Documento di Progettazione

Smart Meal

Gruppo: Quadcore

Facci Matteo 875377
Guidobene Davide 877150
Pittini Enrico 877345
Rosin Giacomo 875724

Versione 1.0

12/12/2020

Indice

1.Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è descrivere e spiegare le strategie di progettazione e le scelte implementative principali riguardanti l'architettura di sistema dell'applicazione Smart Meal.
Si procederà con un approccio top-down, passando per il modello di dati e del controllo, partendo dall'architettura di sistema, scendendo fino alla presentazione dei modelli UML.
Infine si darà una descrizione dettagliata dell'interfaccia e delle schermate dell'applicazione.

1.2 Struttura del documento

- **Glossario**: vengono esplicati i termini specifici del documento
- **Architettura del sistema**: viene spiegato come è stata strutturata e pensata l'applicazione, specificando sottosistemi e scelte di design.
- **Modello dei dati e del controllo**: viene presentato il modello con cui avviene la comunicazione e lo scambio dei dati tra i sottosistemi e il modello che descrive il flusso di controllo dell'applicazione
- **Modelli UML: diagrammi delle classi**. Vengono rappresentate le classi principali dell'applicazione, mostrando per ciascuna i campi e i metodi e mostrando le relazioni tra esse.
- **Modelli UML: diagrammi di sequenza**. Viene rappresentata, tramite sequenza temporale, la logica interna di ciascun caso d'uso descritto nel documento di specifica dei requisiti.
- **Progettazione dell'interfaccia utente**: vengono rappresentate le interfacce e schermate principali dell'applicazione

2. Glossario

- **Aggregazione**: tipo di relazione tra classi, in cui una classe è una componente di un'altra.
- **Ereditarietà**: tipo di relazione tra classi, in cui una classe è un sottotipo di un'altra e ne eredita campi e metodi.
- **Floating button**: bottone tipico dell'interfaccia Android, che appare come sospeso. E' generalmente situato in basso a destra.
- **Tipo enumerativo**: tipo che può assumere solo un numero ristretto di valori.
- **Shake**: azione che viene eseguita scuotendo il dispositivo.
- **Nearby**: API forniti da Google che permettono la connessione e lo scambio di messaggi tra due dispositivi android.

3. Architettura del Sistema

3.1 Sottosistemi

La struttura globale dell'applicazione è composta dai seguenti sottosistemi:

- Gestione comunicazione
- Gestione locale (tavoli e notifiche)
- Gestione sensori (rilevazione shake, localizzazione)
- Gestione memoria (nome, descrizione locale, modalità applicazione, …)
- Gestione autenticazione

Gestione comunicazione

Sistema che si occupa di gestire la trasmissione di messaggi tra lato gestore e lato cliente sfruttando il servizio di Nearby.
La sua implementazione è basata sulle 2 classi che si occupano di astrarre la comunicazione: CustomerCommunication da lato cliente e ManagerCommunication da lato gestore; fornendo così un'API comoda ed elegante.
La comunicazione tra le 2 classi avviene tramite lo scambio di una struttura dati Message apposita che permette di inviare i dati e distinguere agilmente il tipo di richiesta.

Gestione locale

Sistema che si occupa di astrarre la struttura del locale.
Nella sua implementazione la gestione dei tavoli e delle chiamate verso i camerieri sono delegate a due strutture interne: rispettivamente TableHandler e WaiterNotificationHandler.

Gestione sensori

Sistema che si occupa di rilevare e riconoscere dati necessari al funzionamento dell'applicazione attraverso i sensori, quali l'accelerometro per la rilevazione di shake e il GPS per la localizzazione del dispositivo.

Gestione memoria

Sistema che si occupa di astrarre la memoria usata dall'applicazione fornendo un'interfaccia semplice e ben tipata basandosi sull'uso delle classi CustomerStorage da lato cliente e ManagerStorage da lato gestore.

Gestione autenticazione

Sistema che si occupa di verificare l'identità dell'utente e proteggere l'accesso ad aree private dell'applicazione quali quelle riservate al gestore.

3.2 Strategia di progettazione

Per lo sviluppo dell'applicazione abbiamo deciso di adottare una strategia di progettazione basata sul modello ad oggetti in quanto abbiamo valutato che mettesse più in risalto le qualità dei membri del nostro team, più abituati all'uso del paradigma a oggetti, e allo stesso tempo semplificasse la programmazione in Java, nato come linguaggio a oggetti, e l'interazione con Android.

3.3 Scelte di design

Nel design della nostra applicazioni ci siamo ritrovati a dover compiere alcune scelte che sono state una componente importante del processo di coesione, disaccoppiamento, comprensibilità e adattabilità del codice.

Rilevamento ingresso

Per il rilevamento dell'ingresso dell'utente abbiamo deciso di usare la localizzazione tramite GPS in quanto le API fornite da Android permettono un'implementazione della nostra idea più chiara rispetto alle API dell'alternativa Nearby che sarebbe comunque molto dispendiosa anche da un punto di vista del consumo energetico, punto particolarmente importante per applicazioni per dispositivi mobili dove la durata della batteria è più limitata.

Codifica dei dati trasmessi con Nearby

Abbiamo deciso di codificare i dati trasmessi tramite Nearby nel sistema dedicato alla gestione della comunicazione sfruttando l'interfaccia Serializable fornita da Java dal momento che esistono metodi già implementati per la trasformazione di oggetti Serializable in byte (tipo primitivo usato da Nearby) e viceversa da byte a oggetti Serializable.
Inoltre Java offre già alcune classi e in particolare alcune Collection che implementano questa interfaccia.
L'alternativa sarebbe stata implementare noi questi metodi fornendo una nostra codifica ad hoc che avrebbe potuto darci una maggiore flessibilità ma che sarebbe stato meno safe e meno nello spirito della programmazione ad oggetti.

Implementazione classe Local

L'implementazione di Local prevede l'uso di 2 classi interne per la gestione delle chiamate ai camerieri e per la gestione dei tavoli, rispettivamente WaiterNotificationHandler TableHandler. Abbiamo optato per questa scelta di design perché nella logica della programmazione a oggetti ha senso disporre di una classe ad hoc per astrarre ogni sotto-funzionalità di Local.

Classe Table

Per i tavoli abbiamo deciso di costruire 2 classi: oltre a una classe Table generica, abbiamo implementato una sottoclasse ManagerTable usata solo dal lato Manager permettendoci così di tenere riservate al gestore eventuali informazioni che non si vogliono mostrare ai suoi clienti.

4. Modello dei Dati e del Controllo

Il nostro team ha ritenuto che il modello dei dati che si adattasse in modo più efficace alle necessità della nostra applicazione fosse il modello Client-Server.
Il dispositivo del gestore funge da server cui i dispositivi dei clienti possono connettersi per disporre di funzionalità come la scelta del tavolo o la chiamata del cameriere.
L'applicazione del gestore e quelle dei clienti, durante la connessione, si scambiano dati utili al funzionamento dell'app. In particolare l'applicazione del gestore si occupa di memorizzare i dati inviati dai clienti, questi dati non sono permanenti, ma sono tenuti memorizzati solo per un periodo di tempo strettamente necessario.
Al contrario, le applicazioni dei clienti tengono memorizzati i propri dati in modo persistente.

Per quanto riguarda il modello di controllo abbiamo deciso di adottare il modello basato su eventi in quanto il più naturale per lo sviluppo di un'app in Android dove sono richiesti diversi input da parte degli utenti e dove le callback sono ormai uno standard.
In particolare utilizziamo il modello broadcast selettivo dove i sottosistemi rendono noto di essere interessati ad un determinato evento in modo che solo i sottosistemi interessati vengano avvertiti nel momento in cui tale evento effettivamente avviene.

5. Modelli UML: diagrammi delle classi

Nelle seguenti pagine sono rappresentati i diagrammi UML delle classi. Per snellire la visualizzazione, prima di tutto sono presentate le varie classi singolarmente, specificando per ciascuna i campi e i metodi. In seguito sono presentate le relazioni di aggregazione e di ereditarietà tra le classi.

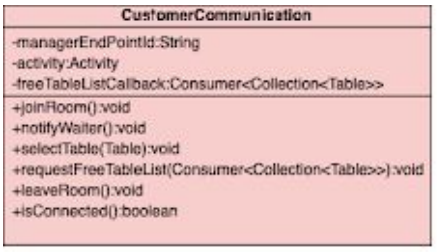
Nei diagrammi non sono rappresentati i tipi enumerativi (Enum). Ne si da ora una breve lista.

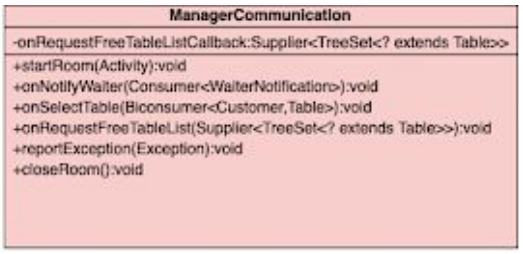
- `RequestType {FREE_TABLE_LIST, WAITER_NOTIFICATION, TABLE_SELECTION}`
- `FoodCategory {STARTERS, MAIN_COURSES, SIDE_DISHES, SALADS, DESSERTS, DRINKS}`
- `ApplicationMode {CUSTOMER, MANAGER, UNDEFINED}`

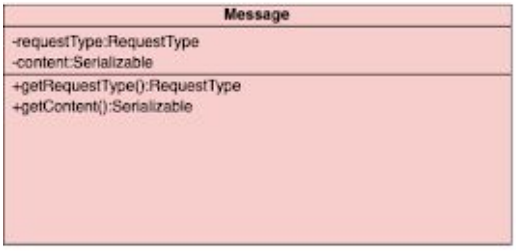
Si da inoltre una breve lista delle classi che nei diagrammi sono citate e che sono di libreria.

