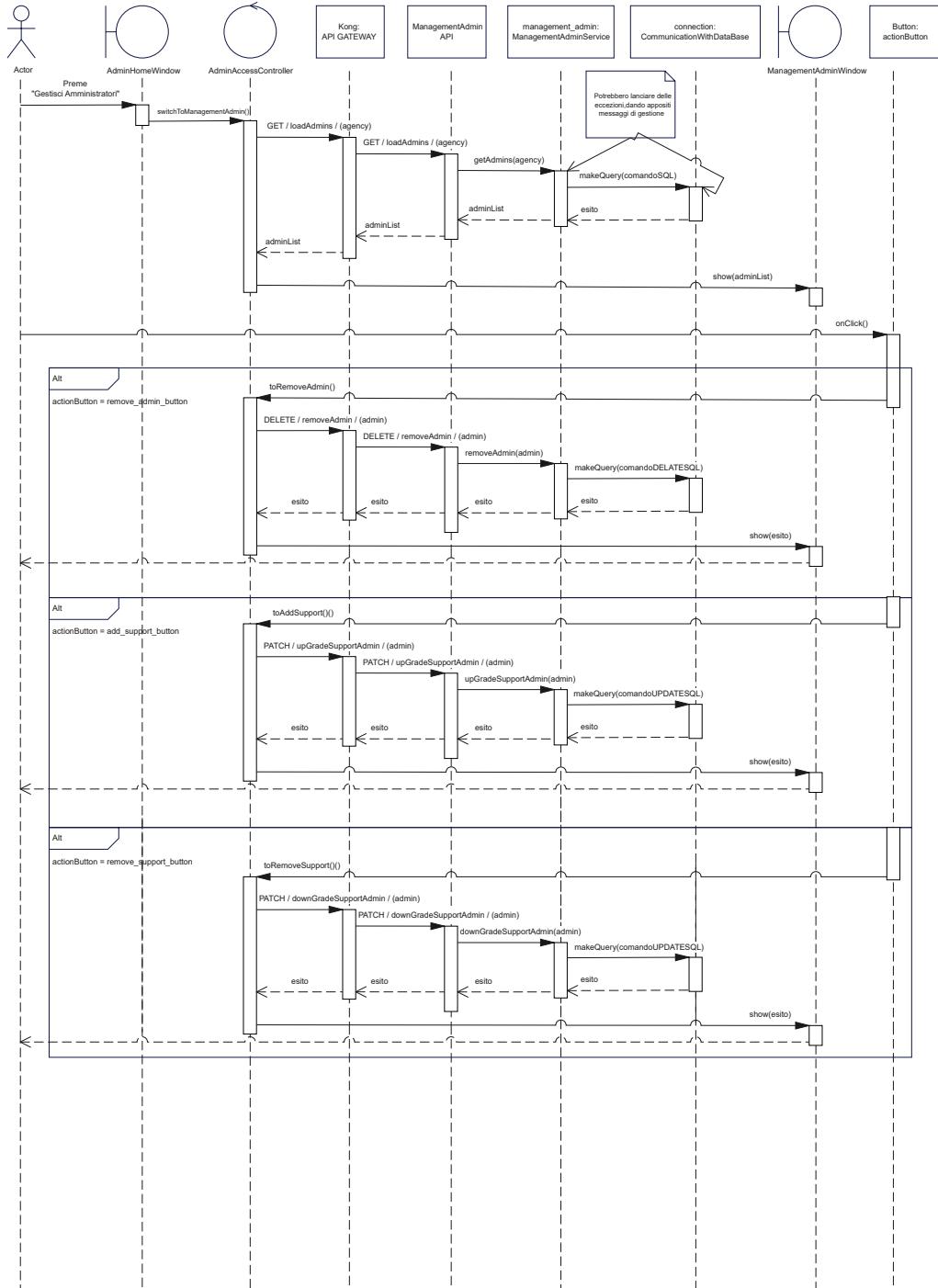


Figura 3.20: Sequence Diagram - Gestisci Admin

3.5.4 Aggiungi Admin

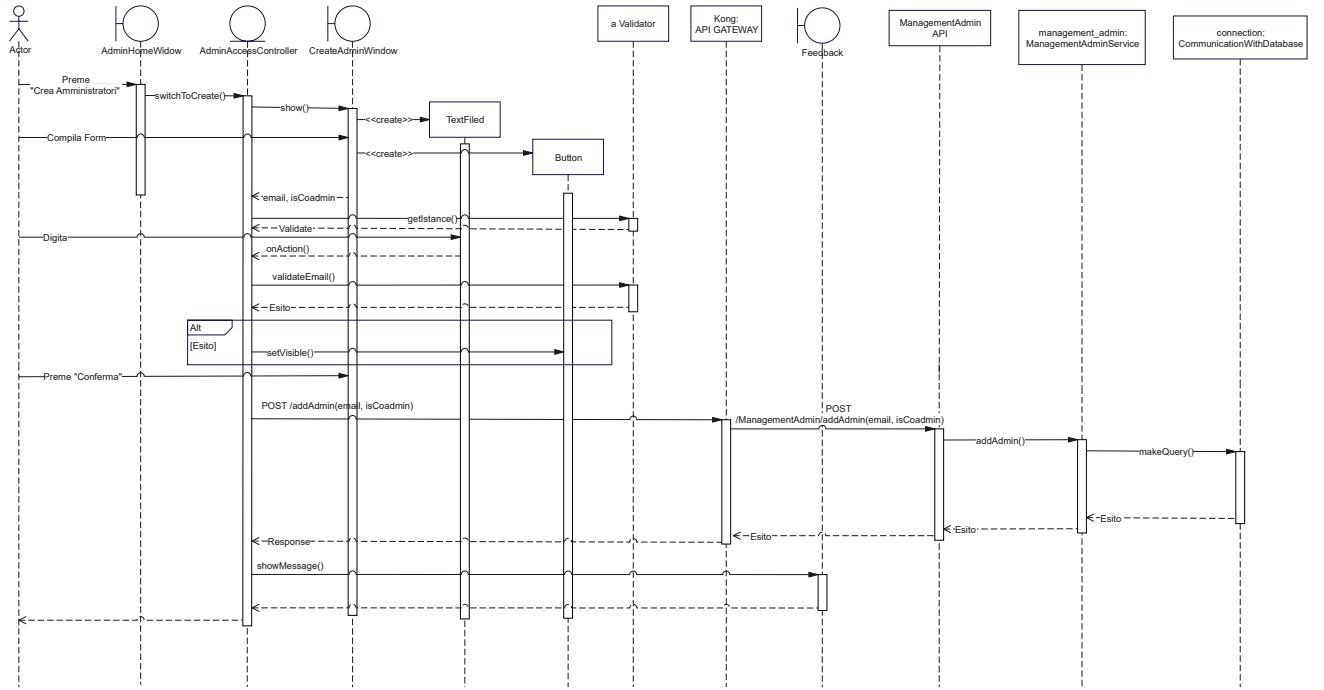


Figura 3.21: Sequence Diagram - Aggiungi Admin

Capitolo 4

UI Design

Questo documento fornisce una panoramica sul design dell'interfaccia utente, definendo principi, layout e interazioni per garantire un'esperienza efficace e intuitiva.

