Università Degli Studi di Napoli

Parthenope

Progetto di Ingegneria del Software

B&B Il Sole
System Design Document

Ilardo Gianluca – 0124/2368
Orlando Giuseppe – 0124/2053
Scuotto Ilaria – 0124/2123
Terrazzano Gianfranco – 0124/2052
Zibaldo Francesco – 0124/2176

Prof. Antonino Staiano

17/01/2022

B&B Il Sole

Indice

Capitolo 1 *4*

Introduzione **4**

1.1 Scopo del sistema 4

1.2 Obiettivi di progettazione 4

1.3 Definizioni, acronimi e abbreviazioni 5

1.3.1 Definizioni 5

1.3.2 Abbreviazioni 5

1.3.3 Acronimi 5

1.4 Riferimenti 5

1.5 Panoramica 6

Capitolo 2 *7*

Sistema corrente **7**

Capitolo 3 *8*

Sistema Proposto **8**

3.1 Panoramica 8

3.2 Decomposizione del sistema 9

3.3 Hardware/Software mapping 10

3.4 Gestione dei dati persistenti 10

3.5 Controllo accessi e sicurezza 11

3.6 Decisioni sul flusso di controllo globale 11

3.7 Condizioni limite 12

Capitolo 4 *13*

Servizi del sottosistema **13**

Capitolo 5 *14*

Glossario **14**

Capitolo 1

Introduzione

1.1 Scopo del sistema

L'obiettivo del sistema B&B Il Sole consiste nel fornire al proprietario della catena di B&B un supporto semplice, che garantisca al contempo la gestione interna delle strutture (anche dal punto di vista finanziario con l'aiuto di un ragioniere) e gli permetta di interfacciarsi coi propri clienti (con l'aiuto della receptionist), automatizzando la procedura di prenotazione.

Molta importanza è stata data alla user experience, cercando di nascondere quanto più possibile le complesse logiche architetturali agli utilizzatori del sistema. Sia il proprietario che i clienti saranno completamente guidati da form preimpostati in grado di semplificare e velocizzare i processi di prenotazione, registrazione, login, gestione finanziaria.

1.2 Obiettivi di progettazione

- **Usabilità**
  B&B Il Sole dovrebbe essere intuitiva da usare e l'interfaccia utente dovrebbe essere semplice da capire.
- **Accessibilità**
  B&B Il Sole dovrebbe essere fruibile con facilità da una qualsiasi tipologia d'utente.
- **Sicurezza**
  B&B Il Sole dovrebbe garantire che i dati di accesso degli utenti siano ben protetti e inaccessibili dall'esterno.
- **Portabilità**
  B&B Il Sole dovrebbe essere sviluppato in modo da essere multipiattaforma.
- **Immediatezza**
  B&B Il Sole dovrebbe fornire risposte immediate ad ogni richiesta dell'utente.

1.3 Definizioni, acronimi e abbreviazioni

All’interno di questo documento, è possibile incontrare la seguente terminologia:

1.3.1 Definizioni

- **Form** indica la parte di interfaccia utente di un'applicazione web che consente all'utente client di inserire e inviare al web server/application server uno o più dati liberamente digitati dallo stesso sulla tastiera attraverso l'uso di componenti grafici.

1.3.2 Abbreviazioni

- **DB** Database.
- **API** Application Programming Interface.