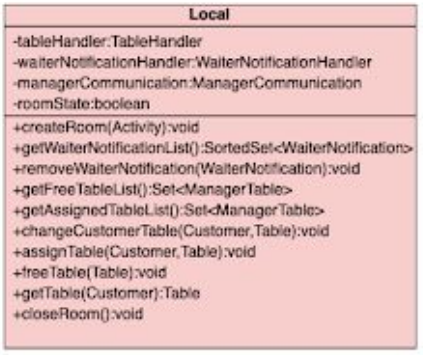
- `Activity (android.app.Activity)`
- `Serializable (java.io.Serializable)`
- `Location (android.location.Location)`
- `Image (android.media.Image)`
- `Collection (java.util.Collection)`
- `Consumer (java.util.function.Consumer)`
- `BiConsumer (java.util.function.BiConsumer)`
- `Supplier (java.util.function.Supplier)`
- `Runnable (java.lang.Runnable)`
- `Map (java.util.Map)`
- `Set (java.util.Set)`
- `SortedSet (java.util.SortedSet)`
- `TreeSet (java.util.TreeSet)`
- `Date (java.util.Date)`

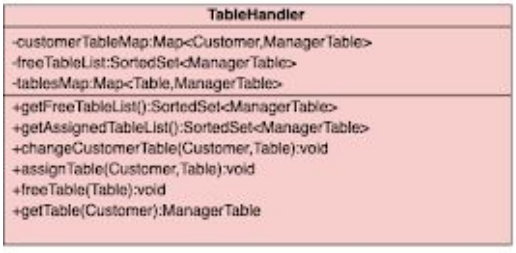
5.1 Diagramma UML delle classi

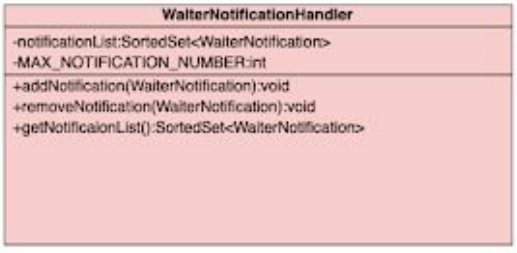


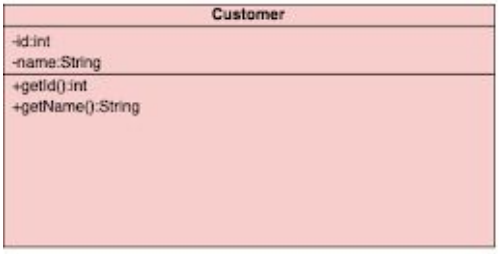


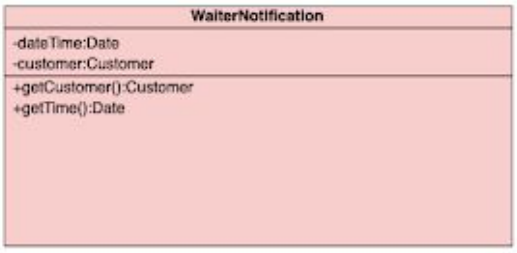


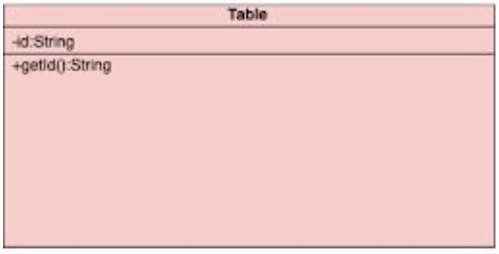


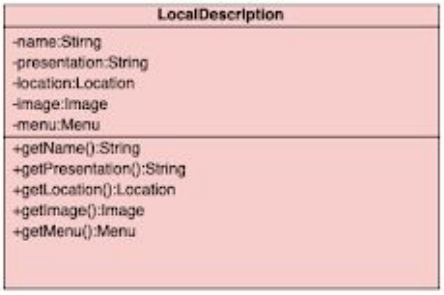




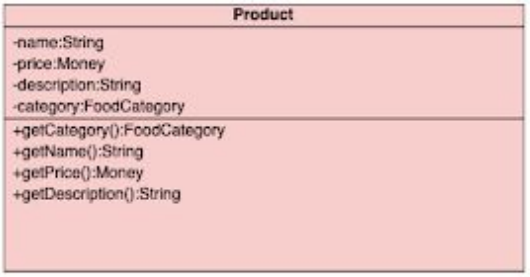




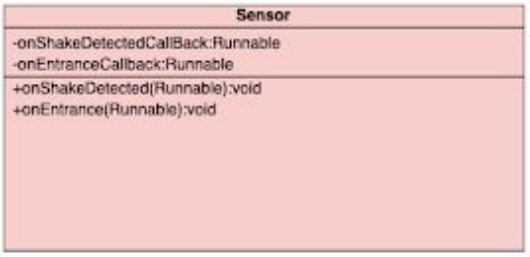


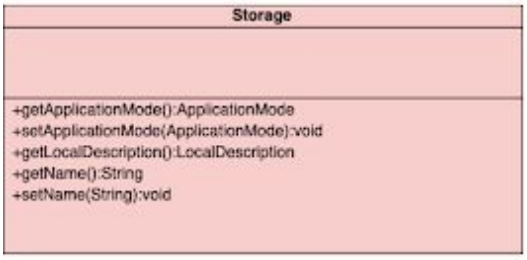


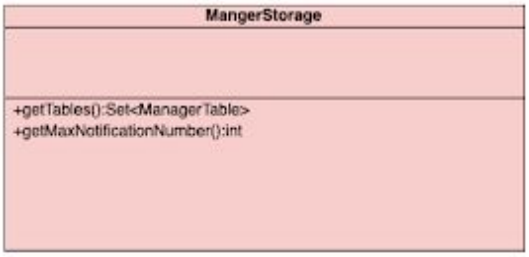

Menu
-categoriesProductsMap:Map<FoodCategory,Set<Product>>
+getCategoriesProductsMap():Map<FoodCategory,Product>
+getProducts():Set<Product>


Product
-name:String
-price:Money
-description:String
-category:FoodCategory
+getCategory():FoodCategory
+getName():String
+getPrice():Money
+getDescription():String


Money
-value:int
+getValue():int


Sensor
-onShakeDetectedCallBack:Runnable
-onEntranceCallback:Runnable
+onShakeDetected(Runnable):void
+onEntrance(Runnable):void

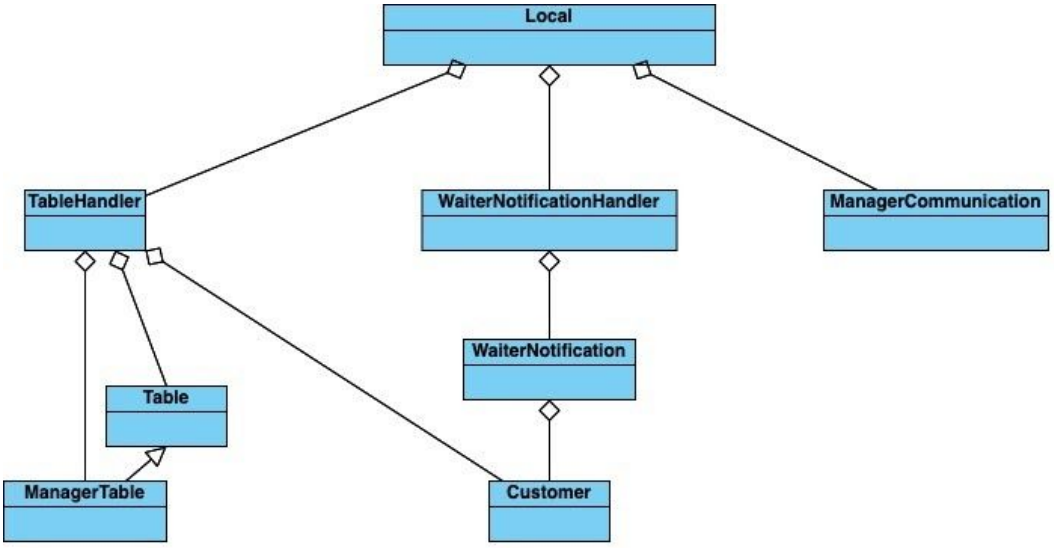

Storage
+getApplicationMode():ApplicationMode
+setApplicationMode(ApplicationMode):void
+getLocalDescription():LocalDescription
+getName():String
+setName(String):void

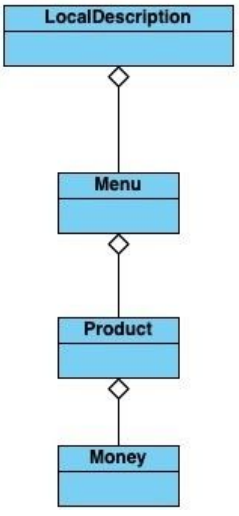

MangerStorage
+getTables():Set<ManagerTable>
+getMaxNotificationNumber():int

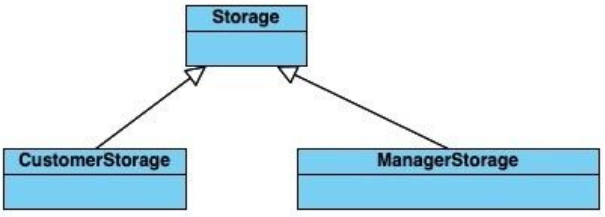

Authentication
+checkPin(String):boolean

5.2 Diagramma di relazioni tra le classi

Nel seguente diagramma vengono evidenziate le effettive relazioni tra le classi del progetto. Non vengono mostrate tutte le classi ma solamente le classi che sono in relazione con altre.
Ci sono due tipi di relazioni: aggregazione ed ereditarietà.
In particolare, usiamo il “diamante” per l’aggregazione e la freccia vuota per indicare l’ereditarietà.