4.1 Mockup ad alta fedeltà

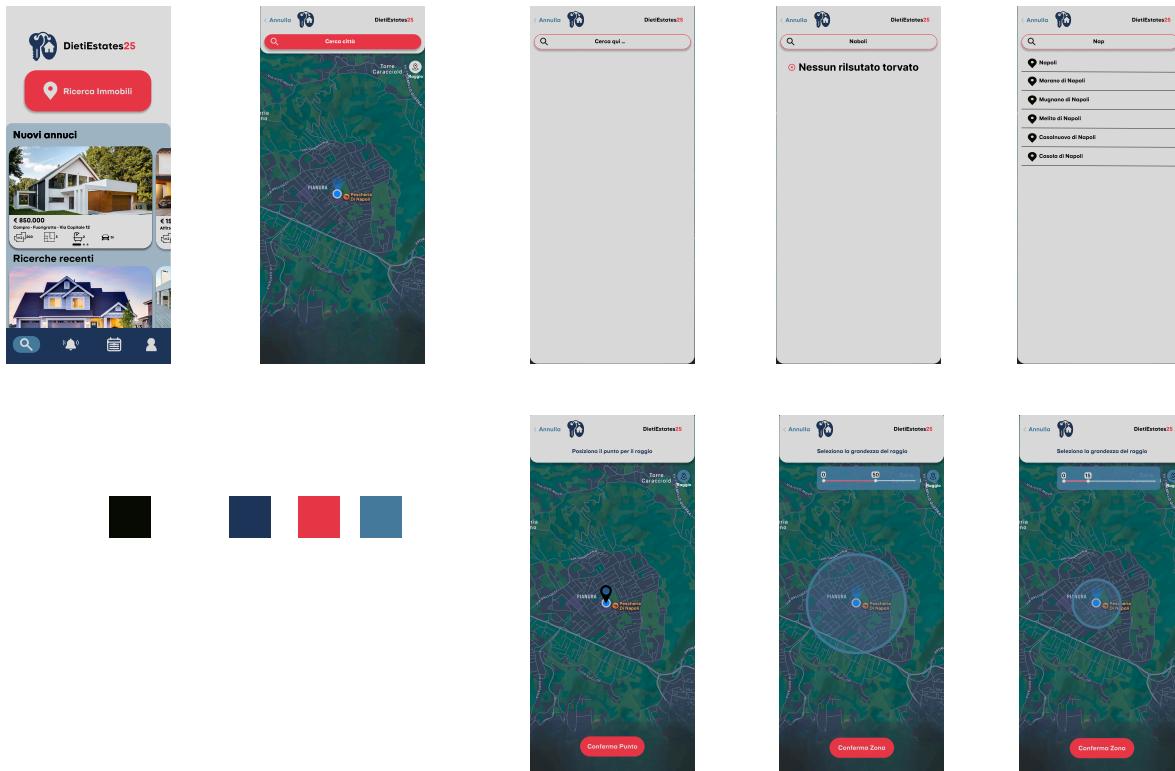


Figura 4.1: MockUp Alta fedeltà

4.2 Attributi di qualità

Questa sezione definisce le caratteristiche fondamentali che l'interfaccia utente deve rispettare per garantire un'esperienza ottimale. Gli attributi di qualità guidano le scelte di design e influenzano la percezione, l'usabilità e l'efficacia del sistema.

- **Learnability**
- **Errors**
- **Efficiency**
- **Memoriability**
- **Satisfacion**

4.2.1 Learnability

Abbiamo deciso una Learnability di tipo "*focus on novice users*", in quanto il nostro software è destinato ad un pubblico generico e non specializzato.

- **Strumenti :**
 - **Feedback** : stabiliamo di dare sempre Feedback immediati sulle azioni dell'utente
 - **Coerenza** : Coerenza nei pattern di interazione per evitare confusione. Ad esempio i feedback spuntano sempre nello stesso modo.

4.2.2 Efficiency

Dopo aver imparato a usare il sistema, gli utenti devono essere in grado di completare le attività in modo rapido e produttivo.

- **Strumenti :**
 - **Flusso di navigazione** : il flusso deve essere chiaro e lineare.
 - **Layout ottimizzato** : il layout deve permettere di ridurre il numero di azioni necessarie.
- **Strumenti (indispensabili)**
 - **Scorciatoie** : non essendo orientato a utenti esperti non vediamo la necessità di utilizzare questo strumento

4.2.3 Memoriability

Gli utenti occasionali devono essere in grado di riprendere l'uso del sistema dopo un lungo periodo di inattività senza dover reimparare da capo.

- **Strumenti :**
 - **Coerenza pattern** : Coerenza nei pattern di interazione, per facilitare il richiamo mnemonico.
 - **Minimizzazione consultare la documentazione** : essendo un software per utenti non esperti vogliamo rendere infinitesima la necessità di consultare una documentazione.

4.2.4 Gestione degli Errori

Gli errori fanno parte dell'esperienza utente, ma devono essere minimizzati e gestiti in modo efficace per evitare frustrazione o conseguenze critiche.

- **Strumenti :**

- **Messaggi di errori** : dobbiamo mostrare chiari messaggi di errori che portano a qualcosa di risolvibile.
- **Undo e rollback** : in ogni modo dobbiamo cercare di dare una strada di ritorno per recuperare eventuali azioni errate
- **Prevenzione** : su azioni piú incisive è richiesto un passaggio in più che permetta la conferma

4.2.5 Satisfaction

L'esperienza utente deve essere piacevole, soprattutto in contesti come il gaming o applicazioni creative, dove il divertimento o il comfort sono essenziali. Essendo questo software non destinato per godersi il tempo libero non necessitiamo di accorgimenti per la soddisfazione.

4.3 Principi del design

Il design UI segue principi fondamentali per garantire un'interfaccia intuitiva ed efficace. Cercando di dare attenzione al "Golfo dell'Esecuzione" e "Golfo della Valutazione" di Don Norman.

Di seguito elenchiamo alcuni principi fondamentali del design UI con esempi concreti su dove vogliamo applicarli. Tuttavia, questi rappresentano solo una minima parte delle strategie che utilizzeremo per garantire un'esperienza utente ottimale.

- **Affordances** : Intendiamo utilizzare numerosi indizi per migliorare l'esperienza dell'utente. Un esempio molto sottile potrebbe essere un insieme di pallini che indicano la presenza di più foto, simile a quelli utilizzati nei social network.
- **Constraints** : Intendiamo introdurre vincoli sugli input, ad esempio per le date utilizzeremo un calendario per evitare inserimenti errati.
- **Feedback** : Forniremo numerosi feedback per confermare azioni, segnalare successi ed evidenziare errori.
- **Consistency** : Garantiremo una consistenza interna, ad esempio i messaggi pop-up appariranno sempre nella stessa area e il tasto "Indietro" sarà posizionato in alto a sinistra. Inoltre, assicureremo una consistenza esterna, allineandoci agli standard di mercato adottati da piattaforme come Immobiliare.it, Idealista e Casa.it.
- **Metaphors** : Come mostrato nei mockup (??), la bottom bar sfrutta modelli mentali già esistenti per facilitare la comprensione.
- **Mapping** : Implementeremo una mappa con selezione di raggio, consentendo all'utente di comprendere visivamente il rapporto tra l'input fornito e l'area geografica risultante nell'output.

4.4 Leggi

4.4.1 The Power Law of Practice

Questa legge afferma che il tempo necessario per eseguire un'attività diminuisce con la pratica, seguendo una curva di apprendimento. In altre parole, più un utente utilizza un sistema, più diventa efficiente nel completare i compiti.

Applicazione nel progetto:

- L'interfaccia utente è progettata per ridurre la curva di apprendimento attraverso un design coerente e intuitivo.
- Viene data priorità a pattern di interazione standard per facilitare l'acquisizione delle competenze.

4.4.2 Hick's law

La legge di Hick descrive il tempo che una persona impiega per prendere una decisione tra un insieme di possibili scelte.

ad esempio nella schermata di home(#M1) abbiamo 7 possibili scelte , quindi applicando la legge di Hick

$$T = a + b \log_2(7 + 1)$$

- Ipotizzando $a \approx 200/250$ e $b \approx 150/200$ e considerando $\log_2(7 + 1) = \log_2(8) = 3$:
 - Utilizzando i minimi otteniamo $200ms + 150ms \cdot 2^3ms$ quindi svolgendo i calcoli $6ms$
 - utilizzando i massimi otteniamo $250ms + 200ms \cdot 2^3ms$ quindi svolgendo i calcoli $6,4ms$

4.4.3 Fitt's law

Un modello temporale che indica il tempo che ci impieghiamo a mirare un target, la formula è

$$MT = a + b \cdot ID$$

dove

$$ID = \log_2 \left(\frac{A}{W} + 1 \right)$$

Considerando che lo schermo preso in considerazione è grande 12cm(852 pixel) \times 7cm(393 pixel)
Quindi usando la formula per l'indice di difficoltà

$$ID = \log_2 \left(\frac{3,1ms}{1,5ms} + 1 \right) = \log_2(2,06 + 1) = \log_2(3,06) = 1,61bit$$

- Ipotizzando $a \approx 50/150ms$ e $b \approx 100/150\frac{ms}{bit}$ e considerando $\log_2(2,06 + 1) = \log_2(3,06) = 1,61bit$:
 - Utilizzando i minimi otteniamo $50ms + 100\frac{ms}{bit} \cdot 1,61bit$ quindi svolgendo i calcoli $211ms$
 - utilizzando i massimi otteniamo $150ms + 150\frac{ms}{bit} \cdot 1,61bit$ quindi svolgendo i calcoli $391,5ms$

4.5 Color Theory

La teoria del colore è lo studio di come i colori interagiscono tra loro e come possono essere combinati per creare armonie visive piacevoli. Si basa sul cerchio cromatico, che organizza i colori in modo logico.

