Progetto di Ingegneria del Software 2022/23
Università Ca' Foscari Venezia

Piano di Progettazione

1.0

Team Angio

27/11/2022

Document Informations

Progetto Standard	MyLocalBooking	MLB
Deliverable	Piano di Progettazione	
Data di Consegna	27/11/2022	
Team Leader	Pietro Visconti	885448@stud.unive.it
Team members	Nicola Marizza Pietro Donega Mirco Mellara	887004@stud.unive.it 881909@stud.unive.it 882963@stud.unive.it

Document History

Version	Issue Date	Stage	Changes	Contributors
1.0	27/11/2022	Draft	Creazione documento	Mellara Mirco

Indice

Piano di progettazione

Introduzione

L'obiettivo di questo documento è stabilire la struttura globale del nostro sistema software attraverso l'uso di sottosistemi e moduli, in particolare spiegando come avviene la gestione dei dati e la comunicazione attraverso le varie classi che compongono tale architettura.
Inoltre verranno mostrate alcune bozze delle interfacce che costituiranno l'applicazione stessa.

Glossario

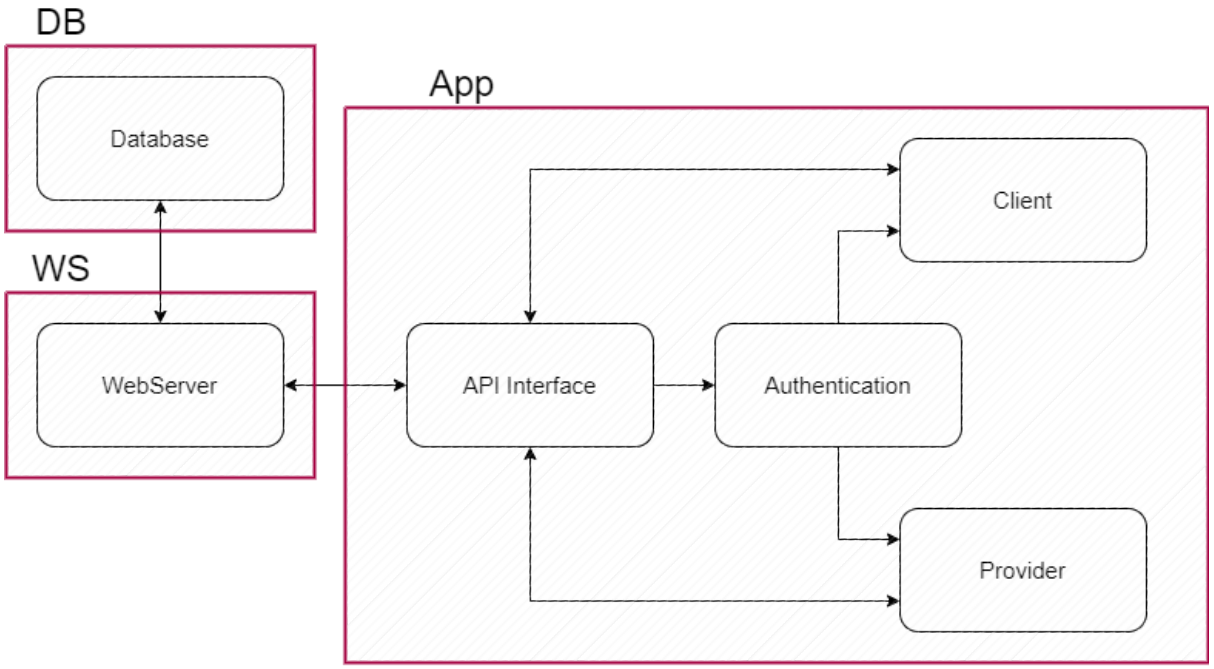
Sottosistema: Una componente del sistema, che è in grado di funzionare in autonomia. Può essere composta da più moduli
Tier: Layer, grado, indica un particolare livello della struttura architetturale.
Sessione: Periodo in cui l'utente utilizza l'app, dopo essersi autenticato, fino al momento di logout.

Architettura del Sistema

Struttura:

Il nostro Sistema è composto da 3 sottosistemi:
DB: Al suo interno vi è il database centrale
WS: Web Server, dove viene hostato il server a cui l'applicazione effettua le richieste
App: Applicazione con cui l'utente interagisce direttamente, dopo l'installazione di essa presso il dispositivo dell'utente. Essa è composta di vari package:

- API Interface
- Authentication
- Client
- Provider



Modello gestione dati

Essendo il nostro un sistema di tipo "Client-Server" (vedi sez. successiva), ogni sottosistema può essere dotato di un **proprio** database.
Nel nostro caso però, vi sono solo due sottosistemi che ne posseggono uno, ossia quello contenente il **Database centrale** (anche detto Database "principale"), chiamato appunto "DB" e quello relativo all'app.

Sotto a questa scelta però vi è un processo di scrittura/lettura dati più semplice, infatti, ogni sottosistema…

- Quando deve **leggere** dei dati, li raccoglie dal database principale (comunicando attraverso eventuali livelli intermedi) e li salva a runtime, dopo averli convertiti in un formato supportato al proprio livello; successivamente al loro utilizzo (o nel caso in cui venga terminata l'applicazione), i dati salvati a runtime verranno eliminati, mentre resteranno fisicamente salvati all'interno del database "principale".
- Quando deve **scrivere**, semplicemente li passa al db "principale" (comunicando attraverso eventuali livelli intermedi), effettuando le opportune operazioni di conversione lungo il tragitto. I dati saranno così fisicamente salvati solo nel db "principale".

Da quel che è stato definito fino ad ora, sembra infatti che l'unico database effettivo esistente, sia quello "principale", ma non è così. Vi è infatti un database particolare per il sottosistema "App", che corrisponde alla **memoria del dispositivo** su cui viene installata.
I dati in questo caso sono salvati in formato **.JSON**

In questo caso però, vengono salvate solo poche informazioni, relative alla "**Sessione**" dell'utente, qualora esso venga autenticato correttamente.
Queste informazioni servono a mantenere autenticato l'utente anche in caso l'app venga terminata, senza forzarlo ad una procedura di login ad ogni singolo avvio dell'applicazione.

Modello dei Dati e del Controllo

Modello dati

Il modello dei dati per cui abbiamo optato è quello **client-server**, che si adatta al nostro sistema distribuito.
Abbiamo effettuato questa scelta perché desideravamo una distribuzione dei dati semplice e veloce tramite l'appoggio alla rete.