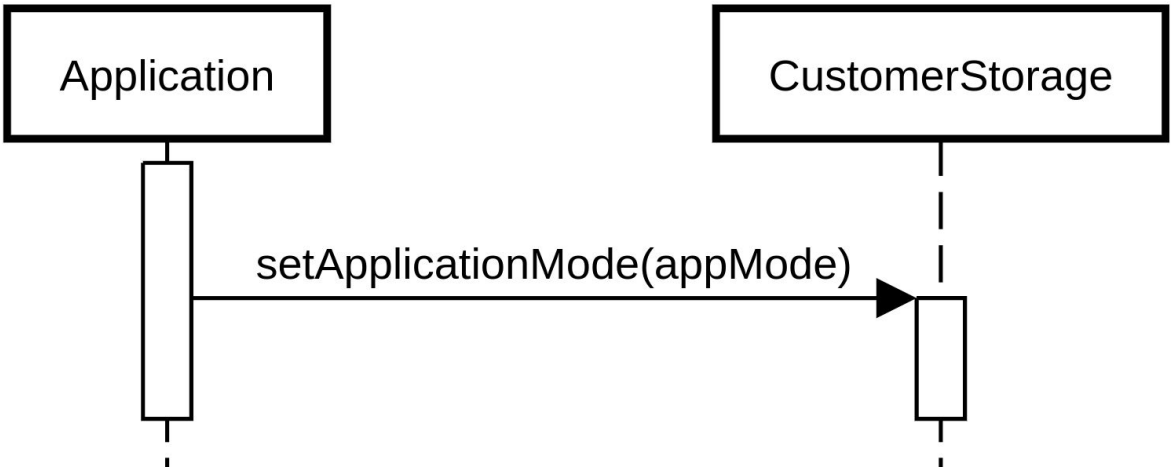




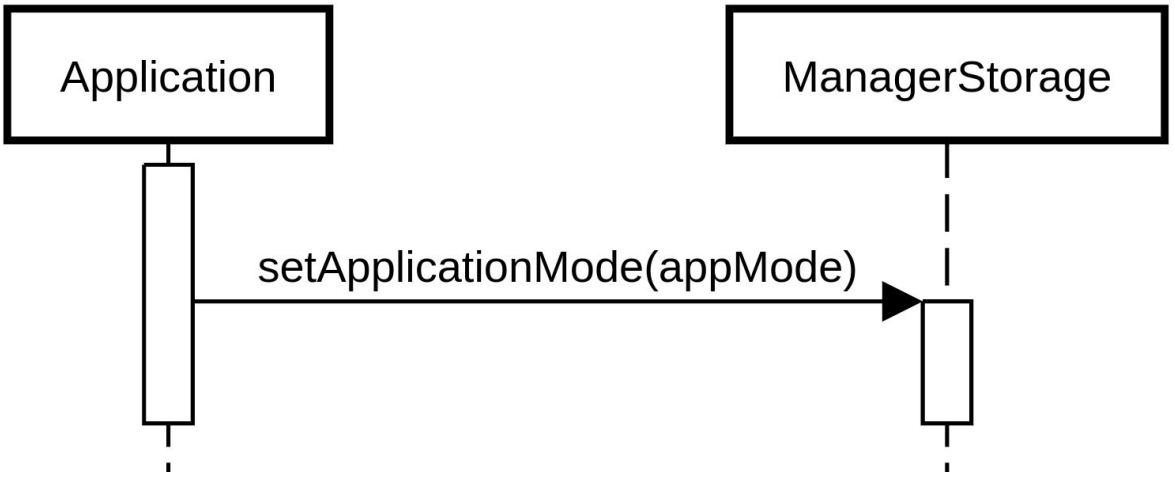


6. Modelli UML: diagrammi di sequenza

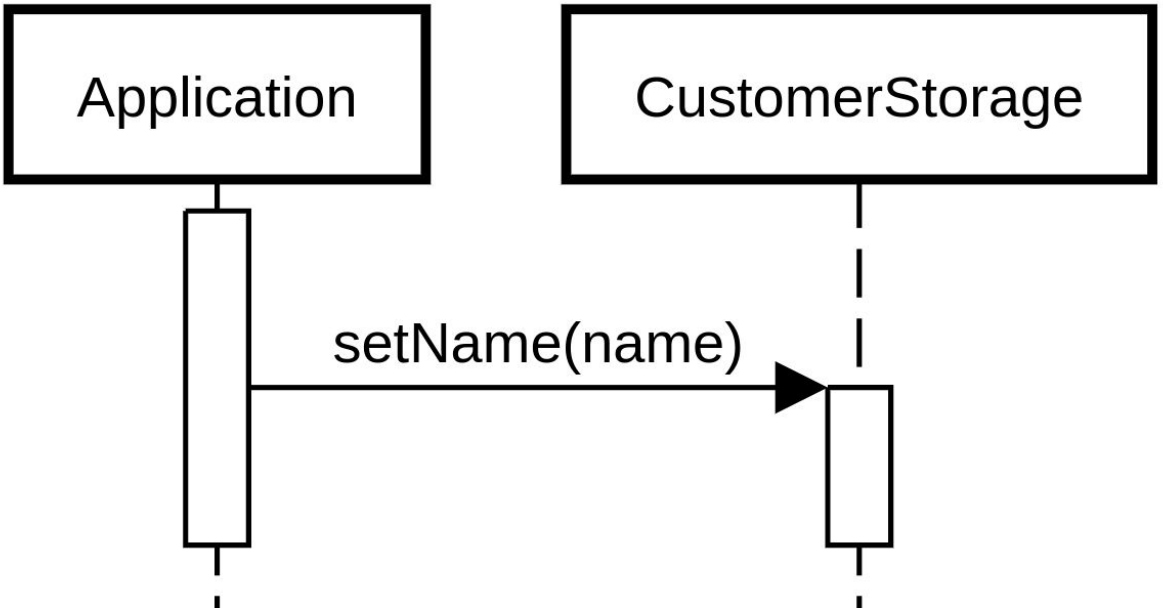
CD-01: Scegli cliente



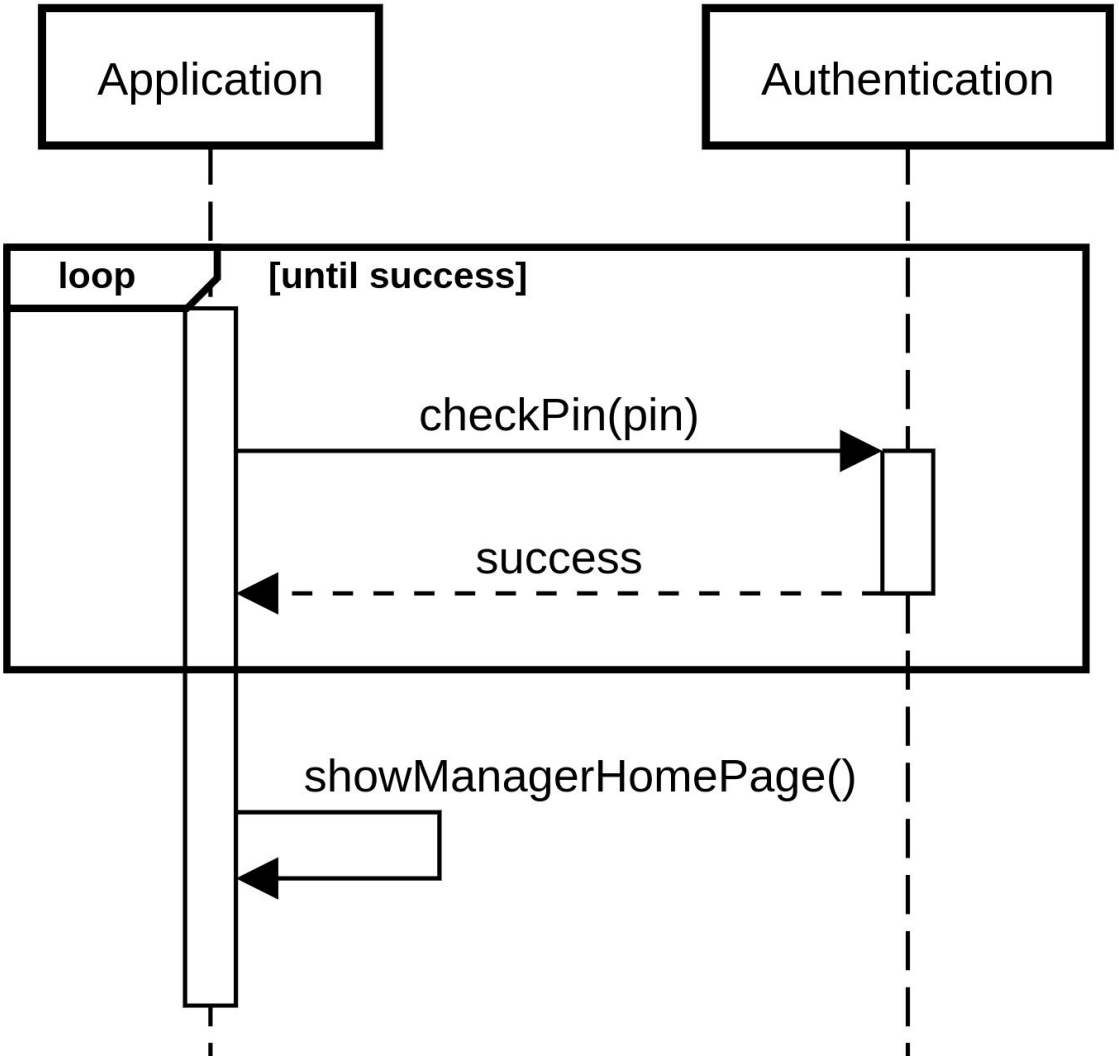
CD-02: Scegli gestore