1.3.3 Acronimi

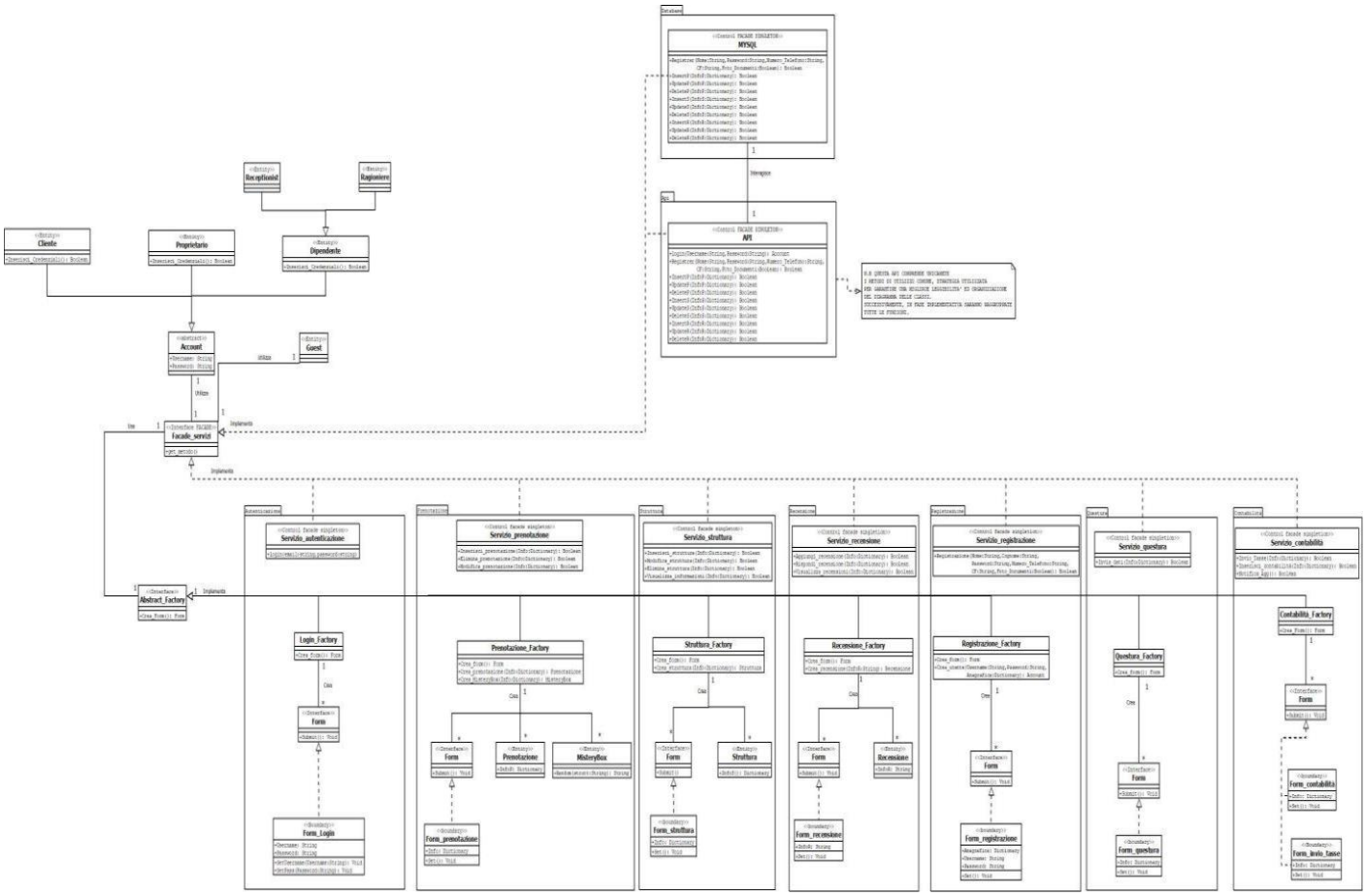
- **B&B** Abbreviazione commerciale di Bed and Breakfast.
- **API** Application Programming Interface.

1.4 Riferimenti

Per una migliore visione d’insieme del progetto, si consideri il documento RAD. Per una migliore visualizzazione delle immagini, si considerino i relativi file.jpeg.

1.5 Panoramica

Abbiamo raffinato il diagramma delle classi ottenuto in fase di analisi, anche attraverso l'uso dei design pattern.



Capitolo 2

Sistema corrente

B&B Il Sole è un progetto Greenfield, pertanto è privo di vincoli provenienti da un lavoro precedente. Abbiamo però tratto ispirazione per alcune questioni da altri sistemi che svolgono delle mansioni affini (https://www.booking.com, https://www.hotel-bb.com).