Vantaggi:

- Distribuzione semplice dei dati
- Si appoggia alla rete, andando ad abbassare i costi relativi all'hardware
- Scalabile facilmente

Svantaggi:

- Non c'è un modello dei dati condiviso da tutti i sottosistemi
- Gestione di dati ridondante in ogni server
- Non c'è un registro centrale dei servizi: può risultare difficile sapere quali dati/servizi sono disponibili

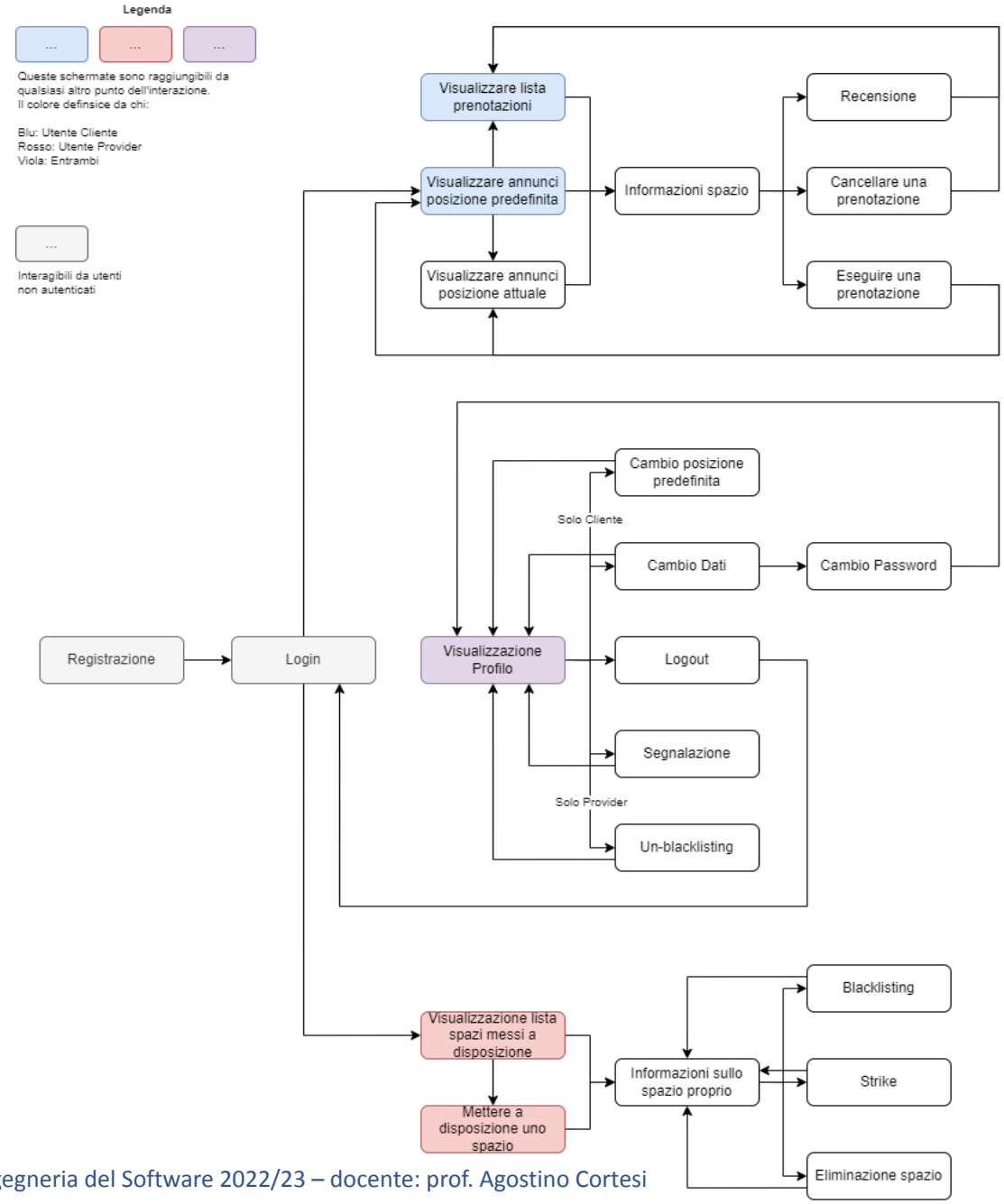
Per ovviare allo svantaggio dei dati ridondanti, si veda la sezione sopra, abbiamo optato per avere la persistenza dei dati in unico database centrale, mentre gli altri database sono solo "di passaggio" e privi di persistenza.
Anche se questo porta ad un rallentamento delle operazioni di comunicazione e computazione, abbiamo ritenuto fosse più semplice a livello implementativo. Inoltre volevamo evitare quanto più possibile la scrittura di file in memoria del dispositivo che ospita l'applicazione, per non renderla troppo corposa ed invadente.

Come ogni modello client-server, anche la nostra architettura è costituita da 3 "**Tier**":

- ★ **Presentazione**: Layer che si occupa di mostrare i risultati della computazione agli utenti che utilizzano il sistema, raccogliendo e gestendo gli input ricevuti da essi.
- ★ **Processamento**: Si occupa di offrire le funzionalità specifiche dell'applicazione
- ★ **Gestione** dati: Deve gestire la comunicazione con il db, attraverso il DBMS

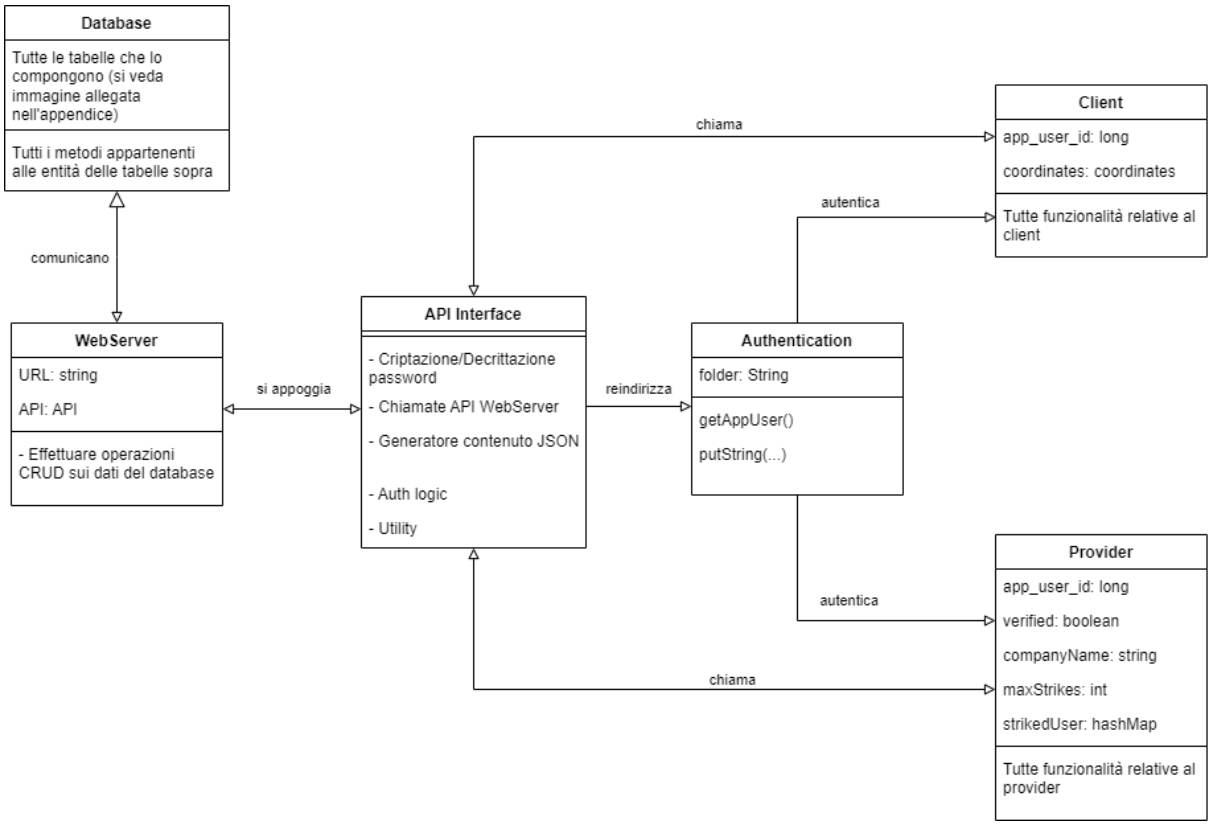
Modello controllo

Il modello di controllo che abbiamo scelto è di tipo **centralizzato**, in particolare il modello **call-return**.
Abbiamo scelto questo modello perché abbiamo un unico sottosistema che possiede il controllo globale e dà inizio e fine agli altri sottosistemi, visto che il nostro sistema è per buona parte di tipo **sequenziale**.
Verrà definita una gerarchia di procedure, in cui il controllo viene gestito **top-down**.