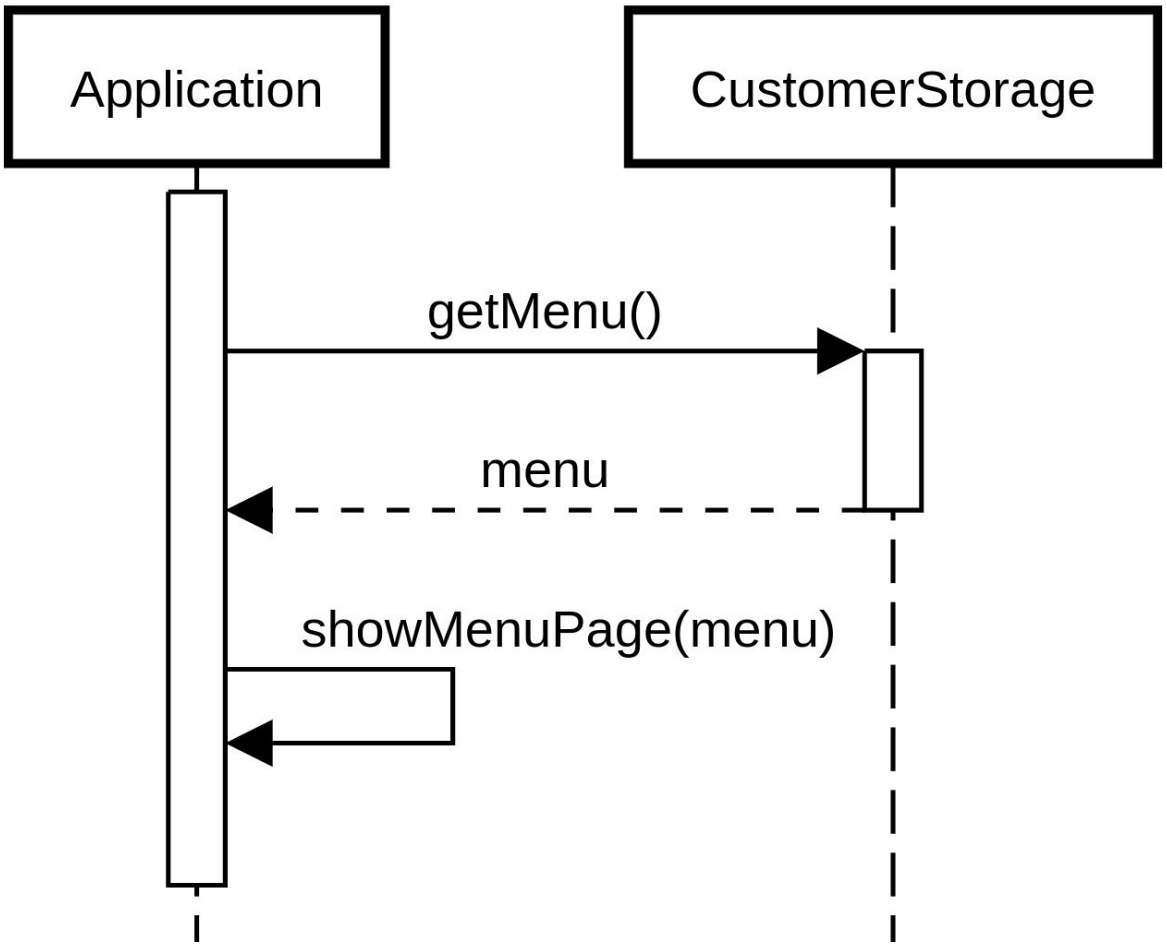
CD-03: Inserimento nome



CD-04: Inserimento pin

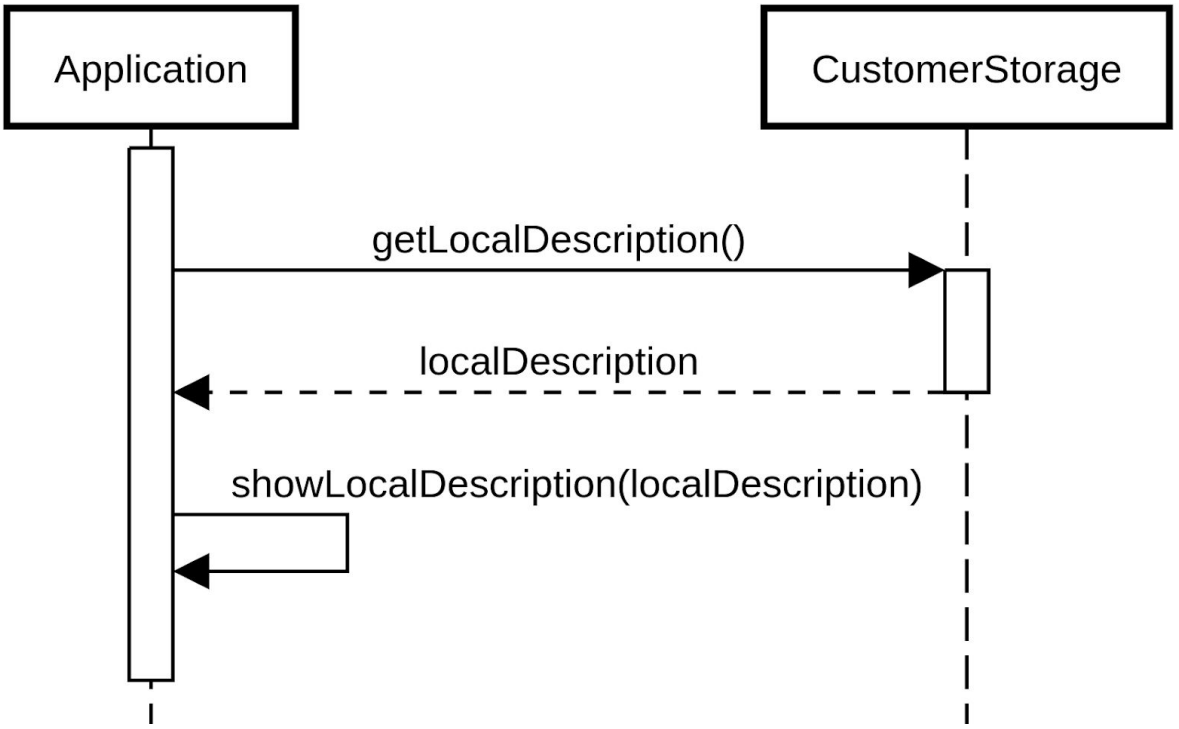


CD-05: Visualizza menù



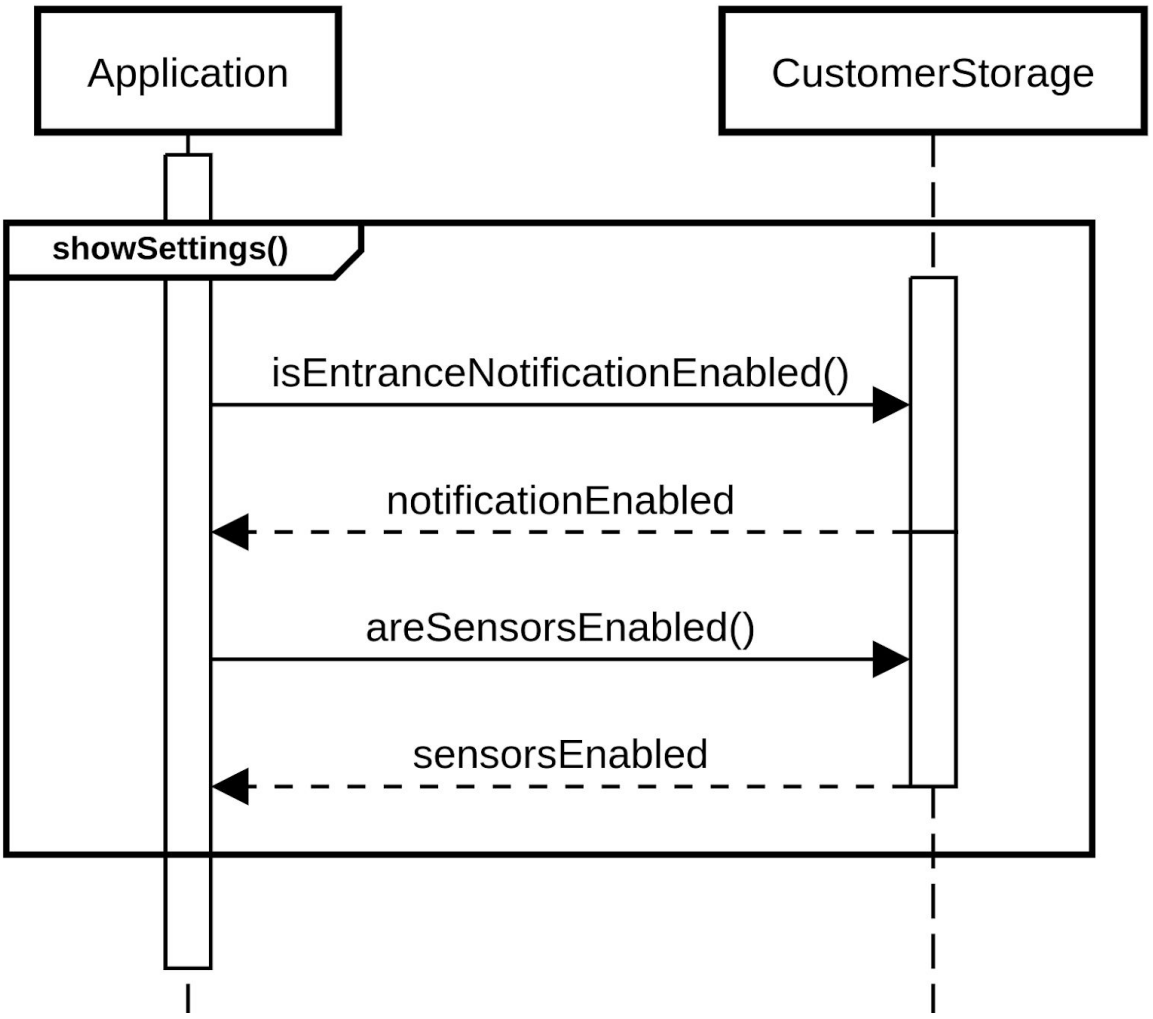
Classi usate ma non specificate: `LocalDescription`, `Menu`