Capitolo 3

Sistema Proposto

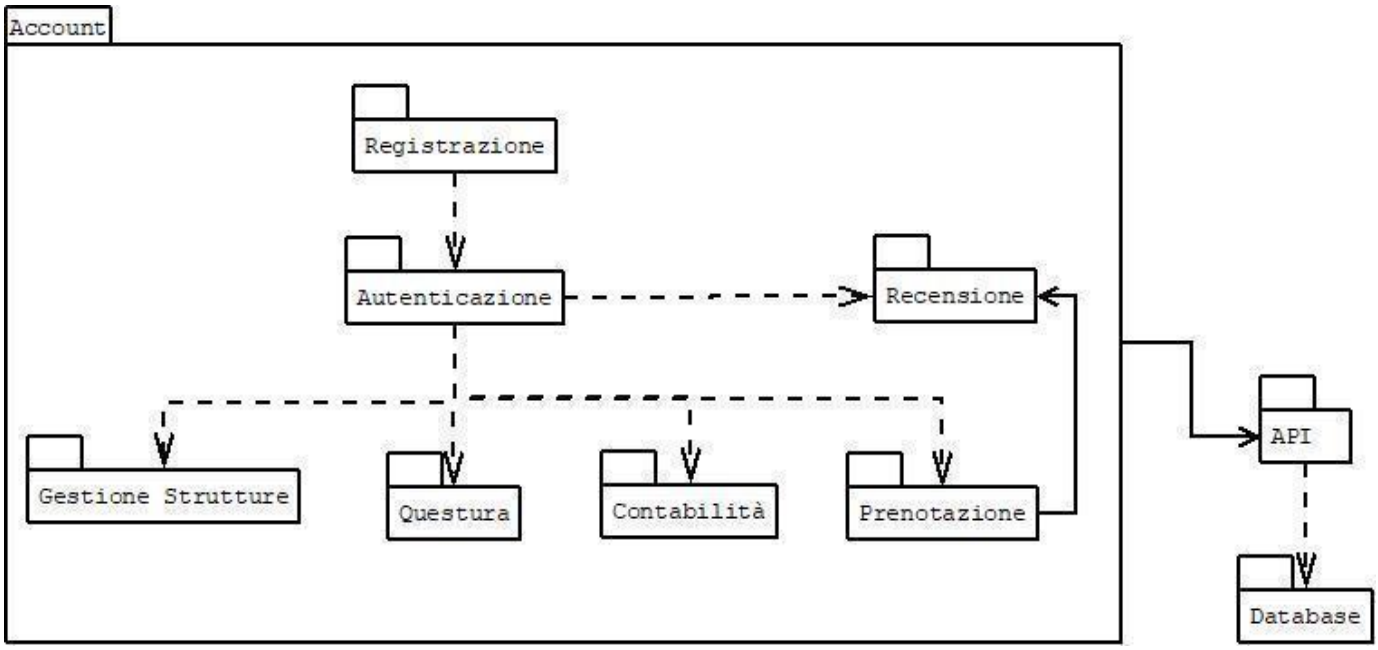
3.1 Panoramica

Lo stile architetturale scelto per il nostro sistema è il client/server. Abbiamo individuato quattro tipologie di client:

- un primo, pensato per l'utilizzo del proprietario della catena, il quale permette di svolgere operazioni inerenti all'aggiunta, la modifica e l'eliminazione delle strutture, di verificare l'andamento finanziario dell'attività e di assolvere alle principali questioni legali e burocratiche;
- un secondo, immaginato per l'uso da parte di clienti, mediante il quale è possibile gestire il proprio account, visualizzare informazioni e servizi, rilasciare e consultare recensioni, effettuare e gestire prenotazioni;
- un terzo, immaginato per l'uso da parte dei guest (possibili futuri clienti), attraverso cui è possibile creare un proprio account, visualizzare informazioni e recensioni;
- un quarto, destinato ai dipendenti (ragioniere e receptionist), che consente di assolvere alle questioni finanziarie e di interfacciarsi con i clienti, permettendo di rispondere alle recensioni.

Questo tipo di stile architetturale è stato preferito per rendere il sistema maggiormente fruibile, poiché garantisce l'accesso da un qualsiasi dispositivo.