4.5.1 Palettes

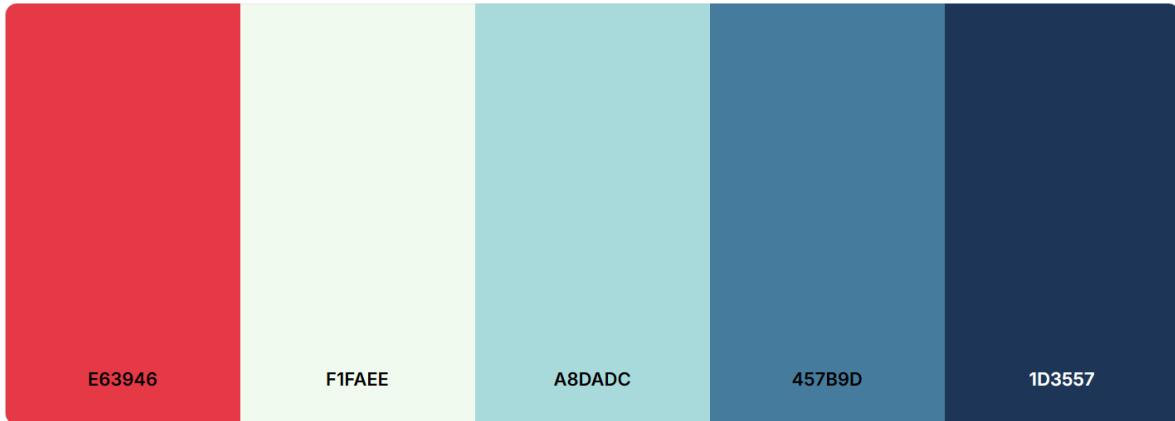


Figura 4.2: Color Palette

4.5.2 Armonies

La color armonies da noi scelta è un ibrido tra una split-complementary e una monochromatic

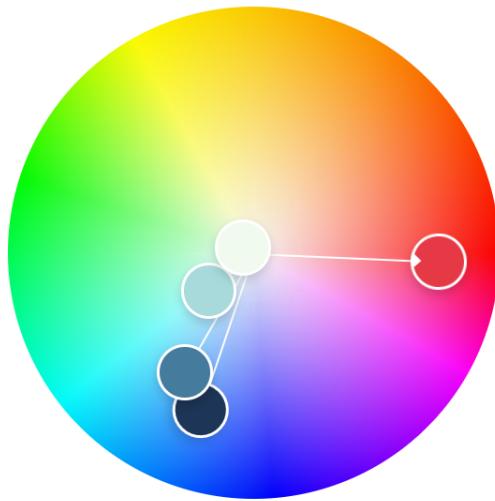


Figura 4.3: Color Armonies

4.6 Tipografia

La tipografia aiuta a migliorare leggibilità, struttura e chiarezza. Ecco i principi chiave:

- **Bold:** Utilizziamo un font con un elevato spessore per migliorare la leggibilità per qualsiasi utente.
- **Semplice:** Per garantire una maggiore leggibilità e adattarsi alla piattaforma, scegliamo un font essenziale e privo di elementi decorativi.

Font Il font scelto è il Goldbill XL Bold, il quale rispecchia le nostre esigenze.

Ingegneria del Software

Figura 4.4: Esempio Tipografia

4.7 Gestalt theory of perception

La percezione, secondo la Gestalt, è orientata alla visione del "tutto" piuttosto che dei singoli elementi. Seguendo i principi di Gestalt è possibile migliorare l'usabilità e rendere più comprensibile l'interfaccia utente.

4.7.1 Principio di similarità

Gli elementi che condividono una caratteristica visiva sono percepiti come più correlati rispetto a quelli che risultano dissimili.

Abbiamo applicato questo principio, ad esempio, nella schermata #M6 (Opzioni di ricerca, ??). I pulsanti utilizzati per selezionare lo stato dell'immobile (Nuovo, Ottimo, Buono, Da ristrutturare) o il numero di garage (Garage singolo, Garage doppio, Posto auto) presentano la stessa forma rettangolare, colore e disposizione. Questa somiglianza aiuta l'utente a riconoscere immediatamente che tali pulsanti appartengono alla stessa categoria e condividono una funzionalità simile.

4.7.2 Principio di prossimità

Gli elementi posizionati vicini tra loro tendono a essere percepiti come appartenenti alla stessa entità o gruppo, anche in assenza di somiglianze visive.

Ad esempio, nella schermata #M7 (Risultati Ricerca) e nella Home, tutte le specifiche di un immobile sono raggruppate vicino all'immagine corrispondente. Questa disposizione, combinata con il principio della regione comune, rafforza l'immediatezza nella comprensione delle informazioni.

4.7.3 Principio di connessione

Gli elementi collegati visivamente da linee, frecce o altri indicatori grafici vengono percepiti come correlati, anche in assenza di prossimità o somiglianza. Questo principio aiuta a stabilire relazioni chiare tra gli elementi, migliorando la comprensione dell'interfaccia.

Abbiamo applicato questo principio nella sezione di ricerca #M5 (Ricerca città), in cui la cronologia e la barra di ricerca condividono lo stesso bordo.

4.7.4 Principio della regione comune

Gli elementi contenuti all'interno di un'area definita vengono percepiti come appartenenti allo stesso gruppo, indipendentemente dalla loro prossimità o somiglianza. Questo principio contribuisce a rendere più intuitiva la relazione tra gli elementi e le loro funzioni.

Linguaggio Ad esempio, tutti i pulsanti all'interno della bottom bar chiariscono visivamente la loro funzione, facilitando la navigazione tra le diverse pagine.

4.7.5 Gerarchia

L'obiettivo di questa analisi è identificare e descrivere la gerarchia visiva utilizzata nell'interfaccia dell'applicazione **UninaEstate25**, applicando i principi di **tipografia**, **colori**, **scala** e **Gestalt**, nel contesto dei pattern naturali di scansione dell'utente (*pattern-Z* e *pattern-F*). Sarà analizzata in particolare la **Home page**



Figura 4.5: Home page

1. Elemento Dominante: Pulsante “Ricerca Immobile”

- Evidenziato con un **colore rosso acceso**, dimensioni rilevanti e icona di lente d'ingrandimento.
- Posizionato nella parte alta dell'interfaccia, subito dopo il logo.
- Segue il *pattern-Z*, attirando immediatamente l'attenzione dell'utente.
- Funzione: **Call to Action primaria**.

2. Secondo Livello: Sezione “Nuovi Annunci”

- Titolo “**Nuovi Annunci**” in grassetto e ben contrastato.
- Posizionato appena sotto il pulsante, segue la direzione naturale di lettura (Z-pattern).
- Inserito in una regione con **sfondo blu più scuro**, rispetto al resto dell’interfaccia.
- Funziona come introduzione ai contenuti successivi (schede immobili).

3. Terzo Livello: Box Annuncio Singolo

Ogni annuncio è strutturato in componenti con gerarchia interna coerente:

- **Immagine immobile**: elemento più grande del box, con forte richiamo visivo.
- **Prezzo**: posizionato subito sotto l’immagine, con buon contrasto.
- **Dettagli sintetici** (affitto, superficie, camere, bagni, garage):
 - Organizzati con **icone + testo**, seguono la *prossimità*.
 - La *regione comune* (box) ne rafforza l’appartenenza visiva.