CD-06: Visualizza descrizione locale

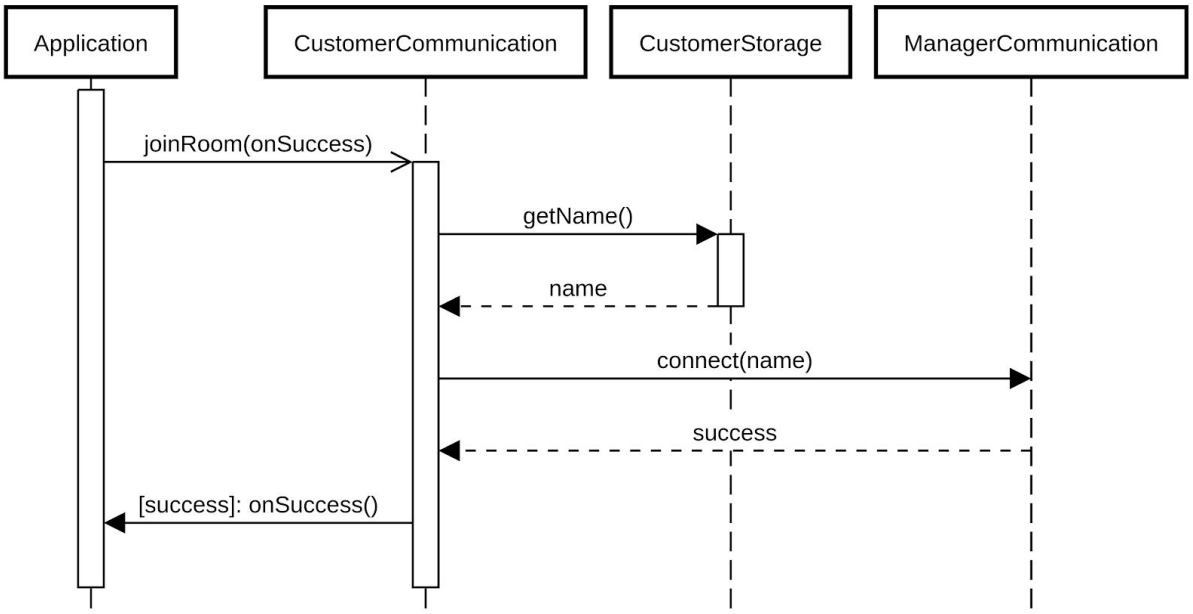


Classi usate ma non specificate: `LocalDescription`

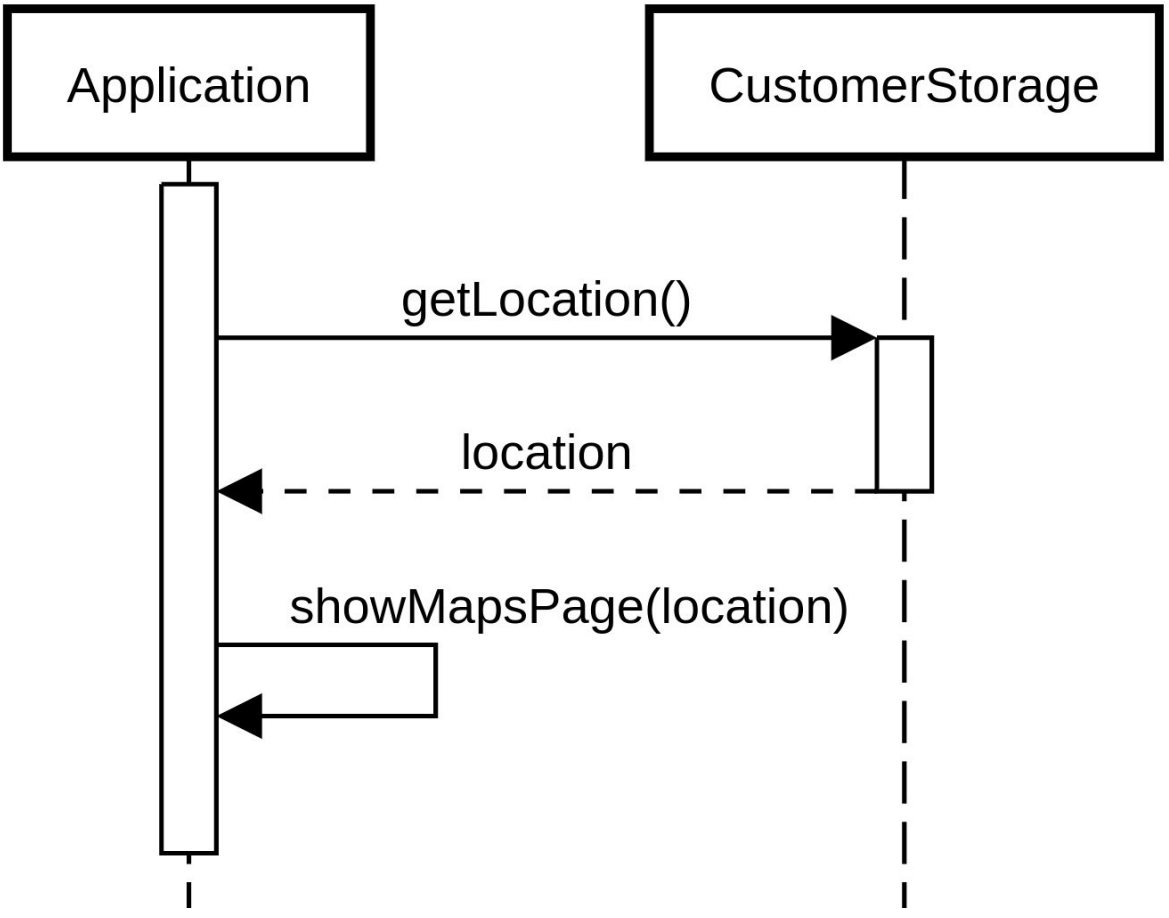
CD-07: Impostazioni



CD-08: Accedi alla stanza virtuale

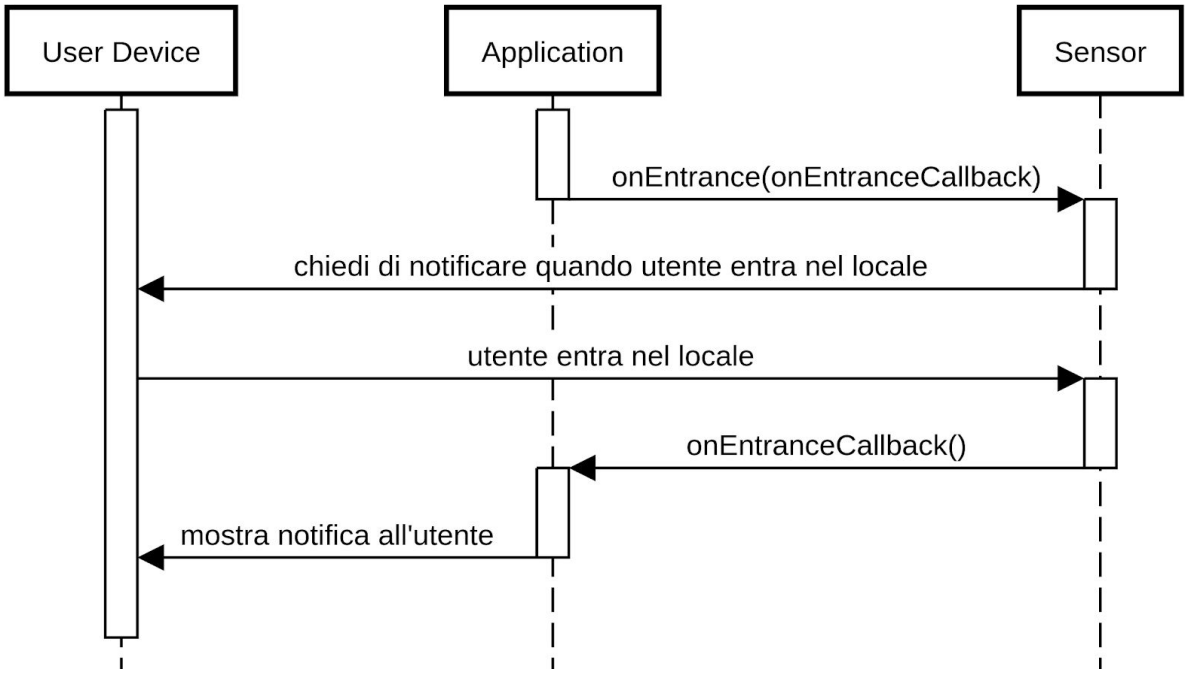


CD-09: Visualizza mappa

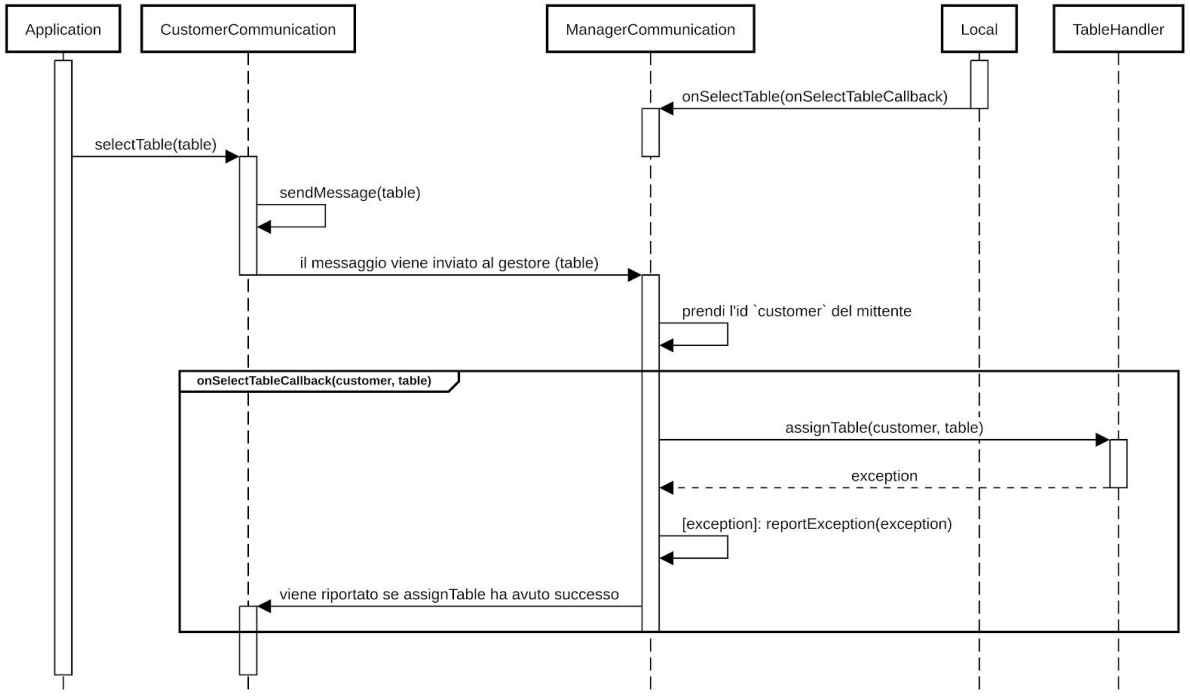


Classi usate ma non specificate: `LocalDescription`

CD-10: Notifica di benvenuto