Modelli UML

Diagramma delle classi



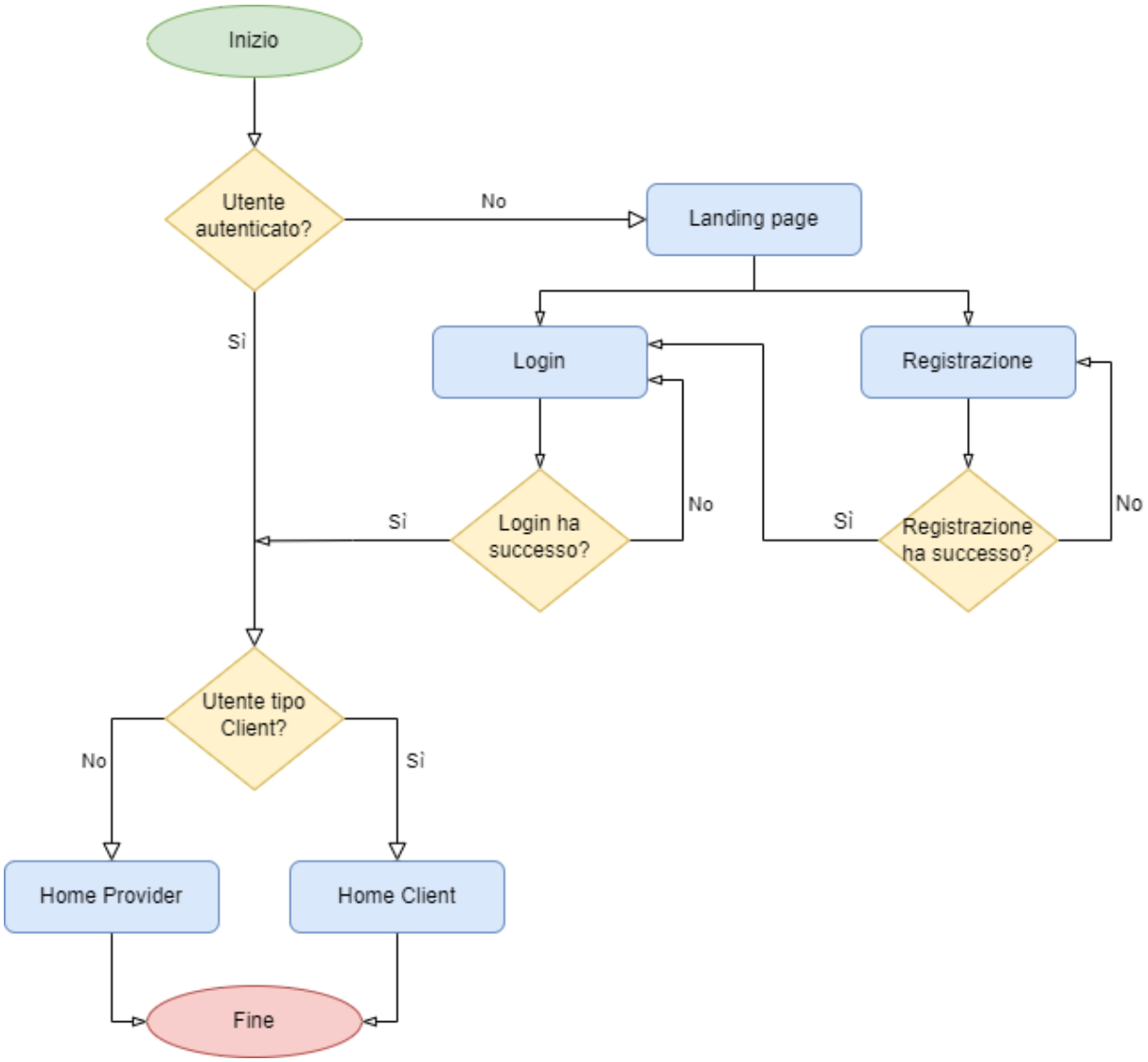
Note:

La classe API Interface non ha attributi effettivi, ma solo metodi.