4. Navigazione Inferiore (Bottom Navigation Bar)

- Sempre visibile, con bassa scala visiva.
- Le icone sono accompagnate da etichette testuali.
- L’elemento attivo (“**Home**”) è evidenziato tramite colore di sfondo e contrasto (*similarità + stato attivo*).

Principi Gestalt Applicati

- **Prossimità**: gli elementi correlati (es. dettagli annuncio) sono visivamente raggruppati.
- **Regione comune**: ogni annuncio è incorniciato da un box con sfondo omogeneo.
- **Similarità**: tutti gli annunci condividono struttura e stile coerente.

Ordine di Attenzione Stimolato (Flow Visivo)

1. Pulsante “Ricerca Immobile” (*Call to Action visiva primaria*)
2. Titolo “Nuovi Annunci”
3. Immagini degli immobili
4. Prezzo
5. Dettagli tecnici (icone, superficie, stanze, ecc.)
6. Navigazione inferiore (con evidenziazione sul pulsante attivo)

4.8 Linee guida e principi nella HCI

4.8.1 Euristiche di Usabilità Nielsen-Molich

Dialogo semplice e naturale

Le interfacce utente devono essere semplici, ma non eccessivamente. Ogni schermata ha un ruolo e un utilizzo ben definiti; per questo abbiamo sfruttato in maniera ottimale il mapping e le metafore. Ad esempio, il tasto "indietro" è sempre posizionato a sinistra, in linea con le abitudini di lettura europee, mentre la bottom bar favorisce una navigazione intuitiva.

Parla il linguaggio dell'utente

Abbiamo adottato un linguaggio familiare per l'utente, mettendolo sempre al centro dell'esperienza. Ad esempio, abbiamo preferito formulazioni come "Hai prenotato l'appuntamento" invece di "Il sistema ha registrato la prenotazione", rendendo la comunicazione più diretta e coinvolgente.

Minimizza il carico cognitivo dell'utente

Le macchine gestiscono la Working Memory in modo significativamente più efficiente e con una capacità superiore rispetto alla limitata quantità di chunk della memoria umana. Per questo motivo, abbiamo ottimizzato l'interfaccia riducendo al minimo il carico cognitivo dell'utente, adottando menu a tendina precompilati, suggerimenti contestuali e metafore visive.

Coerenza

- **Consistenza Interna:** Abbiamo garantito una coerenza logica tra le diverse schermate, mantenendo elementi comuni come la bottom bar e la upbar per offrire un'esperienza utente uniforme e intuitiva.
- **Consistenza Esterna:** Ci siamo allineati agli standard di settore, ispirandoci ai principali competitor, come Immobiliare.it, Idealista e Casa.it, per garantire familiarità e facilità d'uso agli utenti.

Feedback

Forniamo all'utente un costante riscontro sulle azioni del sistema, non solo in caso di errore, ma anche attraverso feedback positivi per confermare il successo delle operazioni. In questo modo, garantiamo maggiore chiarezza e sicurezza nell'interazione con l'interfaccia.

Bias del lavoro percepito: Abbiamo sfruttato questo bias ad esempio mostrando un'animazione durante il processo di ricerca, accompagnata da un messaggio che informa l'utente sull'analisi e la selezione dei migliori annunci.

Uscite chiaramente segnalate e inversione delle azioni

Indicare chiaramente come annullare un'azione in corso o revocare un'azione già compiuta aiuta gli utenti a commettere meno errori, indipendentemente dalla loro gravità.

abbiamo aggiunto diverse finestre di conferma per le azioni, come nella rimozione degli admin, dove è necessario confermare prima di eliminare uno o più amministratori.

Shortcuts

I shortcut sono utili per gli utenti esperti, ma nel nostro caso, la nostra applicazione sarà utilizzata prevalentemente da nuovi utenti, quindi non è necessaria l'integrazione di scorciatoie da tastiera.

Messaggi errori

I messaggi di errore sono fondamentali per l'utente finale, poiché aiutano a comprendere cosa sta accadendo, qual è l'errore e, se possibile, come risolverlo. È essenziale essere cordiali, specificare chiaramente l'errore, fornire indicazioni su come risolverlo e, se necessario, mostrare informazioni aggiuntive per gli addetti ai lavori.

ad esempio, uno di questi messaggi di errore compare durante la ricerca: se la città ricercata non esiste, verrà visualizzato un avviso a schermo. verrà visualizzato un messaggio molto chiaro al utente(il messaggio è: "città selezionata non trovata").

Previene gli errori

Gli errori è meglio prevenirli che correggerli, quindi è importante impostare dei constraint per evitarli. Ad esempio, quando inseriamo una data, è previsto un controllo che impedisce di selezionare una data finale precedente a quella di inizio.

Aiuto e documentazione

Una documentazione è utile per comprendere strumenti complessi, specialmente quando devono essere utilizzati da persone esperte. Tuttavia, nel nostro caso, l'applicazione sarà utilizzata da persone non esperte ed è stata progettata per essere intuitiva e semplice da usare.

Capitolo 5

Software Design

5.1 Istruzioni utilizzo

5.1.1 Descrizione

Questa cartella contiene tutti i **microservizi** del progetto DietiEstates25.

5.1.2 Build delle immagini Docker

Per effettuare il build di tutte le immagini Docker dei microservizi, utilizza i seguenti script:

- **Windows:** `ImageBuilder.bat`
- **Linux/Mac:** `ImageBuilder.sh`

Questi script eseguono automaticamente il build di tutte le immagini necessarie tramite Docker.

5.1.3 Variabili d'ambiente

Per configurare le variabili d'ambiente necessarie (es. credenziali del database, chiavi API, client ID di Google, ecc.), utilizza:

- **Windows:** `SetEnvForProduction.bat`
- **Linux/Mac:** `SetEnvForProduction.sh`

Questi script configurano tutte le variabili per l'esecuzione in ambiente di produzione.

5.1.4 Avvio dei servizi

Dopo aver buildato le immagini e configurato le variabili d'ambiente, puoi avviare tutti i servizi con il comando:

¹ `docker compose up -d`

Questo comando utilizza il file `docker-compose.yaml` per avviare tutti i microservizi in modalità *detached*.

5.1.5 Struttura principale

La cartella MicroService include i seguenti microservizi:

- `AccessService/`
- `AdminManagementService/`
- `AdsEstateService/`
- `AgencyService/`
- `AgentManagementService/`
- `API Gateway/`
- `AppointmentService/`
- `Dependacy/`
- `ManagementAccountService/`
- `NotifyService/`
- `SearchService/`

Note finali

- Assicurati di avere Docker correttamente installato e configurato.
- Esegui prima lo script per impostare le variabili d'ambiente, poi procedi con il build delle immagini.
- Consulta la documentazione specifica di ciascun microservizio per ulteriori dettagli.

5.1.6 Qualità del codice

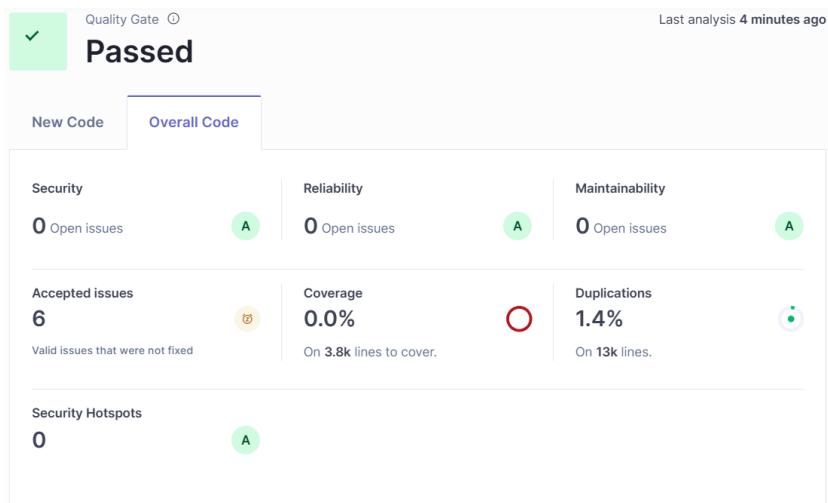


Figura 5.1: Qualità del software misurata da SonarQube

5.1.7 Versioning

Il sistema di Versioning da noi adottato è stato una repository comune ([github](#)) mediante l'utilizzo dello standard [git](#).