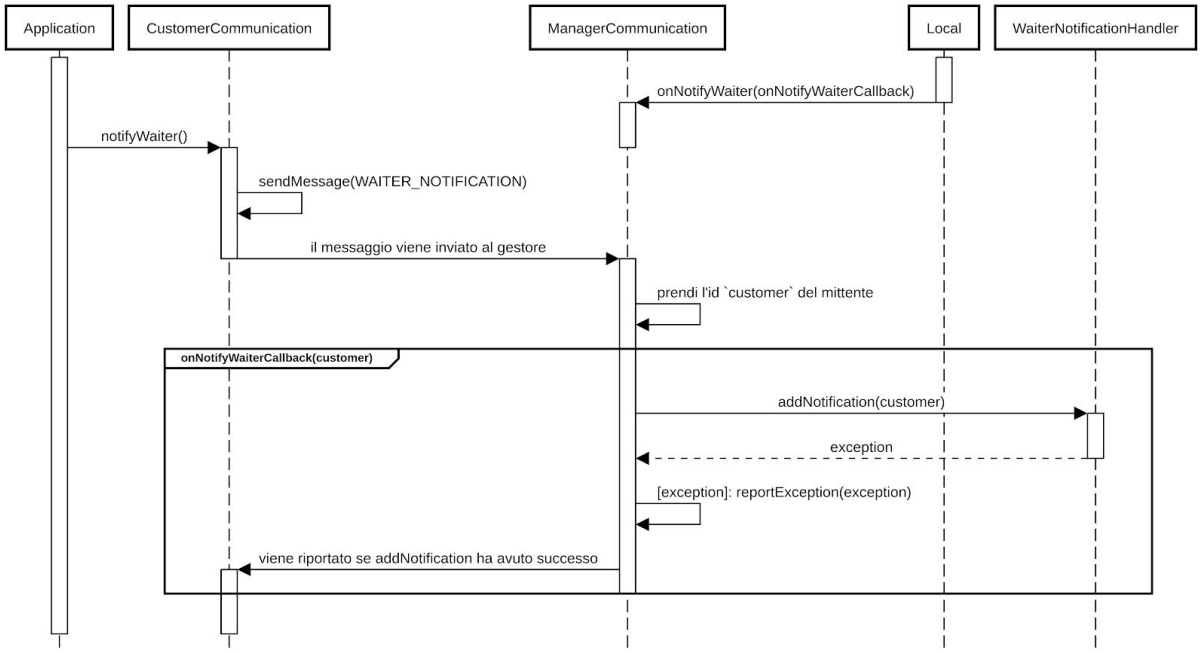


CD-11: Immetti numero tavolo



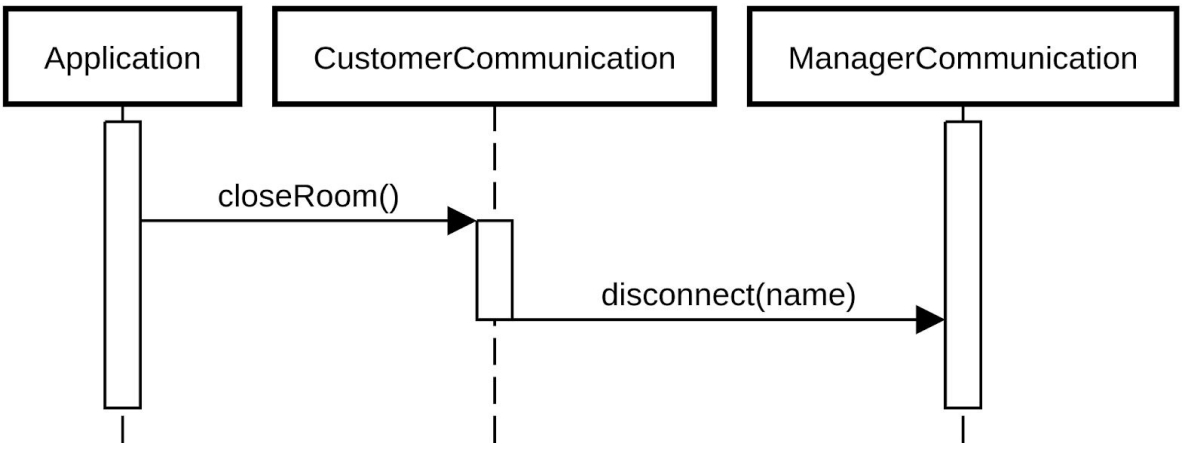
Classi usate ma non specificate: `Table`, `ManagerTable`, `Message`

CD-12: Chiama cameriere

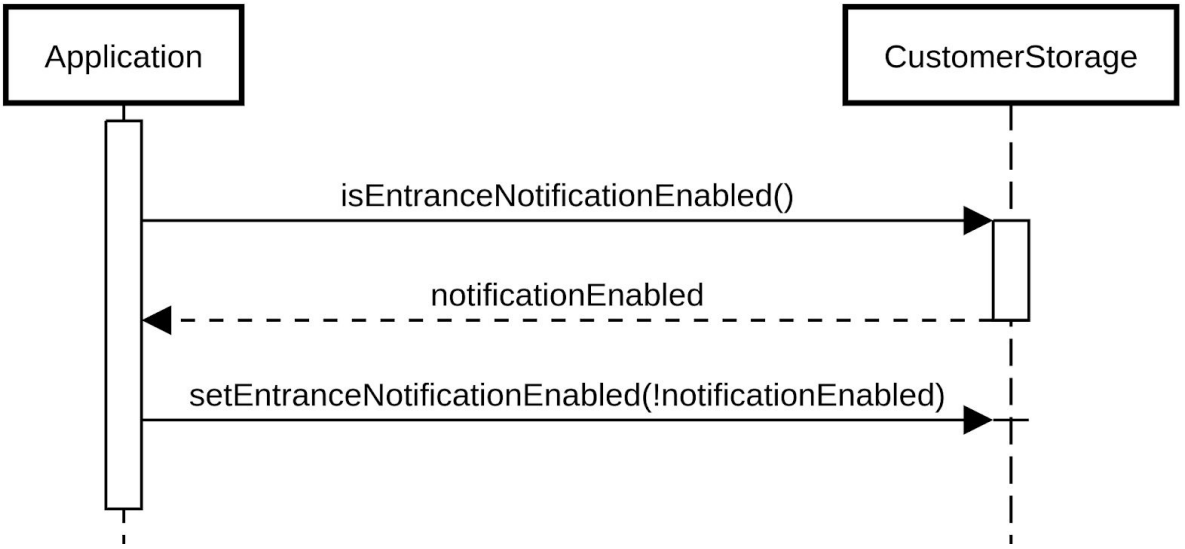


Classi usate ma non specificate: `WaiterNotification`, `Message`

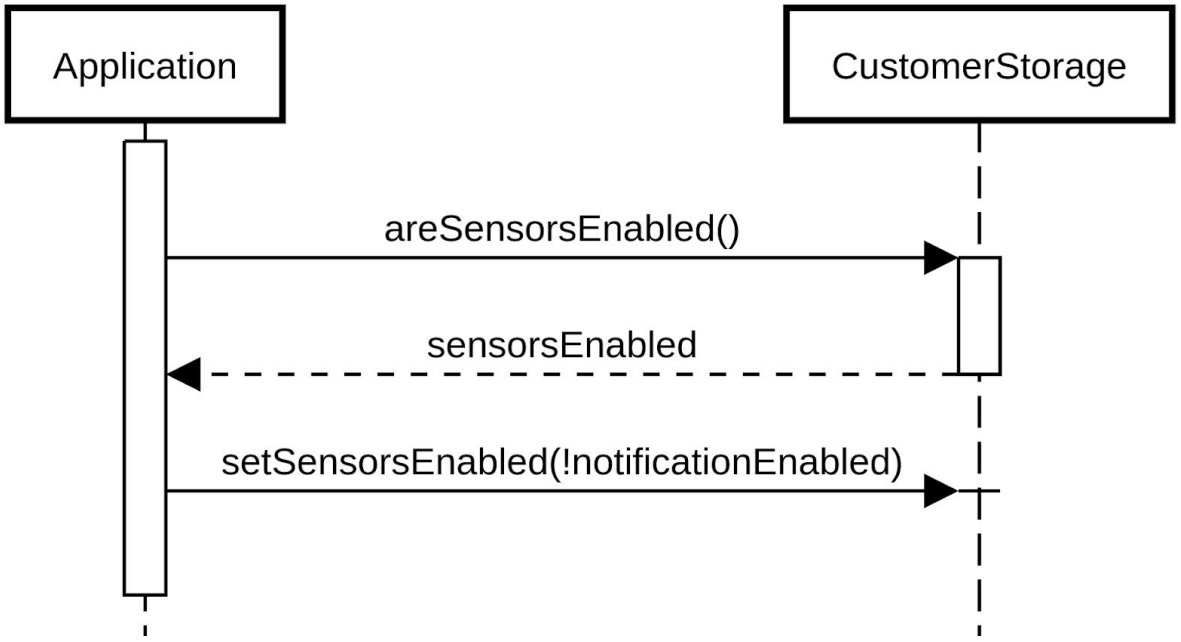
CD-13: Esci



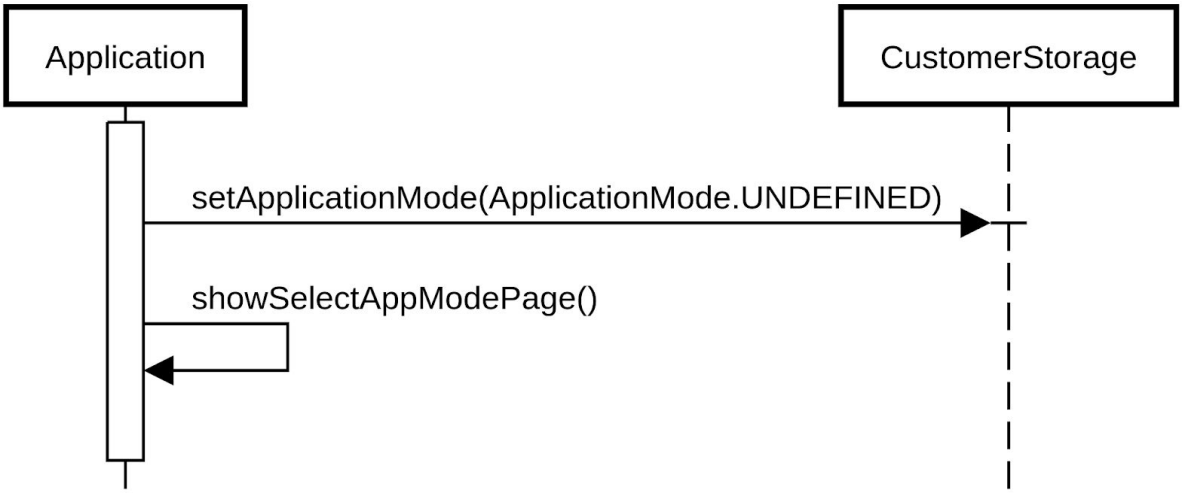
CD-14: Attiva/disattiva notifica benvenuto