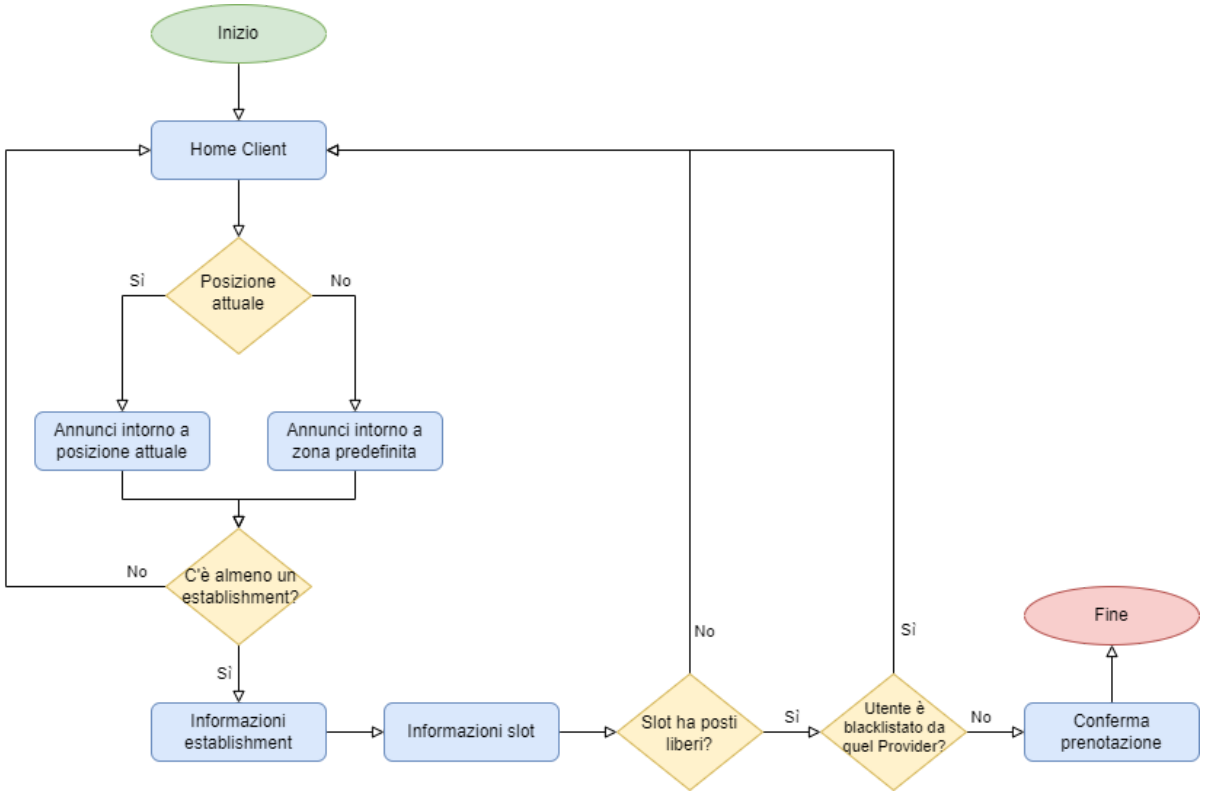
Diagrammi delle attività

Legenda

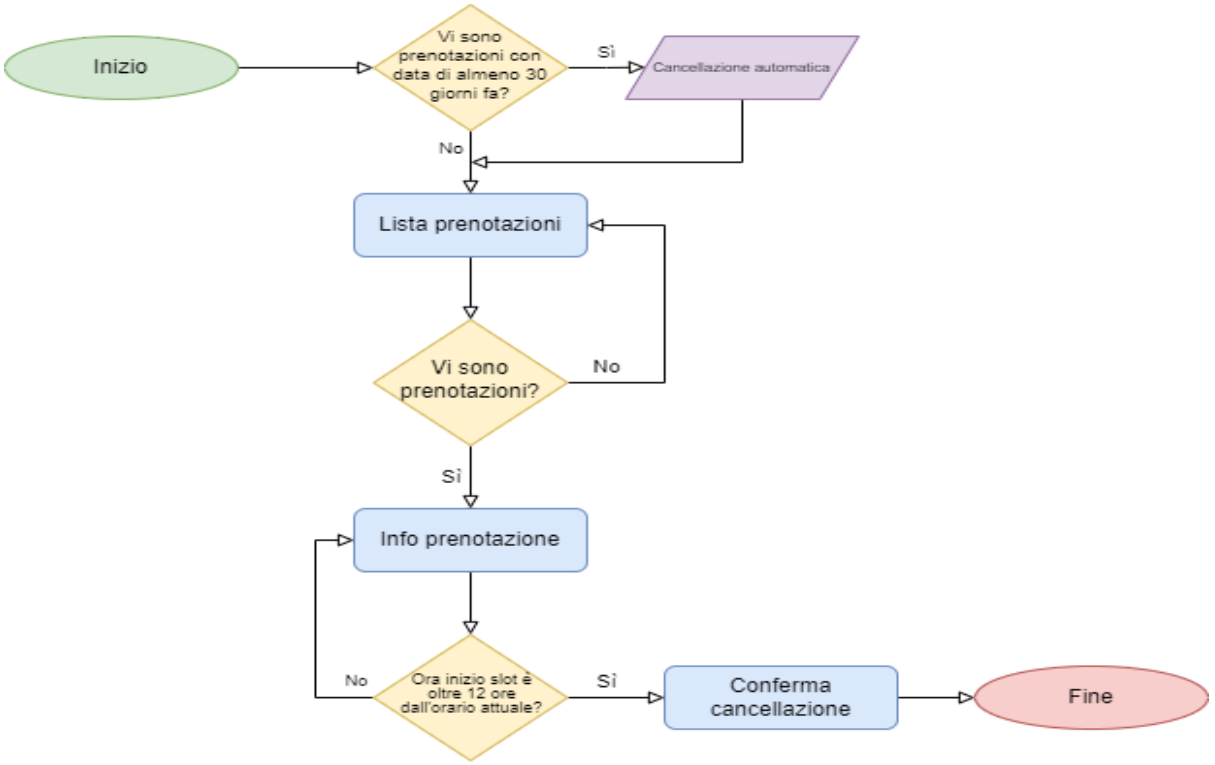
Auth



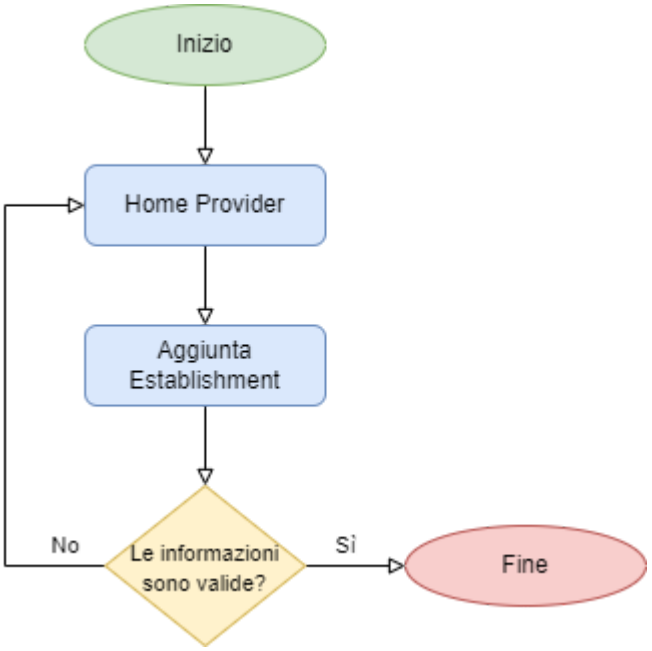
Eseguire prenotazione



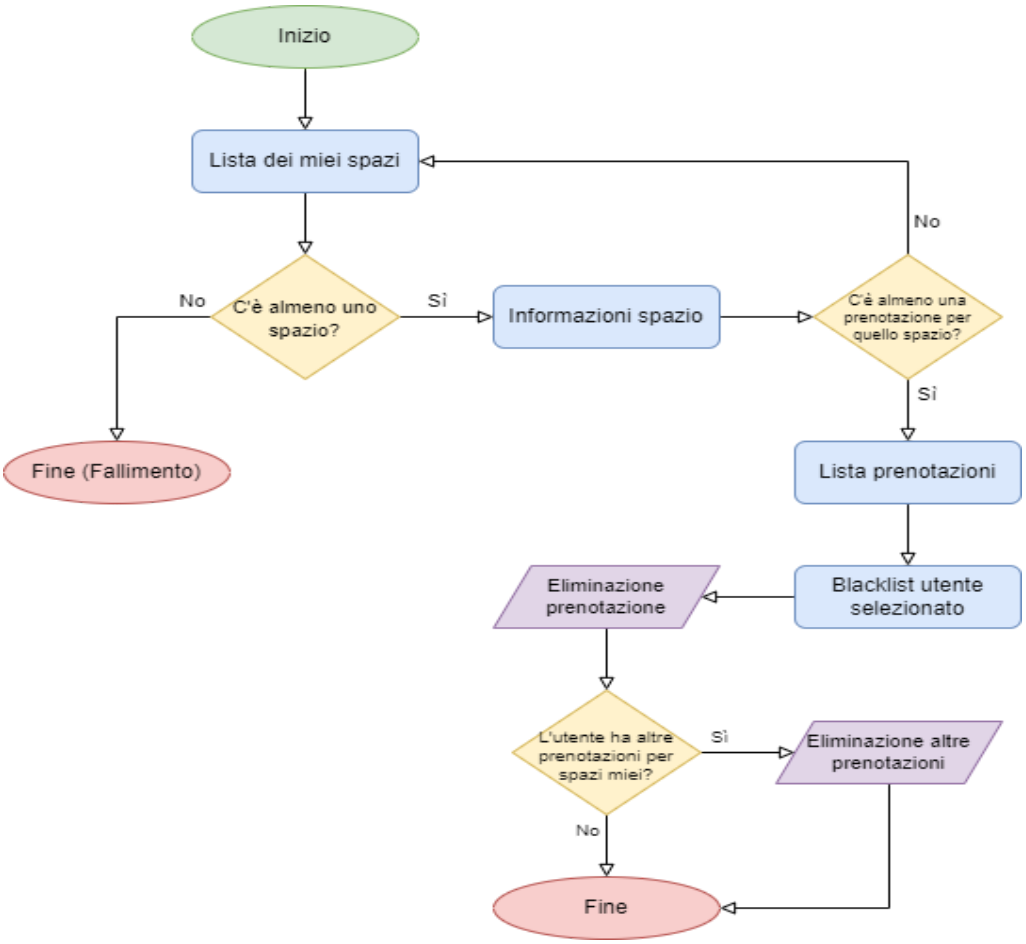
Cancellare prenotazione



Aggiunta establishment

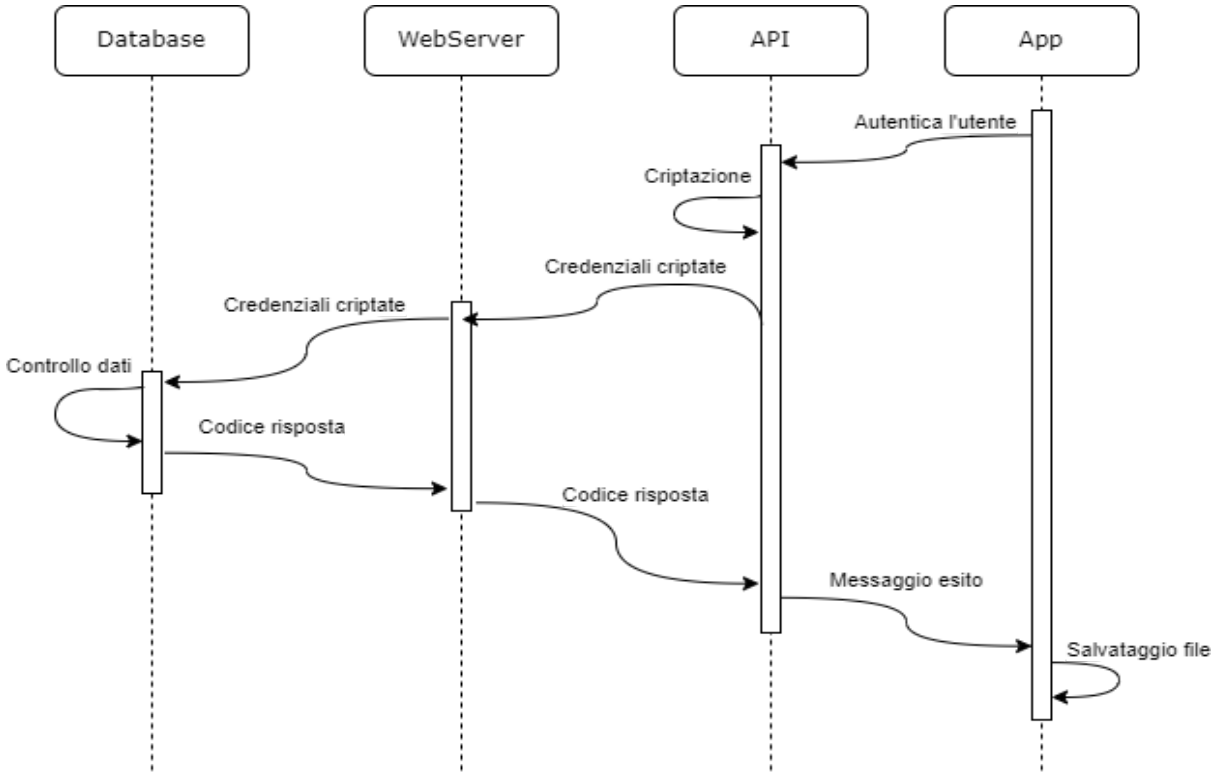


Blacklisting di un utente

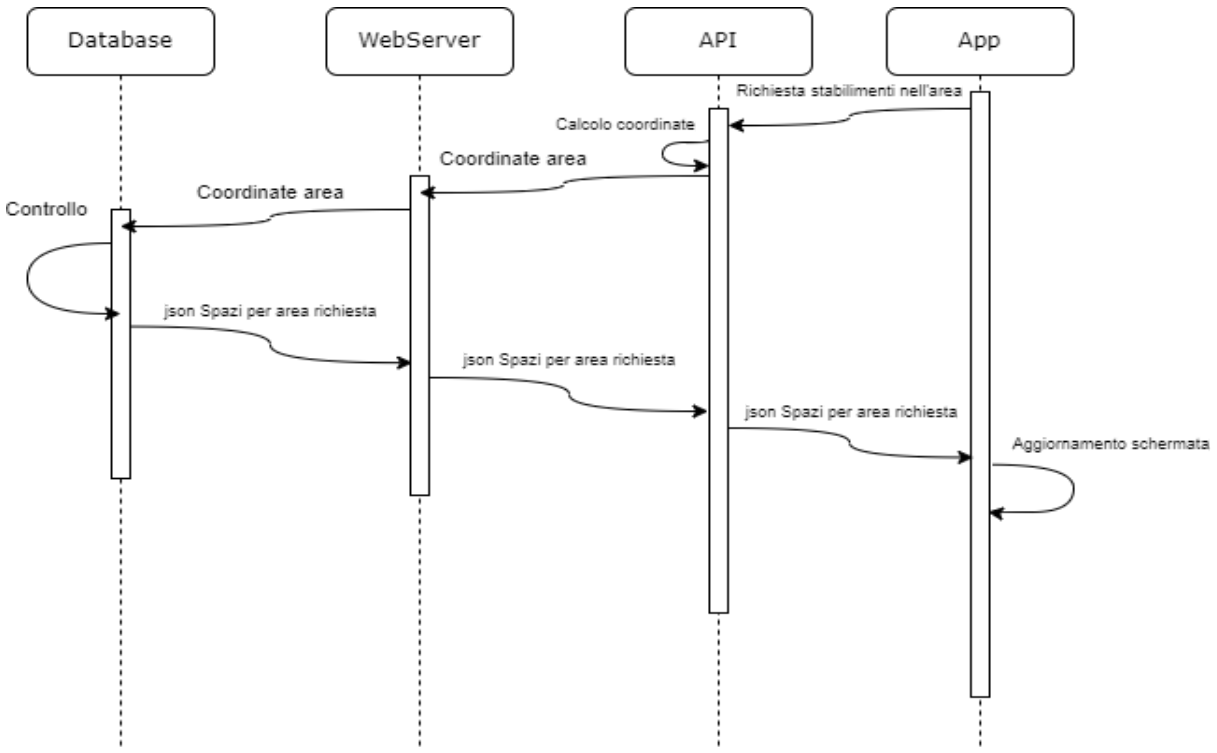


Diagrammi di sequenza

Auth



Lista spazi



Progettazione dell'interfaccia Utente

In questa sezione si illustrano le prime bozze di interfacce, realizzate in Android Studio, che comporranno la **UI** dell'utente e saranno alla base per definire anche la sua UX.
Le seguenti interfacce non è detto che corrispondano a quelle della versione di release!