Statistiche

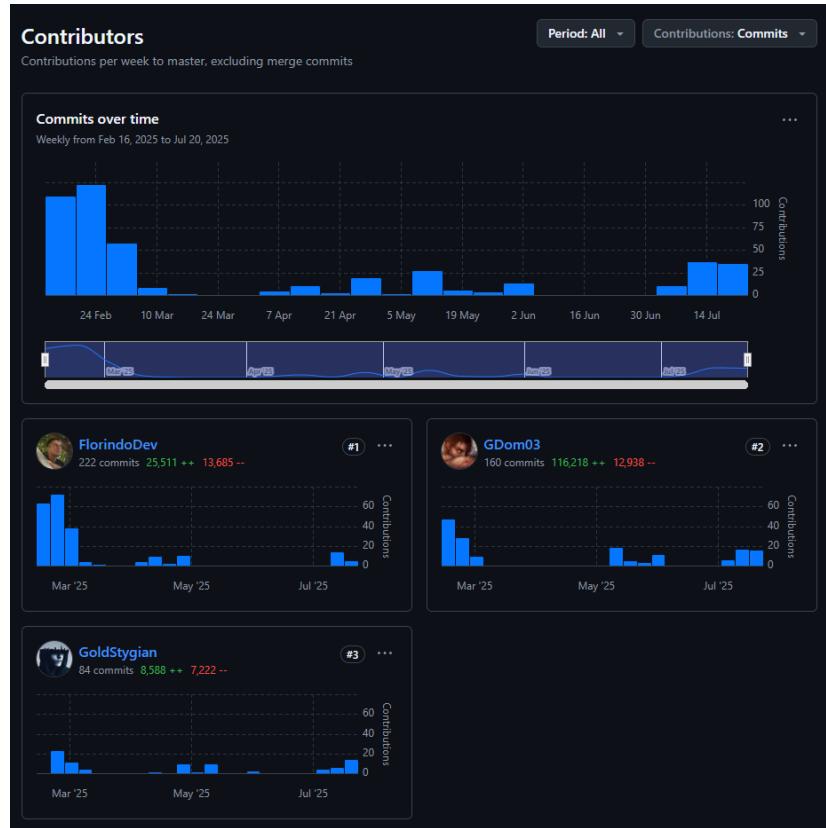


Figura 5.2: Versioning - Statistiche

Capitolo 6

Testing

6.1 Progettazione dei test

6.1.1 AddAdmin test

Il metodo `addAdmin(Admin admin)` consente l'aggiunta di un nuovo utente amministratore al sistema. Durante l'aggiunta vengono effettuate varie validazioni sui dati inseriti, tra cui email, password, nome, cognome e partita IVA. In caso di errore, il metodo restituisce un messaggio di errore corrispondente.

Criteri di Validazione

Email

- **CE1 - Valida:** Formato corretto (`xxx@yyy.zzz`)
- **CE2 - Valida ma già presente nel database**
- **CE3 - Non valida:** Assenza di @
- **CE4 - Non valida:** Assenza del suffisso di dominio (es. `.com`)
- **CE5 - Non valida:** Campo vuoto

Password

- **CE1 - Valida:** Minimo 8 caratteri
- **CE2 - Non valida:** Meno di 8 caratteri
- **CE3 - Non valida:** Campo vuoto

Nome

- **CE1 - Valida:** Inizia con lettera maiuscola, nessun numero
- **CE2 - Non valida:** Campo vuoto
- **CE3 - Non valida:** Inizia con lettera minuscola
- **CE4 - Non valida:** Contiene numeri

Cognome

- **CE1 - Valida:** Inizia con lettera maiuscola, nessun numero
- **CE2 - Non valida:** Campo vuoto

Partita IVA

- **CE1 - Valida:** Esattamente 11 cifre
- **CE2 - Non valida:** Campo vuoto
- **CE3 - Non valida:** Meno di 11 cifre

Casi di Test (R-WECT)

Tabella 1 – Dati in input

ID Test	Nome	Cognome	Password	Email	P.IVA
T01	Marco	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T02	(vuoto)	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T03	marco	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T04	Mar1o	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T05	Marco	(vuoto)	abcdefghijklm	marco@dominio.com	12345678901
T06	Marco	Rossi	abcd	marco@dominio.com	12345678901
T07	Marco	Rossi	(vuoto)	marco@dominio.com	12345678901
T08	Marco	Rossi	abcdefghijklm	marcodominio.com	12345678901
T09	Marco	Rossi	abcdefghijklm	marco@dominio	12345678901
T10	Marco	Rossi	abcdefghijklm	(vuoto)	12345678901
T11	Marco	Rossi	abcdefghijklm	marco@dominio.com	(vuoto)
T12	Marco	Rossi	abcdefghijklm	marco@dominio.com	1234567
T13	Marco	Rossi	abcdefghijklm	admin@dominio.com	12345678901

Tabella 2 – Classi di equivalenza e risultato

ID Test	CE Nome	CE Cognome	CE Password	CE Email	CE P.IVA	Esito Atteso
T01	CE1	CE1	CE1	CE1	CE1	Successo
T02	CE2	CE1	CE1	CE1	CE1	Errore nome vuoto
T03	CE3	CE1	CE1	CE1	CE1	Errore nome minuscolo
T04	CE4	CE1	CE1	CE1	CE1	Errore nome con numero
T05	CE1	CE2	CE1	CE1	CE1	Errore cognome vuoto
T06	CE1	CE1	CE2	CE1	CE1	Password troppo corta
T07	CE1	CE1	CE3	CE1	CE1	Password vuota
T08	CE1	CE1	CE1	CE3	CE1	Email senza '@'
T09	CE1	CE1	CE1	CE4	CE1	Email senza suffisso
T10	CE1	CE1	CE1	CE5	CE1	Email vuota
T11	CE1	CE1	CE1	CE1	CE2	P.IVA vuota
T12	CE1	CE1	CE1	CE1	CE3	P.IVA troppo corta
T13	CE1	CE1	CE1	CE2	CE1	Email già presente

6.1.2 Test: updateEstateAgent

Il metodo `updateEstateAgent(Estate estate, Agent agent)` consente, dato l'ID di un'immobile (`idEstate`) e l'email di un agente (`emailAgent`), di associare l'immobile specificato al relativo agente.

Criteri di Validazione

`idEstate`

- **CE1 - Valido:** `idEstate` ≥ 0 e corrisponde a un'immobile presente nel database.
- **CE2 - Valido:** `idEstate` ≥ 0 ma non corrisponde a un'immobile presente nel database.
- **CE3 - Non valido:** `idEstate` < 0 .

`emailAgent`

- **CE1 - Valido:** non vuota, correttamente formattata e corrispondente a un agente presente nel database.
- **CE2 - Valido:** non vuota, correttamente formattata, ma non corrispondente a un agente presente nel database.
- **CE3 - Non valido:** stringa vuota.
- **CE4 - Non valido:** non correttamente formattata.

Casi di Test (R-WECT): 5

Tabella 1 – Dati in input

ID Test	<code>idEstate</code>	<code>emailAgent</code>
T01	-1	utente@esempio.com
T02	0	(vuoto)
T03	0	utenteemailsbagliata
T04	0	nonesiste@esempio.com
T05	1	utente@esempio.com

Tabella 2 – Classi di equivalenza e risultati attesi

ID Test	CE <code>idEstate</code>	CE <code>emailAgent</code>	Esito Atteso
T01	CE3	CE1	Errore: <code>idEstate</code> negativo
T02	CE1	CE3	Errore: <code>emailAgent</code> vuota
T03	CE1	CE4	Errore: <code>emailAgent</code> non formattata correttamente
T04	CE1	CE2	Errore: <code>emailAgent</code> non corrispondente a un agente esistente
T05	CE2	CE1	Errore: <code>idEstate</code> non corrispondente a un'immobile esistente

6.1.3 AddAgent test

Il metodo `addAgent(Agent agent)` consente l'aggiunta di un nuovo utente amministratore al sistema. Durante l'aggiunta vengono effettuate varie validazioni sui dati inseriti, tra cui email, password, nome, cognome e partita IVA. In caso di errore, il metodo restituisce un messaggio di errore corrispondente.