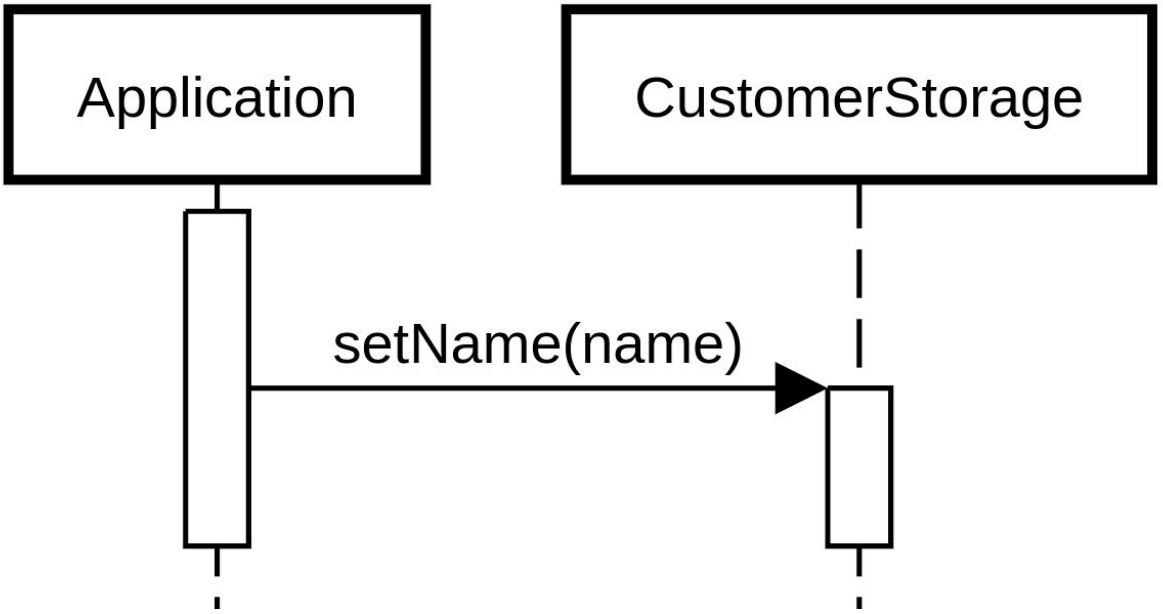
CD-15:Attiva/disattiva uso sensori



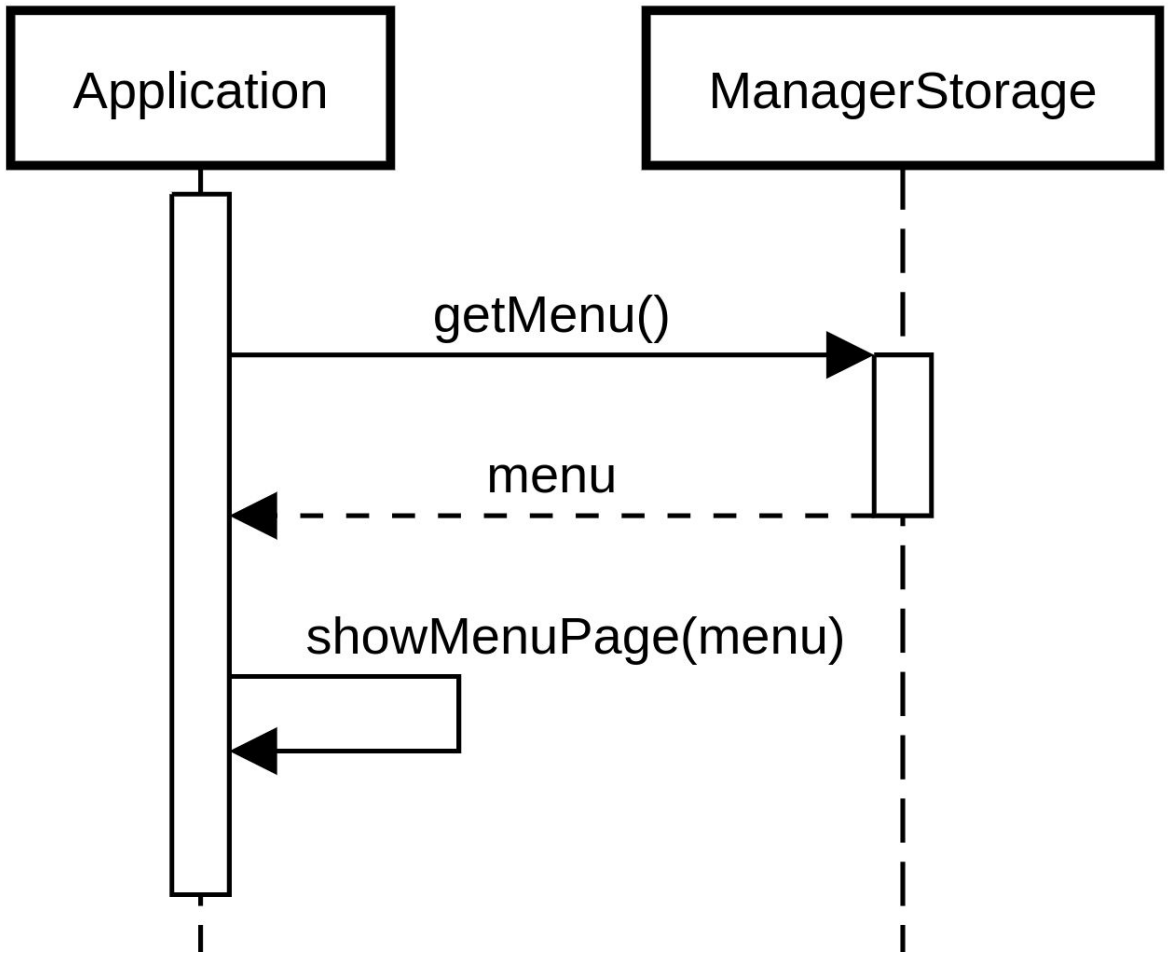
CD-16: Esci da modalità cliente



CD-17: Modifica nome

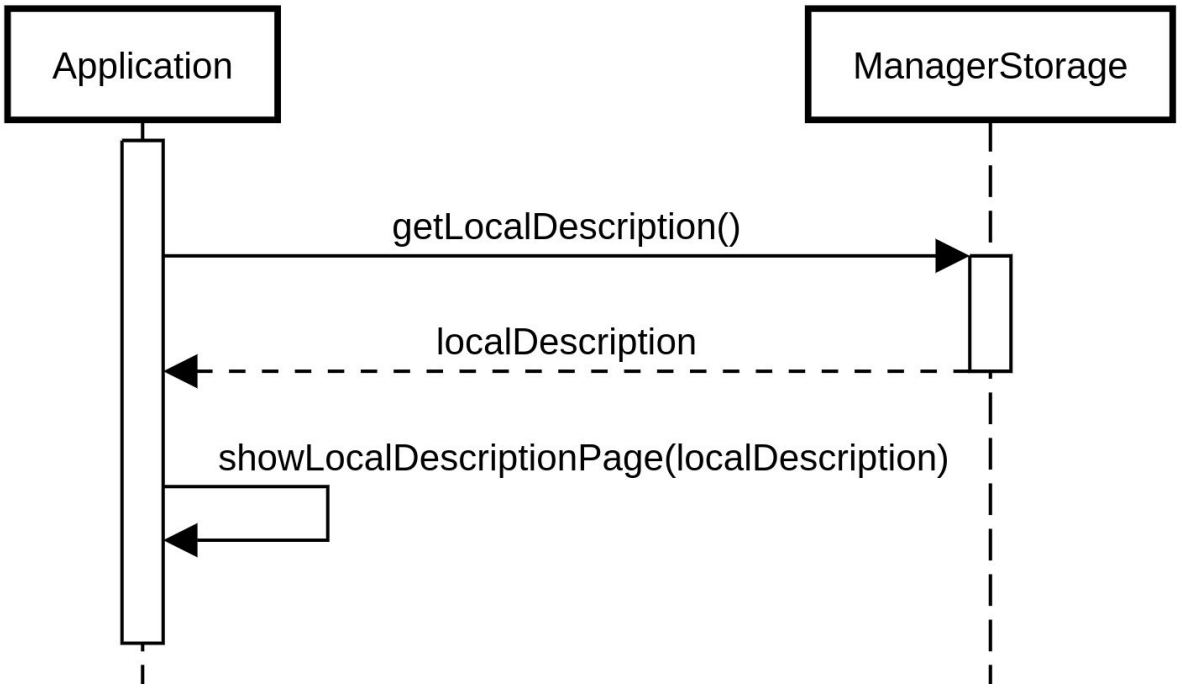


CD-18: Visualizza menù



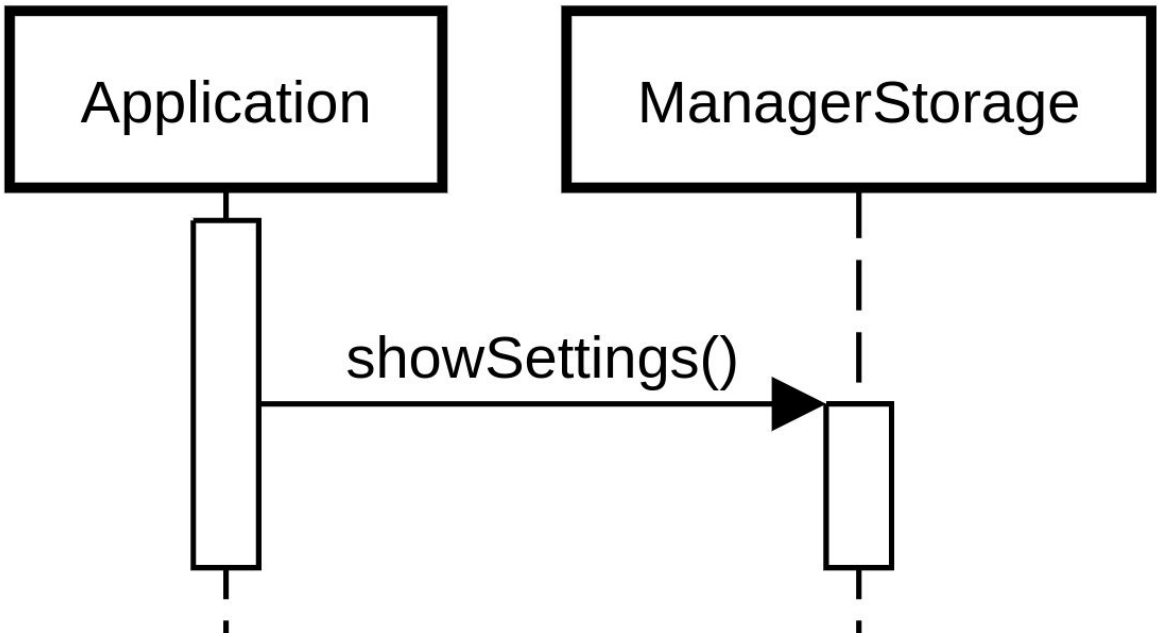
Classi usate ma non specificate: `LocalDescription`