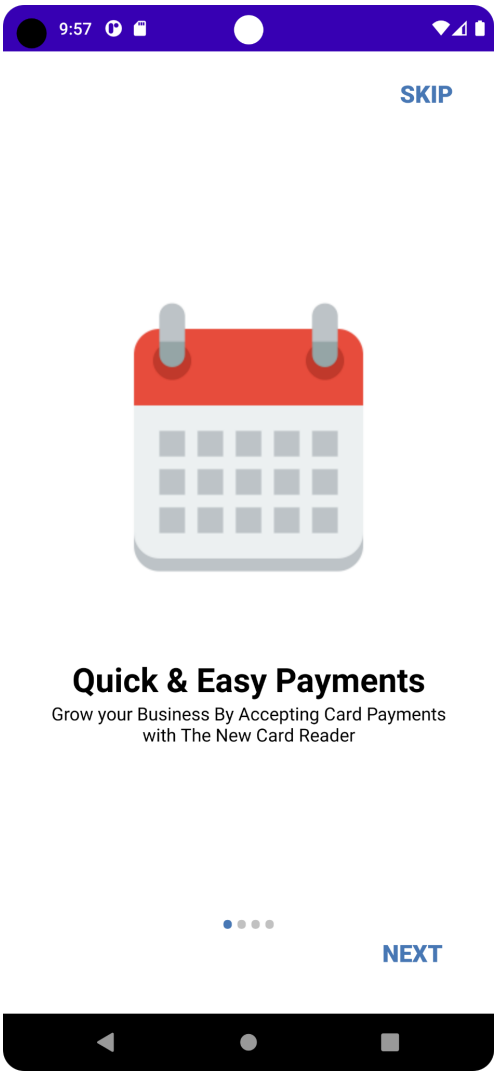
Piccola nota prima di iniziare, si può notare che, tranne nelle schermate sotto l'intestazione "Landing page", nella parte inferiore di **ogni schermata** via è una "**Navigation Bar**" a 3 voci, che facilita lo spostamento all'interno dell'applicazione da parte dell'utente.
Questa barra è presente in ogni singola schermata, sia per i Client che per i Provider, anche se le voci tra le due categorie differenzieranno leggermente.
In particolare le voci condivise sono "**Home**" e "**Profilo**", con il Client che ha come terza voce "**Le mie prenotazioni**", mentre il Provider "**I miei spazi**".

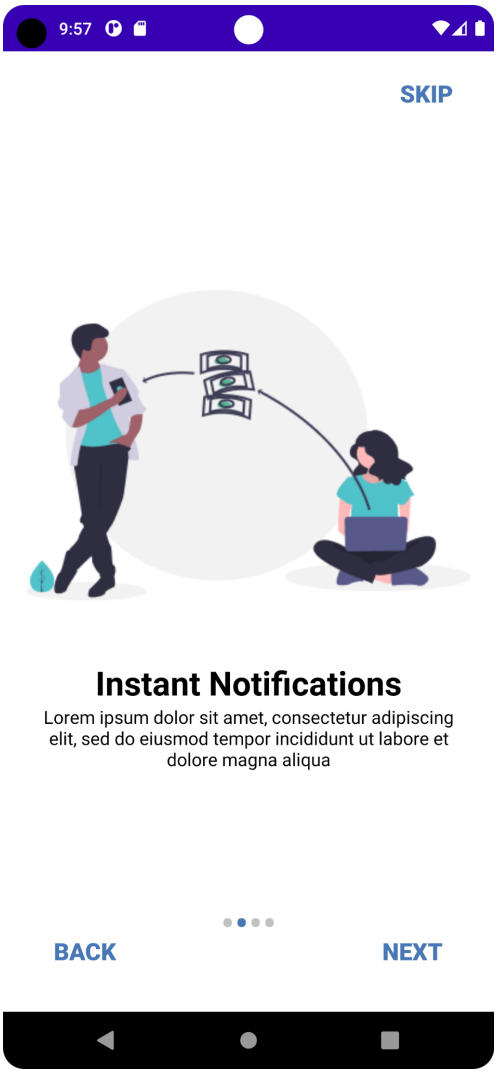
Alcune interfacce, in particolare "Le mie prenotazione", "I miei spazi" e "Aggiunta spazi" al momento non sono ancora state brevettate e verranno aggiunte nelle prossime versioni di questo documento.

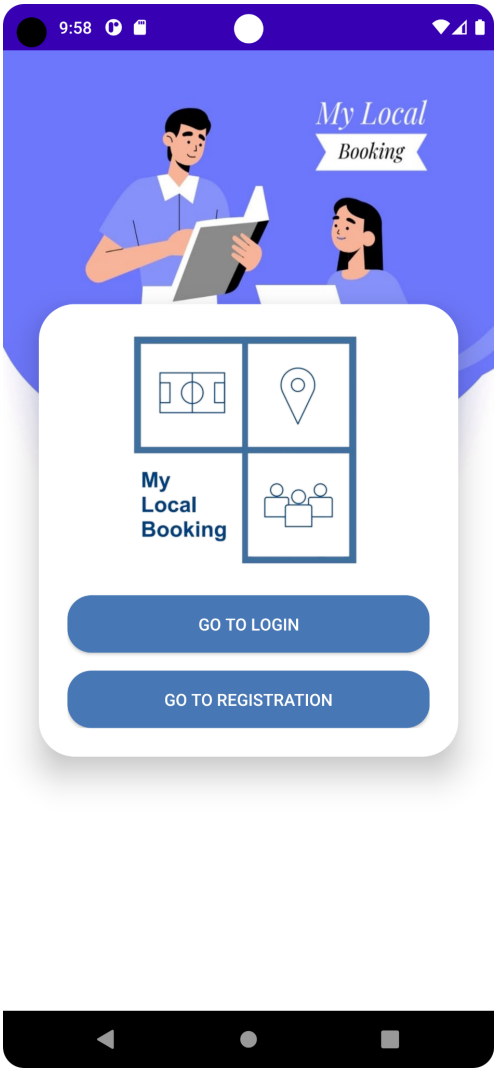
Nota: Le incongruenze di stile tra le varie interfacce verranno risolte in una versione successiva ed il presente documento verrà aggiornato a dovere. Si ricorda comunque che si tratta delle primissime bozze e non delle versioni finali!

Landing page

Questa schermata accoglie l'utente alla sua prima esperienza nell'app, o gli utenti che hanno effettuato il logout.
Ha lo scopo di informare gli utilizzatori delle funzionalità principali offerte dall'app.
Da questa pagina sono facilmente raggiungibili le schermate di **Login** e **Registrazione**.