3.2 Decomposizione del sistema



Autenticazione all' interno di essa vengono gestite le fasi attraverso cui un utente, mediante l'immissione di credenziali, viene autenticato dal sistema;

Prenotazione permette all' utente registrato di effettuare una nuova prenotazione, di modificarla o eventualmente di eliminarla sottostando a dei vincoli;

Gestione_Strutture rappresenta un'area riservata, accessibile al proprietario, che permette di effettuare diverse operazioni quali inserimento, modifica ed eliminazione della struttura. Consente inoltre al guest di visualizzare le informazioni relative alle varie strutture;

Recensione all'interno di essa il cliente ha la possibilità di inserire una nuova recensione o visualizzare quelle già esistenti, l'user può consultare le recensioni già inserite e la receptionist può rispondere alle recensioni;

Registrazione area accessibile al guest, che prevede al suo interno la possibilità di registrare un nuovo utente alla piattaforma mediante l'inserimento di dati;

Questura area accessibile al proprietario, al cui interno è possibile inviare i dati dei clienti alla questura;

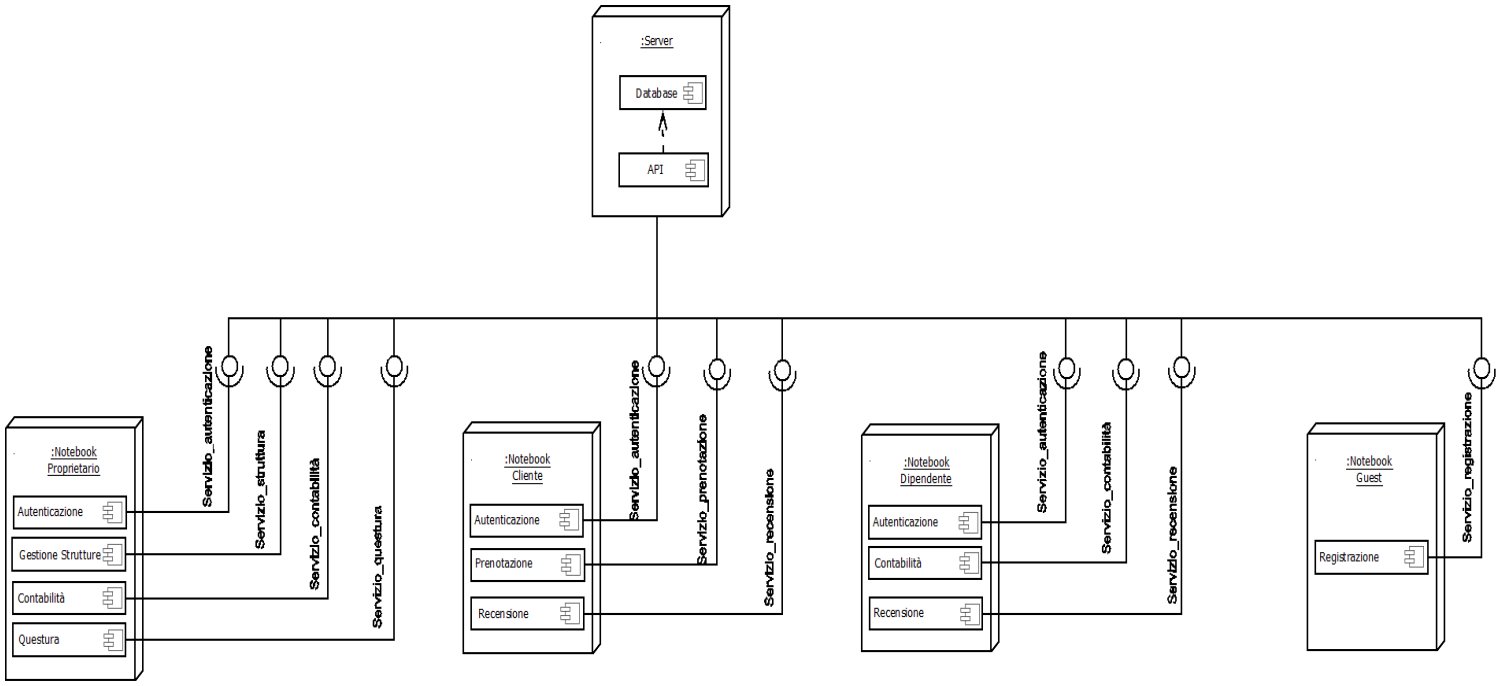
Contabilità area riservata, accessibile solo al proprietario e al ragioniere, in cui è possibile effettuare operazioni di natura contabile;

Database si riferisce al salvataggio dei dati del sistema;

API L'API sono set di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi;

3.3 Hardware/Software mapping

Il sistema deve essere fruibile e accessibile da un qualsiasi notebook, che abbia però un account attivo che consenta l'autenticazione.