Criteri di Validazione

Email

- **CE1 - Valida:** Formato corretto (`xxx@yyy.zzz`)
- **CE2 - Valida ma già presente nel database**
- **CE3 - Non valida:** Assenza di @
- **CE4 - Non valida:** Assenza del suffisso di dominio (es. .com)
- **CE5 - Non valida:** Campo vuoto

Password

- **CE1 - Valida:** Minimo 8 caratteri
- **CE2 - Non valida:** Meno di 8 caratteri
- **CE3 - Non valida:** Campo vuoto

Nome

- **CE1 - Valida:** Inizia con lettera maiuscola, nessun numero
- **CE2 - Non valida:** Campo vuoto
- **CE3 - Non valida:** Inizia con lettera minuscola
- **CE4 - Non valida:** Contiene numeri

Cognome

- **CE1 - Valida:** Inizia con lettera maiuscola, nessun numero
- **CE2 - Non valida:** Campo vuoto

Partita IVA

- **CE1 - Valida:** Esattamente 11 cifre
- **CE2 - Non valida:** Campo vuoto
- **CE3 - Non valida:** Meno di 11 cifre

Casi di Test(R-WECT)

Tabella 1 – Dati in input

ID Test	Nome	Cognome	Password	Email	P.IVA
T01	Marco	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T02	(vuoto)	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T03	marco	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T04	Marlo	Rossi	abcdefghijklm	marco@dominio.com	12345678901
T05	Marco	(vuoto)	abcdefghijklm	marco@dominio.com	12345678901
T06	Marco	Rossi	abcd	marco@dominio.com	12345678901
T07	Marco	Rossi	(vuoto)	marco@dominio.com	12345678901
T08	Marco	Rossi	abcdefghijklm	marcodominio.com	12345678901
T09	Marco	Rossi	abcdefghijklm	marco@dominio	12345678901
T10	Marco	Rossi	abcdefghijklm	(vuoto)	12345678901
T11	Marco	Rossi	abcdefghijklm	marco@dominio.com	(vuoto)
T12	Marco	Rossi	abcdefghijklm	marco@dominio.com	1234567
T13	Marco	Rossi	abcdefghijklm	admin@dominio.com	12345678901

Tabella 2 – Classi di equivalenza e risultato

ID Test	CE Nome	CE Cognome	CE Password	CE Email	CE P.IVA	Esito Atteso
T01	CE1	CE1	CE1	CE1	CE1	Successo
T02	CE2	CE1	CE1	CE1	CE1	Errore nome vuoto
T03	CE3	CE1	CE1	CE1	CE1	Errore nome minuscolo
T04	CE4	CE1	CE1	CE1	CE1	Errore nome con numero
T05	CE1	CE2	CE1	CE1	CE1	Errore cognome vuoto
T06	CE1	CE1	CE2	CE1	CE1	Password troppo corta
T07	CE1	CE1	CE3	CE1	CE1	Password vuota
T08	CE1	CE1	CE1	CE3	CE1	Email senza '@'
T09	CE1	CE1	CE1	CE4	CE1	Email senza suffisso
T10	CE1	CE1	CE1	CE5	CE1	Email vuota
T11	CE1	CE1	CE1	CE1	CE2	P.IVA vuota
T12	CE1	CE1	CE1	CE1	CE3	P.IVA troppo corta
T13	CE1	CE1	CE1	CE2	CE1	Email già presente

6.1.4 applyChangeAcquirente test

Il metodo `applyChangeAcquirente` consente l'aggiornamento dei dati di un agente immobiliare. Durante l'aggiunta vengono effettuate varie validazioni sui dati inseriti, tra cui nome, email, cognome e password. In caso di errore, il metodo restituisce un messaggio di errore corrispondente.

Criteri di Validazione

Nome

- **NCE1 - Valida:** Inizia con lettera maiuscola
- **NCE2 - Non valida:** Campo vuoto
- **NCE3 - Non valida:** Non inizia con lettera maiuscola

Email

- **ECE1 - Valida:** Formato corretto (`xxx@yyy.zzz`)
- **ECE2 - Valida ma già presente nel database**
- **ECE3 - Non valida:** Assenza di @
- **ECE4 - Non valida:** Assenza del suffisso di dominio (es. .com)
- **ECE5 - Non valida:** Campo vuoto

Cognome

- **NCE1 - Valida:** Inizia con lettera maiuscola
- **NCE2 - Non valida:** Campo vuoto
- **NCE3 - Non valida:** Non inizia con lettera maiuscola

Password

- **PCE1 - Valida:** Almeno 8 caratteri
- **PCE2 - Non valida:** Meno di 8 caratteri
- **PCE3 - Valida:** Campo null (password opzionale)

Casi di Test (R-WECT)

Tabella 1 – Dati in input

ID Test	Nome	Email	Cognome	Password
1	mario	test@email.it	Rossi	pass1234
2	(vuoto)	test@email.it	Rossi	pass1234
3	Mario	emailGià@email.it	Rossi	null
4	Mario	testdom.it	Rossi	pass1234
5	Mario	test@email	Rossi	pass1234
6	Mario	(vuoto)	Rossi	pass1234
7	Mario	test@email.it	(vuoto)	pass1234
8	Mario	test@email.it	rossi	pass1234
9	Mario	test@email.it	Rossi	pass

10	Mario	test@email.it	Rossi	pass1234
----	-------	---------------	-------	----------

Tabella 2 – Classi di equivalenza e risultato

ID Test	CE Nome	CE Email	CE Cognome	CE Password	Esito Atteso
1	NCE3	ECE1	NCE1	PCE1	Errore di validazione nome
2	NCE2	ECE1	NCE1	PCE1	Errore di validazione nome
3	NCE1	ECE2	NCE1	PCE3	Errore: email già presente
4	NCE1	ECE3	NCE1	PCE1	Errore di validazione email
5	NCE1	ECE4	NCE1	PCE1	Errore di validazione email
6	NCE1	ECE5	NCE1	PCE1	Errore di validazione email
7	NCE1	ECE1	NCE2	PCE1	Errore di validazione cognome
8	NCE1	ECE1	NCE3	PCE1	Errore di validazione cognome
9	NCE1	ECE1	NCE1	PCE2	Errore di validazione password
10	NCE1	ECE1	NCE1	PCE1	Successo creazione admin

6.2 Tecniche di valutazione dell'usabilità

6.2.1 Expert reviews / inspections

Obiettivo

L'obiettivo dell'analisi è valutare l'interfaccia grafica dell'applicazione **UninaEstates25** in termini di usabilità, coerenza visiva e chiarezza informativa, attraverso un processo di *expert review* basato sui principi euristici di Jakob Nielsen.

Metodo

L'interfaccia è stata ispezionata secondo le seguenti categorie:

- Chiarezza delle informazioni
- Coerenza e standard visivi
- Efficienza della navigazione
- Estetica e design
- Compatibilità con dispositivi mobili

Risultati

L'interfaccia si è dimostrata coerente e ben progettata sotto diversi aspetti chiave:

- **Organizzazione dei contenuti:** la gerarchia visiva è chiara e favorisce una lettura immediata delle informazioni relative agli annunci immobiliari.
- **Chiarezza comunicativa:** le etichette testuali e le icone risultano facilmente interpretabili anche da utenti non esperti. Simboli per metri quadrati, stanze, bagni e garage sono utilizzati in maniera appropriata.
- **Design visivo:** l'interfaccia adotta uno stile grafico moderno e coerente, con un buon bilanciamento tra spazi, colori e tipografia.
- **Accessibilità mobile:** la disposizione degli elementi è ottimizzata per l'uso da smartphone. La navigazione tramite barra inferiore risulta fluida e intuitiva.
- **Interattività:** i pulsanti e gli elementi cliccabili (es. "Ricerca Immobile") sono ben evidenziati e facilmente accessibili, garantendo un buon livello di feedback visivo.