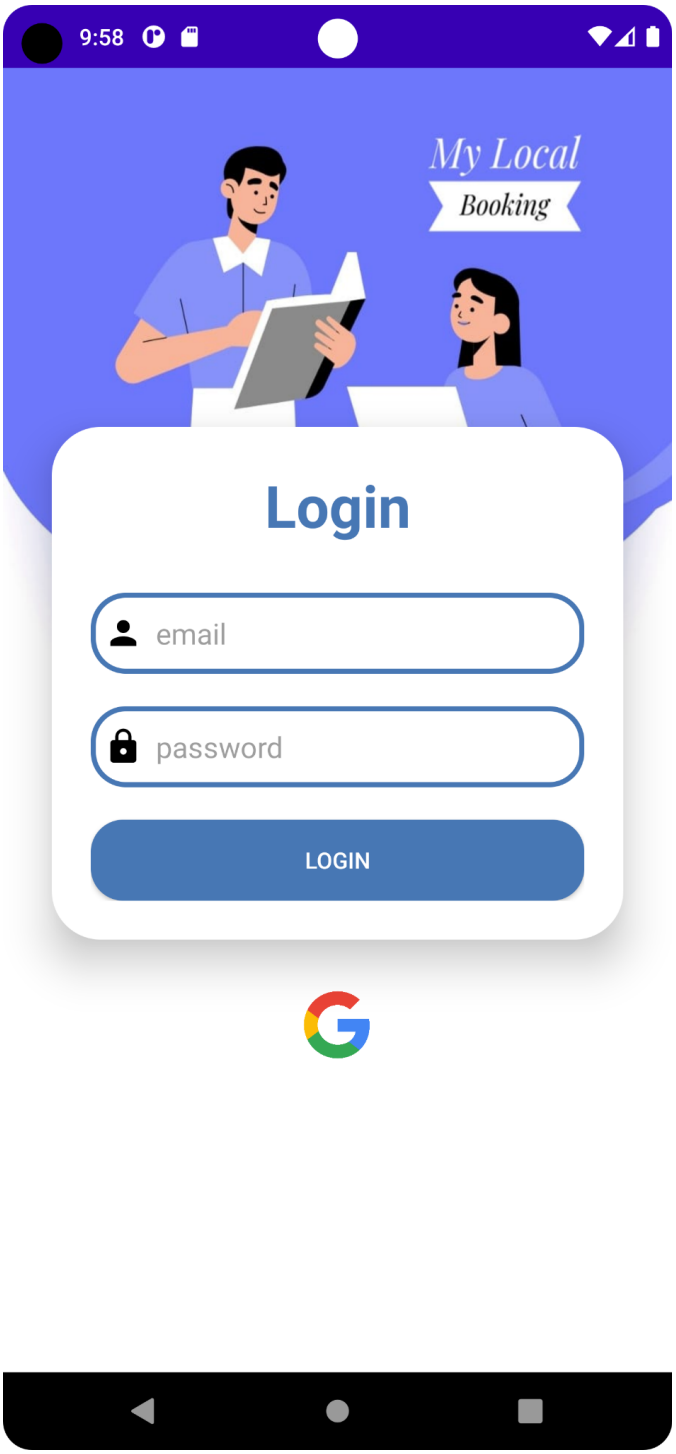






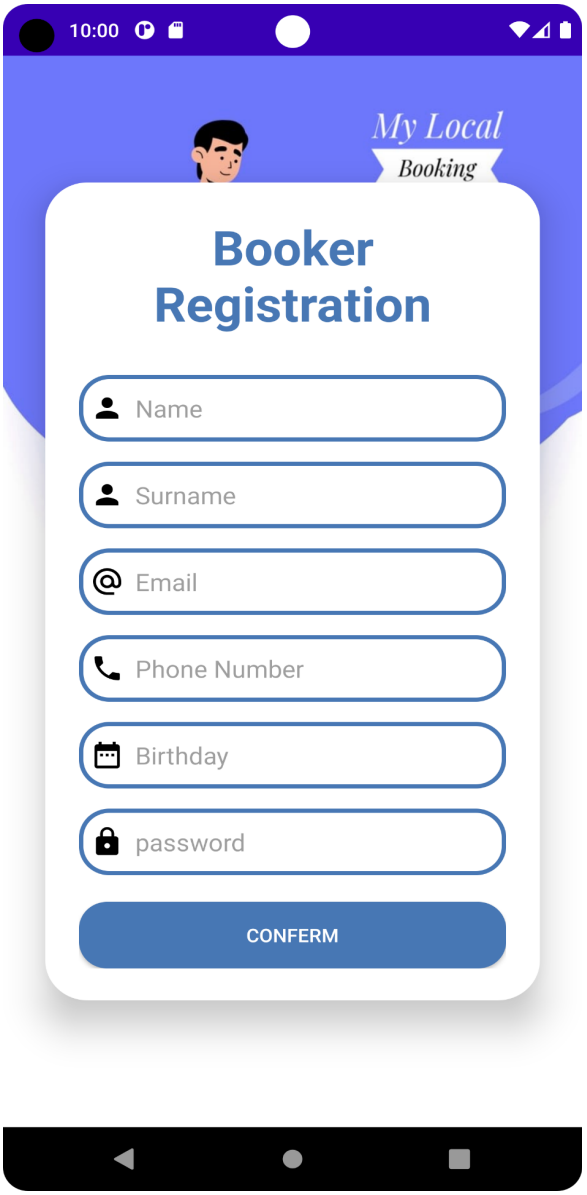
Login

Al momento del Login vengono chiesti il numero di cellulare e la password definiti dall'utente in fase di registrazione.



Registrazione

Nel form di registrazione invece, che differisce in base alla tipologia di utente, verranno sempre e comunque chiesti un numero di telefono, la password, nome e cognome, data di nascita, un eventuale email ed infine l'accettazione del contratto relativo alle condizioni d'uso e trattamento dei dati.
Poi per i Client anche la posizione che vogliono definire come "Predefinita", mentre per i Provider le informazioni in caso dovesse appartenere/rappresentare una società anziché essere un privato.



Client

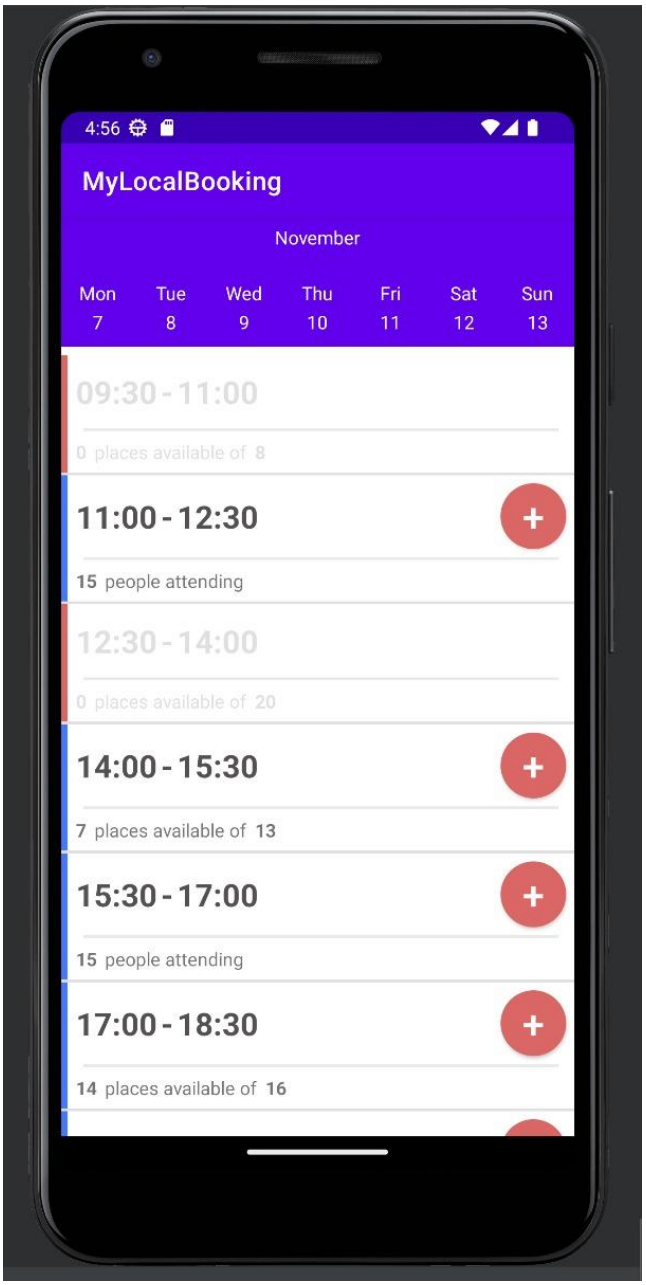
Questa sezione racchiude le interfacce principali che potrà visualizzare il Client