3.4 Gestione dei dati persistenti

B&B Il Sole è un'applicazione di booking che si occupa di gestire sia i compiti del proprietario e dei vari lavoratori sia i compiti dei clienti. Ciò comporta il dover lavorare con una quantità significativa di dati, il che ci ha portato ad utilizzare il database relazionale MySQL, al fine di tenere traccia di tutti i dati degli utenti (ogni utente possiede un proprio account con dati di accesso ed informazioni personali), delle prenotazioni e dell'andamento dell'attività. Questa scelta consentirà una maggiore affidabilità, efficienza e privatezza dei dati.

3.5 Controllo accessi e sicurezza

Oggetti/attori	Cliente	Proprietario	Dipendente	Guest
Servizio_autenticazione	Login()	Login()	Login()	-
Servizio_prenotazione	Inserisci_prenotazione() Elimina_prenotazione() Modifica_prenotazione()	-	-	-
Servizio_struttura	Visualizza_informazioni()	Inserisci_struttura() Modifica_struttura() Elimina_struttura()	-	Visualizza_informazioni()
Servizio_recensione	Aggiungi_recensione() Visualizza_recensioni()	-	Rispondi_recensione()	Visualizza_recensioni()
Servizio_registrazione	-	-	-	Registrazione()
Servizio_questura	-	Invia_dati()	-	-
Servizio_contabilità	-	Invio_tasse()	Inserisci_contabilità()	-

3.6 Decisioni sul flusso di controllo globale

Nel nostro sistema si presentano diverse richieste per il server; dunque, si è deciso di gestirle con un'architettura thread based.
I thread sono oggetti all'interno di un processo che eseguono istruzioni di programma.
I thread consentono operazioni simultanee all'interno di un processo, in modo che un processo possa eseguire diverse parti del suo programma contemporaneamente su processori diversi.
Un'architettura basata su thread offre i seguenti vantaggi:

- Cambio di contesto più rapido;
- Routine di allocazione dell'area globale del sistema più semplice, perché non richiede l'uso della memoria condivisa;
- Generazione più rapida di nuove connessioni, perché i thread vengono creati più rapidamente dei processi;
- Utilizzo di memoria ridotto, perché i thread condividono più strutture di dati rispetto ai processi.