Valutazione sintetica

Conclusione

L'interfaccia dell'app **UninaEstates25** presenta un buon livello di maturità progettuale. Pur trattandosi di una versione preliminare, il sistema risulta già altamente utilizzabile e graficamente ben strutturato. L'esperienza utente appare fluida, con contenuti chiari e un design coerente con le aspettative del dominio immobiliare.

Non sono stati rilevati problemi critici. Si raccomanda comunque di validare i contenuti dinamici (es. prezzi e metrature) in fase di integrazione con il back-end.

Categoria	Valutazione (1–5)
Estetica e coerenza visiva	4
Chiarezza delle informazioni	4
Navigabilità e interazione	4
Compatibilità mobile	5
Reattività e feedback	3.5

Tabella 6.9: Valutazione qualitativa dell'interfaccia

6.2.2 Progettazione e conduzione di un esperimento

Procediamo con la fase di usability test, in cui viene descritto il nostro esperimento, condotto con 7 soggetti di età compresa tra 39 e 64 anni. La scelta di questo range è stata effettuata perché rappresenta la fascia preponderante degli utenti ([fonte](#)).

Soggetti reclutati

Tabella 6.10: Soggetti esperimento

Età	Genere	Familiarità [1–5]
64	Uomo	4
62	Donna	1
60	Uomo	5
58	Donna	3
54	Donna	1
40	Uomo	4
39	Donna	5

Procedura sperimentale

La procedura ha previsto l'esecuzione delle seguenti task:

1. Acquirente: Ricerca per città
2. Acquirente: Prenotazione appuntamento estivo
3. Acquirente: Ricerca per raggio geografico
4. Agente immobiliare: Accettazione di un appuntamento

L'ambiente di esecuzione era un appartamento, con l'applicazione testata tramite un emulatore Android su un computer desktop.

Metriche

Di seguito vengono riportati i dati osservati.

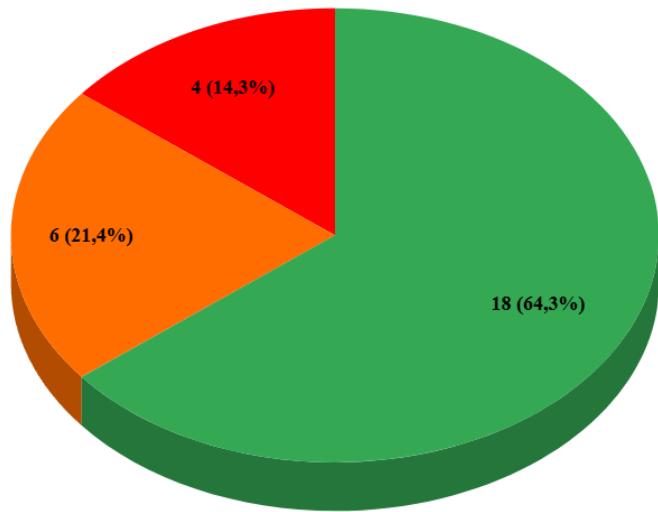
Semaforo Classificazione dei risultati*Semaforo %*

Figura 6.1: Semaforo – Classificazione dei risultati

I criteri di classificazione sono i seguenti:

- **Verde**: esecuzione fluida, errori irrilevanti, buona scioltezza;
- **Arancione**: comportamento sufficiente con alcuni errori minori;
- **Rosso**: esecuzione lenta e frequente presenza di errori.

Distribuzione Semaforo Analisi della distribuzione dei risultati

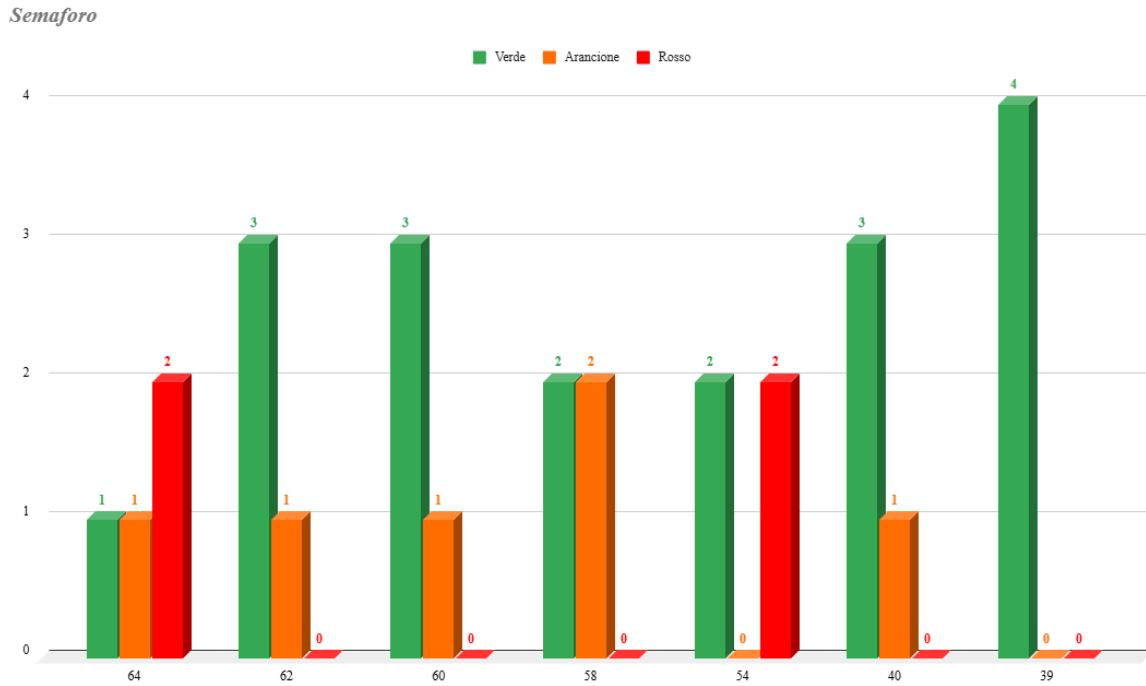


Figura 6.2: Semaforo – Distribuzione dei risultati

Dall'analisi dei dati emerge una lieve correlazione tra i risultati ottenuti e l'età dei partecipanti, mentre il genere non sembra avere alcuna influenza significativa.

Di particolare rilievo è invece la forte correlazione osservata con il livello di familiarità autovalutato (“Avvezzo”) riportato nella tabella precedente. Questo dato rappresenta un indicatore positivo: suggerisce che gli utenti più esperti nel settore digitale hanno riconosciuto nel nostro software numerosi standard di usabilità consolidati, ritrovando modalità d’interazione già note e percependo l’interfaccia come intuitiva e coerente. Ciò conferma la capacità del sistema di aderire a pratiche progettuali consolidate, facilitando l’apprendimento e l’uso anche in contesti eterogenei.

Medie valori Medie di tempo, errori e richieste di aiuto

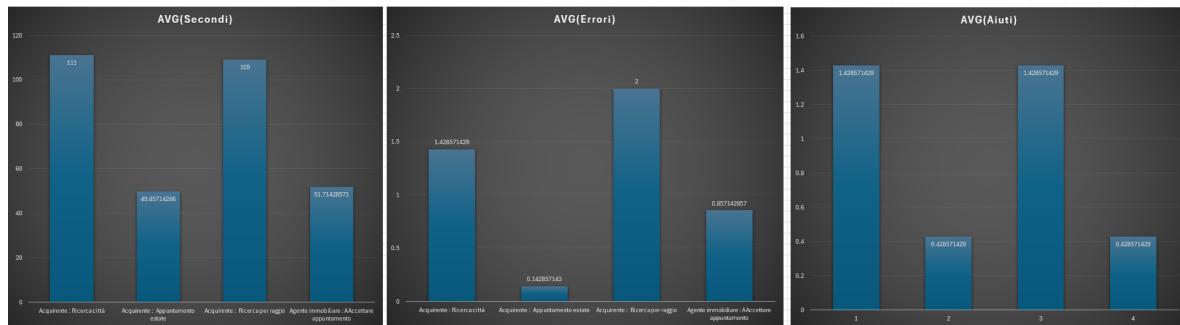


Figura 6.3: Medie – Tempo, errori e richieste di aiuto

L’analisi dei dati evidenzia buoni risultati per quanto riguarda le task relative alle prenotazioni, mentre le operazioni di ricerca sembrano presentare maggiori difficoltà per gli utenti.