Home

La schermata qui presente consiste nella cosiddetta “Home” del Client. In essa è possibile osservare gli spazi messi a disposizione intorno alla zona predefinita, oppure è possibile tramite lo switch in alto, visualizzare gli spazi intorno alla posizione attuale dell’utente, previo accesso alle funzionalità di geolocalizzazione.

Eseguire Prenotazione

In quest'altra schermata invece vengono mostrati i dati e le possibilità per l'utente nel momento che desidera eseguire una prenotazione presso uno spazio.
Nella schermata si possono selezionare gli slot orari ed i giorni in cui effettuare la prenotazione, con le relative disponibilità.

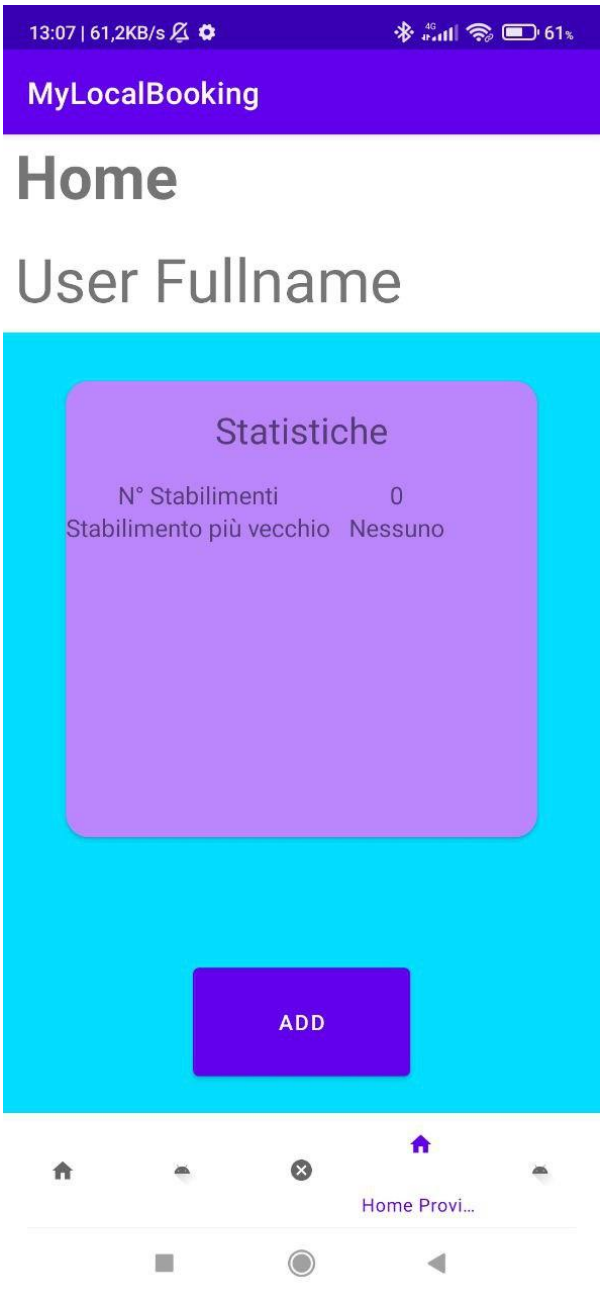


Provider

Questa sezione racchiude le interfacce principali che potrà visualizzare il Provider

Home

La schermata qui presente consiste nella cosiddetta “Home” del Provider. In essa è possibile osservare alcune statistiche relative agli spazi messi a disposizione.

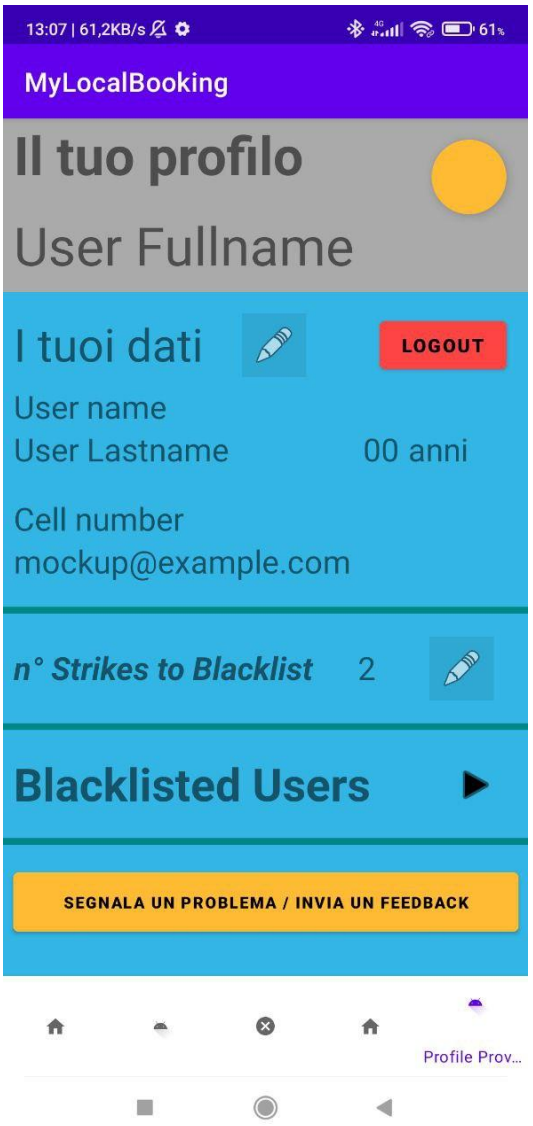


Condivise

In questa sezione vi sono le schermate “condivise” da entrambe le tipologie di utenti, in particolare vi sono solo le componenti relative al profilo dell’utente.

Profilo

Nonostante vi siano delle leggere differenze tra quello Client e quello Provider, le funzionalità comuni, come la modifica dei propri dati e password, oppure la possibilità di segnalare un problema, sono entrambe disponibili ad entrambe le tipologie.



Dalle foto è possibile notare le differenze tra i due, in particolare:

- **Client** ha la possibilità di modificare la propria posizione predefinita su cui basare gli annunci
- **Provider** invece può impostare il numero massimo di strike, prima che un utente venga blacklistato; inoltre può accedere ad un'altra schermata dove vengono mostrati tutti gli utenti blacklistati da lui (vedi eventuale schermata Blacklist sopra)