3.7 Condizioni limite

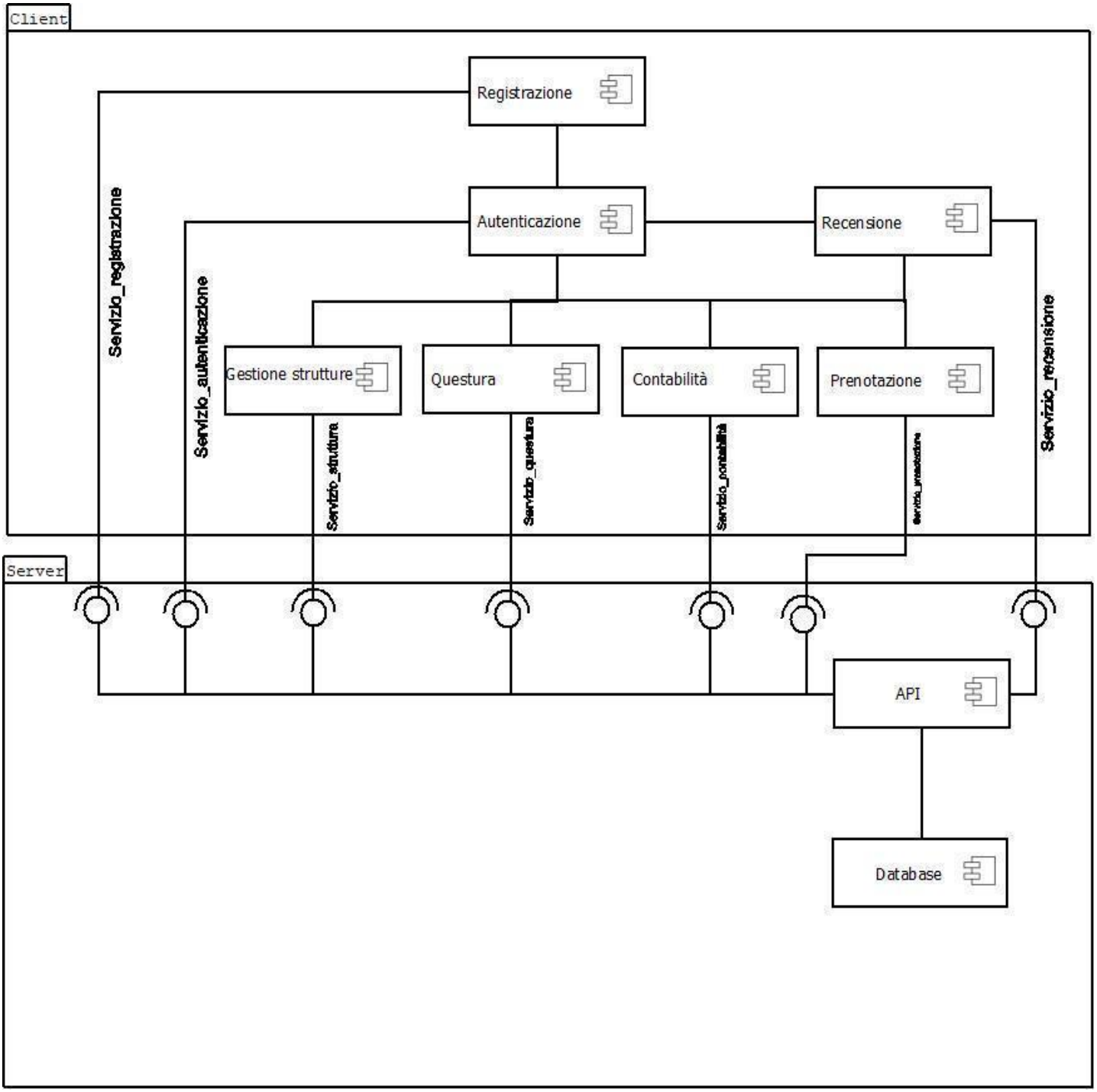
In fase di avvio, in base al client che si collega, saranno mostrate diverse informazioni. Sarà possibile accedere ai vari servizi tramite un'interfaccia di facciata.

In base alla nostra progettazione, a nostro giudizio, i sottosistemi sono indipendenti tra di loro. Essi comunicano al database i relativi aggiornamenti.

Queste funzionalità incrementano la probabilità di eventuali crash/anomalie del sistema. Il nostro applicativo non prevede un server proprietario, il quale effettua punti di backup/ripristino ad intervalli di tempo regolari. Ovviamente, effettuare un ripristino nel caso di una grave anomalia, comporta la perdita dei dati inseriti successivamente al punto di ripristino. Per questo motivo, in caso di guasto, l'amministratore provvederà a risolvere il problema prima di utilizzare un punto di backup.

Capitolo 4

Servizi del sottosistema



Capitolo 5

Glossario

Account Generico utente registrato alla piattaforma.

Proprietario Entità che specializza Account e rappresenta gli attributi e i metodi specifici del proprietario.

Cliente Entità che specializza Account e rappresenta gli attributi e i metodi specifici del cliente.

Dipendente Entità che specializza Account e rappresenta gli attributi e i metodi specifici del dipendente.

Guest Generico visitatore del sito non registrato alla piattaforma.

