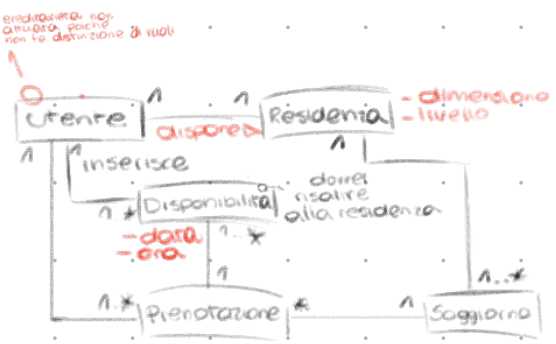
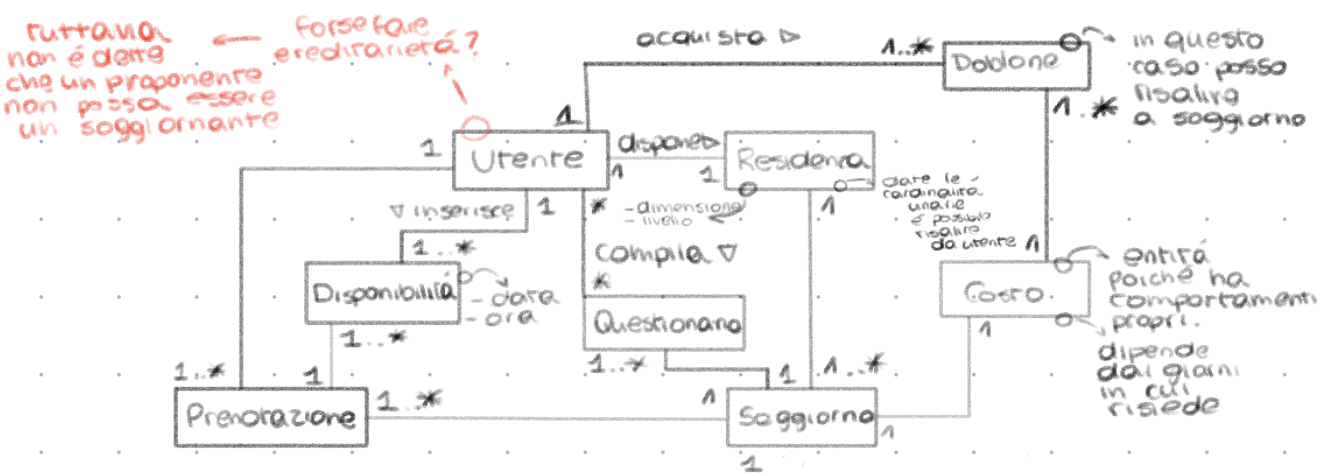
Domain model

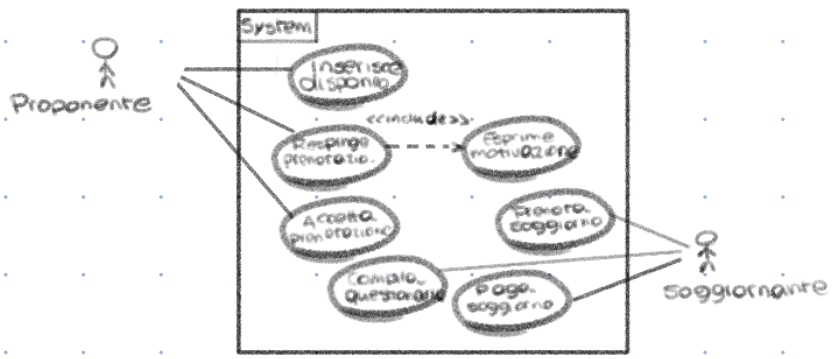
Prima stesura



Final model



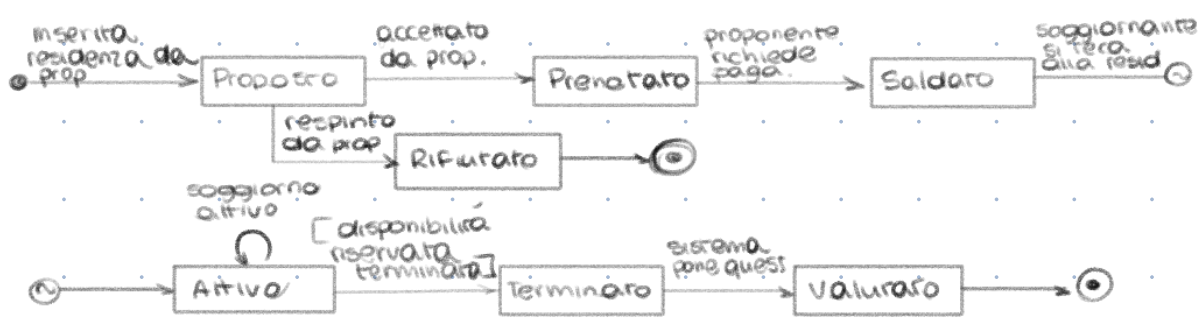
Use case



UC: Respinge prenotazione
Pre: Proponente autenticato
Proponente inserito disponibilità per la propria residenza
1. Proponente seleziona prenotazioni
2. Sistema visualizza prenotazioni
3. Proponente declina offerte
4. Sistema richiede inserimento motivazione
5. Proponente inserisce motivazione ed invia
Post: Soggiornante visualizza respinta prenotazione