In base ai feedback raccolti durante i test, sono emerse alcune perplessità specifiche, che abbiamo successivamente analizzato e risolto:

- **Menu di navigazione:**



Figura 6.4: Menu di navigazione

Alcuni utenti hanno manifestato confusione nell'utilizzo del menu di navigazione, in particolare nel distinguere la funzione di ricerca dal flusso di navigazione generale.

- **Tasto per la sezione di ricerca:**



Figura 6.5: Tasto per la ricerca

È stata rilevata confusione anche nell'identificazione e nell'uso del tasto dedicato alla ricerca, che alcuni partecipanti hanno percepito come poco intuitivo.

- **Dimensioni e importanza visiva degli elementi:** Una parte significativa del feedback riguarda le dimensioni ridotte di alcuni elementi dell'interfaccia, che non ricevevano la giusta attenzione rispetto ad altri. Per risolvere questo problema abbiamo ridefinito le proporzioni generali delle schermate, aumentando la dimensione e la visibilità degli elementi più importanti.

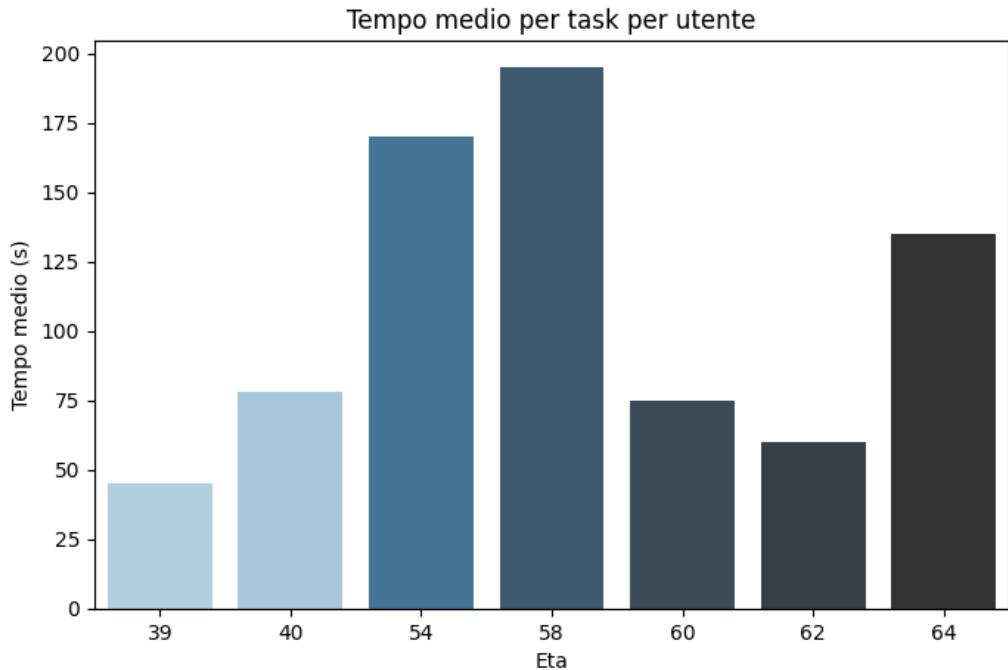


Figura 6.6: Tempo medio per utente

Tempo medio per utente L’analisi del tempo medio impiegato da ciascun partecipante evidenzia nuovamente una correlazione tra i risultati e il livello di familiarità autovalutato (“Avvezzo”) riportato nella tabella precedente.

In particolare, gli utenti con maggiore esperienza nell’utilizzo di strumenti digitali tendono a completare le task con tempi significativamente inferiori rispetto a quelli meno avvezzi. Questo dato rafforza l’ipotesi che il nostro sistema, pur essendo progettato per un’utenza ampia, risulta immediatamente più intuitivo per chi possiede già competenze di base nell’interazione con applicazioni digitali.

Tuttavia, è importante sottolineare casistiche in cui anche utenti meno esperti sono riusciti a fare un buon risultato.

Questionario

A fine esperimento abbiamo sottoposto ai soggetti le seguenti domande:

1. Penso che userei frequentemente questa app per cercare o gestire immobili.
2. Ho trovato l’app inutilmente complessa per le operazioni di ricerca o contatto.
3. Le funzionalità dell’app sono ben integrate e intuitive.
4. La maggior parte delle persone imparerebbe rapidamente a usare quest’app per immobili.
5. Ho trovato l’app troppo complicata per le esigenze di un utente medio.

Risultati

Analisi dei risultati del questionario

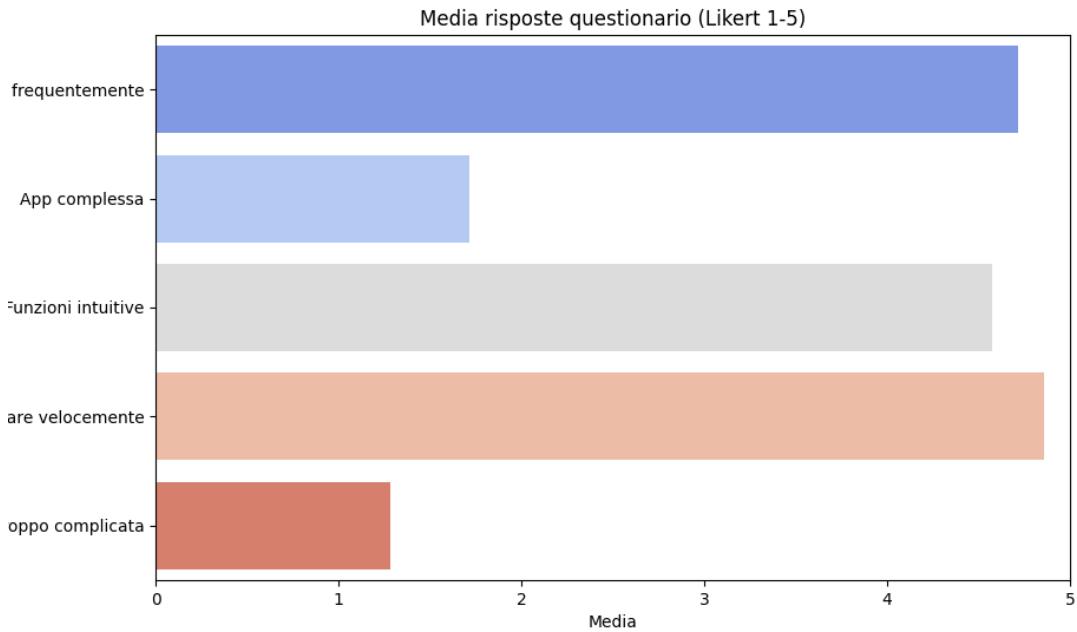


Figura 6.7: Media dei punteggi del questionario

L’analisi dei risultati del questionario mostra complessivamente punteggi molto positivi, a conferma della buona percezione che gli utenti hanno avuto dell’esperienza d’uso offerta dall’applicazione.

Di particolare rilievo è la domanda numero 4, la quale indaga la facilità di utilizzo per utenti con scarsa familiarità con strumenti digitali. Il punteggio ottenuto su questo punto evidenzia come il nostro obiettivo di progettare un’applicazione con un approccio “*focus on novice users*” sia stato raggiunto con successo.

Questo risultato suggerisce che le scelte progettuali adottate hanno contribuito a rendere l’app accessibile anche a utenti poco esperti, riducendo al minimo la curva di apprendimento.