Facade_servizi È un'interfaccia che conosce le classi nel sottosistema e che delega le richieste del client agli oggetti appropriati. Per costruire questa interfaccia di facciata abbiamo utilizzo il pattern strutturale Facade, poiché mette a disposizione un'interfaccia unificata per implementare un sistema complesso, la quale fornisce un punto di accesso per ogni sottosistema, promuovendo la portabilità e l'indipendenza dei sottosistemi.

Servizio_autenticazione Entità che rappresenta il punto di accesso al sottosistema di autenticazione e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

Servizio_registrazione Entità che rappresenta il punto di accesso al sottosistema di registrazione e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

Servizio_prenotazione Entità che rappresenta il punto di accesso al sottosistema di prenotazione e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

Servizio_struttura Entità che rappresenta il punto di accesso al sottosistema di gestione strutture e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

Servizio_recensione Entità che rappresenta il punto di accesso al sottosistema di recensione e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

Servizio_contabilità Entità che rappresenta il punto di accesso al sottosistema di contabilità e ne implementa le funzionalità. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

AbstractFactory Abbiamo utilizzato questo pattern perché fornisce gli strumenti necessari alla creazione di famiglie di oggetti. Il client è indipendente dalle classi utilizzate per implementare gli oggetti, poiché la factory incapsula il processo di creazione del prodotto. In questo mondo la manipolazione degli oggetti avviene attraverso interfacce astratte.

Form Interfaccia implementata da oggetti che si occupano della creazione dei form, mette a disposizione il metodo Submit.

Registrazione_Factory Implementa AbstractFactory, crea il form necessario alla registrazione, inoltre si occupa della creazione di oggetti di tipo Account tramite il metodo Crea_Utente.

Form_registrazione Form creato da Registrazione_factory, necessario per registrarsi al sistema, include tutti i dati anagrafici.

Questura_Factory Implementa AbstractFactory, crea il form per l'invio di dati alla questura.

Form_questura Form creato da Questura_Factory, necessario per inviare i dati alla questura.

Contabilità_Factory Implementa AbstractFactory, crea il form per la gestione della contabilità.

Form_contabilità Form creato da Contabilità_Factory, necessario per inserire la contabilità nel sistema.

Form_invio_tasse Form creato da Contabilità_Factory, necessario per inviare i dati relative alle tasse all'ufficio del turismo.

Login_Factory Implementa Abstract_Factory e crea il form necessario ad effettuare l'autenticazione

Form_Login Form necessario all'autenticazione

Prenotazione_Factory Implementa Abstract_Factory e crea il form necessario ad effettuare una prenotazione. Inoltre, permette la creazione di oggetti di tipo Prenotazione e MisteryBox.

Form_Prenotazione Form necessario ad effettuare una prenotazione.

Prenotazione Classe che rappresenta le prenotazioni effettuate dai clienti e contiene tutte le informazioni inerenti ad esse.

MisteryBox Classe che rappresenta le prenotazioni effettuate attraverso il metodo Random che consente ai clienti, stabilite le date, di prenotare in una qualsiasi delle strutture della catena.

Struttura_Factory Implementa Abstract_Factory e crea il form necessario alla gestione delle strutture. Inoltre, permette la creazione di oggetti di tipo Struttura.

Form_Struttura Form necessario alla gestione delle strutture.

Struttura Classe che rappresenta tutte le strutture della catena B&B Il Sole e ne contiene le informazioni inerenti.

Recensione_Factory Implementa Abstract_Factory e crea il form necessario all'inserimento di una recensione da parte di un cliente. Inoltre, permette la creazione di oggetti di tipo Recensione.

Form_Recensione Form necessario all'inserimento di una recensione da parte di un cliente.

Recensione Classe che rappresenta le recensioni inserite dai clienti; esse possono essere anche visualizzate dal guest e ottengono risposte da parte della receptionist.

Database Classe che rappresenta il punto di ingresso al database. Contiene tutte le funzioni che permettono di effettuare operazioni sul database (inserimento, cancellazione e aggiornamento dei dati). Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.

API Classe che permette ai Service di interfacciarsi con il database fornendo tutte le funzioni atte a modificare l'interazione con il Database. Si è scelto il pattern singleton, il quale crea una sola istanza, garantendo un maggiore controllo sulle modalità di accesso dei client.