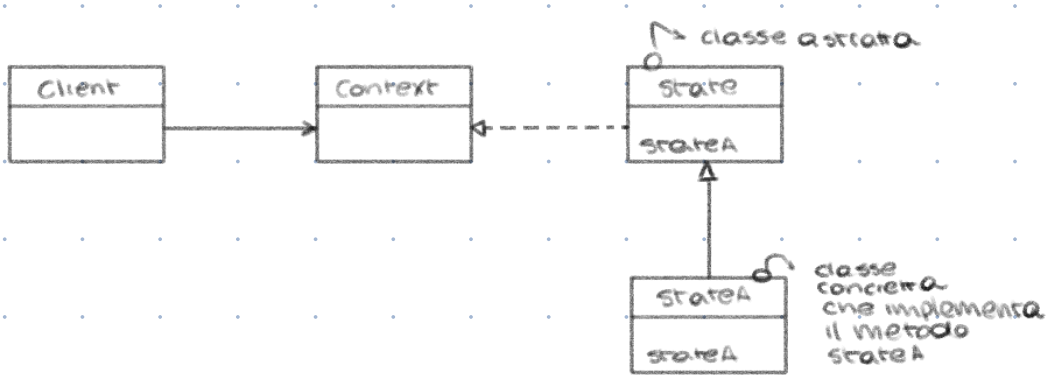
Stare diagram

① Proposto
② Accettato
③ Rifiutato
④ Saldato
⑤ Valutato
⑥ Prenotato



State

State è un pattern comportamentale del catalogo GoF, basato sulla concezione della composizione, ossia ogni classe software adegua un unico comportamento.
L'intento di state consiste nel favorire l'intercambiabilità comportamentale di un oggetto rispetto allo stato corrente che lo contraddistingue, durante run-time.
Pertanto state permette di variare funzionalità di istanze durante il processo esecutivo, per cui per certi aspetti adegua un approccio descritto da Strategy: ma vige una differenza sostanziale, non sono classi esterne ad imporre qualora si debba adeguare intercambiabilità. Ma l'interfaccia pone quando attuare il meccanismo.
A livello implementativo si compone di un'interfaccia, che mantenga un riferimento dello stato corrente dell'oggetto, e un'estensione comportamentale indotta da una gerarchia che sviluppi funzionalità affini agli stati acquisibili.
In tal senso, il processo è totalmente automatizzato, poiché in completa autonomia sarà ricondotto il flusso di controllo destinato alla classe che adotti la funzionalità affine allo stato.
Proseguendo, il pattern presentato soddisfa i principi SOLID, tra cui dipendancy inversion principle, grazie alla sovrapposizione di un'interfaccia affinché sia posta schermatura tra le due entità, in modo tale che modifiche non abbiano ripercussioni, e Liskov substitution, dato che non è violata compatibilità comportamentale, poiché sottoclassi non sovrascrivono interamente il metodo ereditato, specializzando solamente per step mancanti, per cui ogni combinazione si traduce in una nuova classe della business logic.



3) Principi agile

Agile software development rappresenta una stretta necessità di team di sviluppo, affinché sia posta una metodica incentrata sull'implementazione della soluzione software, piuttosto che concretarsi su attività di contorno.
Da questa sommativa descrizione nasce il manifesto, il quale pone alcuni principi relativi allo sviluppo software, avviando alla stesura di una documentazione esaustiva; proprio per questo ragione spesso si tende ad adeguare attività descrittive al di fuori di momenti legati all'implementazione, data la natura dinamica di un sistema software, da cui si evince l'inutilità di un'analisi di specifica attuata fino ai minimi particolari.
Proseguendo, le supposizioni sono suddivise in

- adeguare codice piuttosto che elaborare documentazioni
- favorire la comunicazione tra sviluppatori, principale strumento per diffondere conoscenza all'interno del team
- attuare una comunicazione con i clienti, piuttosto che una negoziazione, affinché l'obiettivo principale risulti la totale soddisfazione dei requisiti funzionali
- reagire prontamente alle modifiche, favorendo vantaggio competitivo.

Pertanto la volontà principale consiste nel distogliere tutte le attività di contorno, le quali provocherebbero elevate perdite di tempo per lo sviluppo e l'implementazione compertamentali che possano rispondere realmente all'esigenze degli stakeholders.