CD-19: Visualizza descrizione

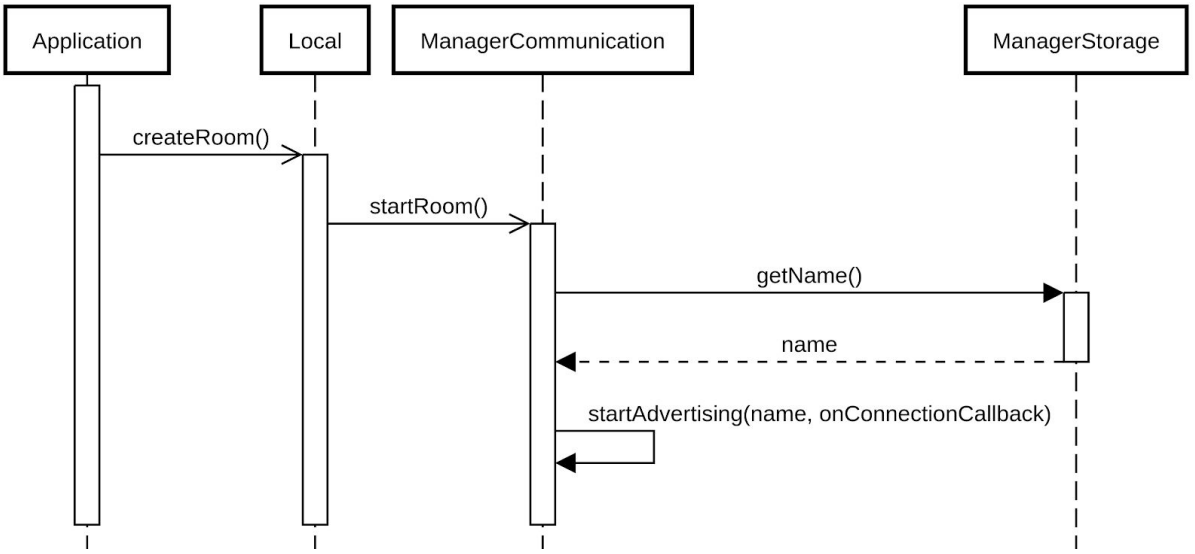


Classi usate ma non specificate: `LocalDescription`

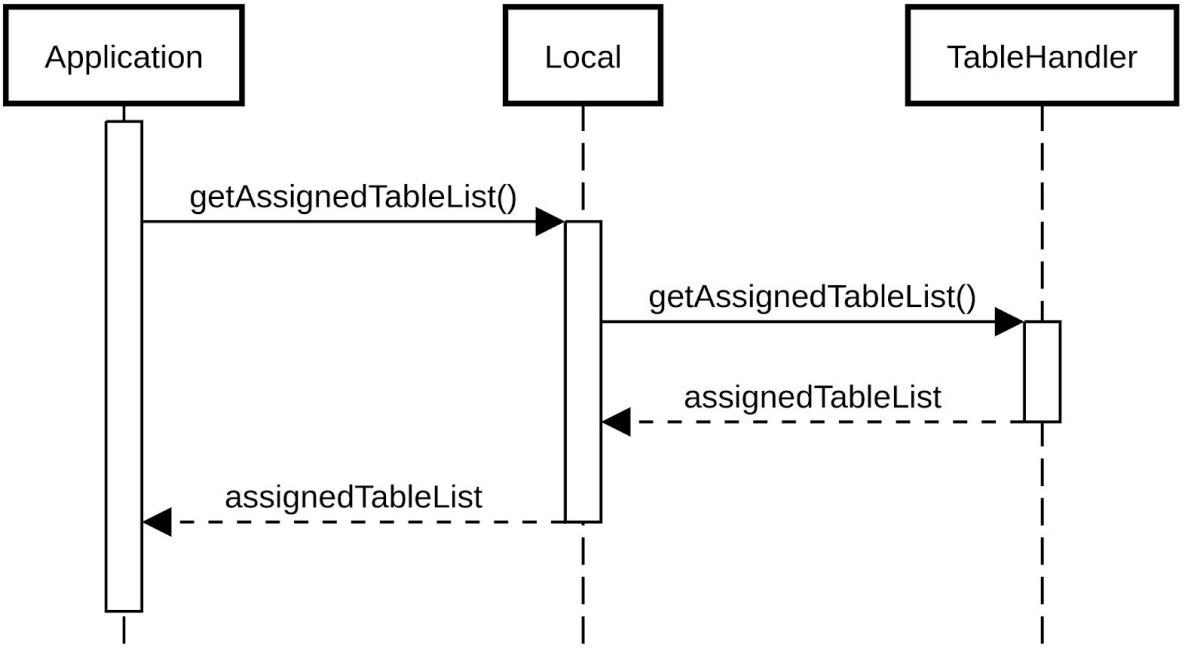
CD-20: Impostazioni



CD-21: Crea stanza virtuale

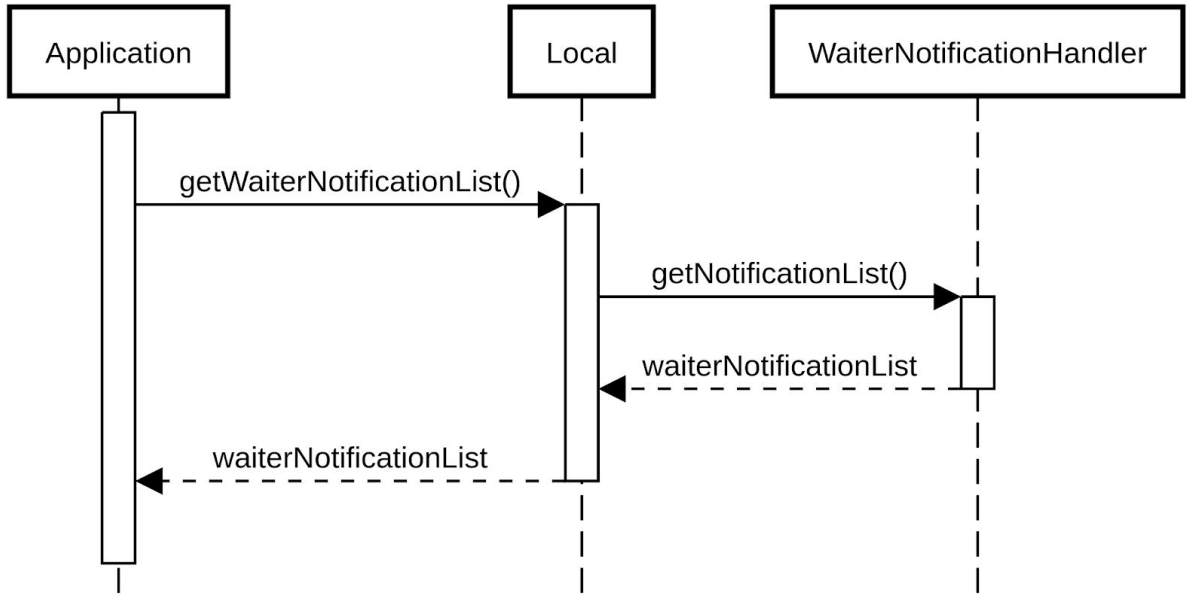


CD-22: Lista tavoli occupati



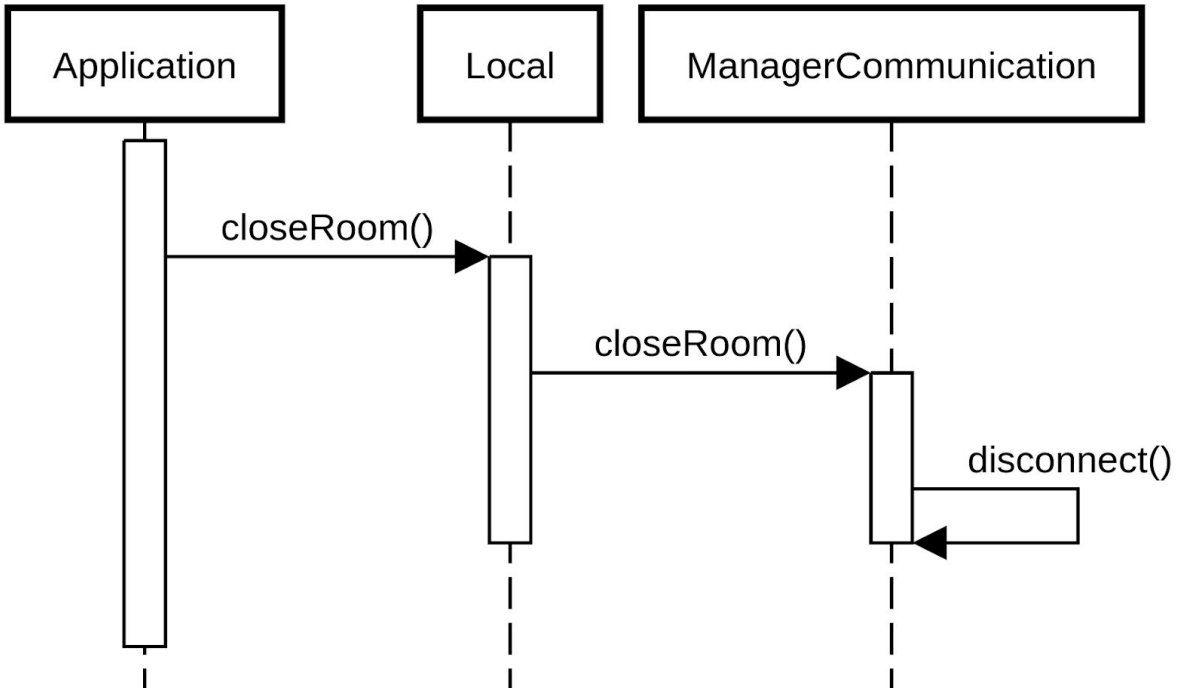
Classi usate ma non specificate: `Table, ManagerTable`

CD-23: Lista notifiche cameriere



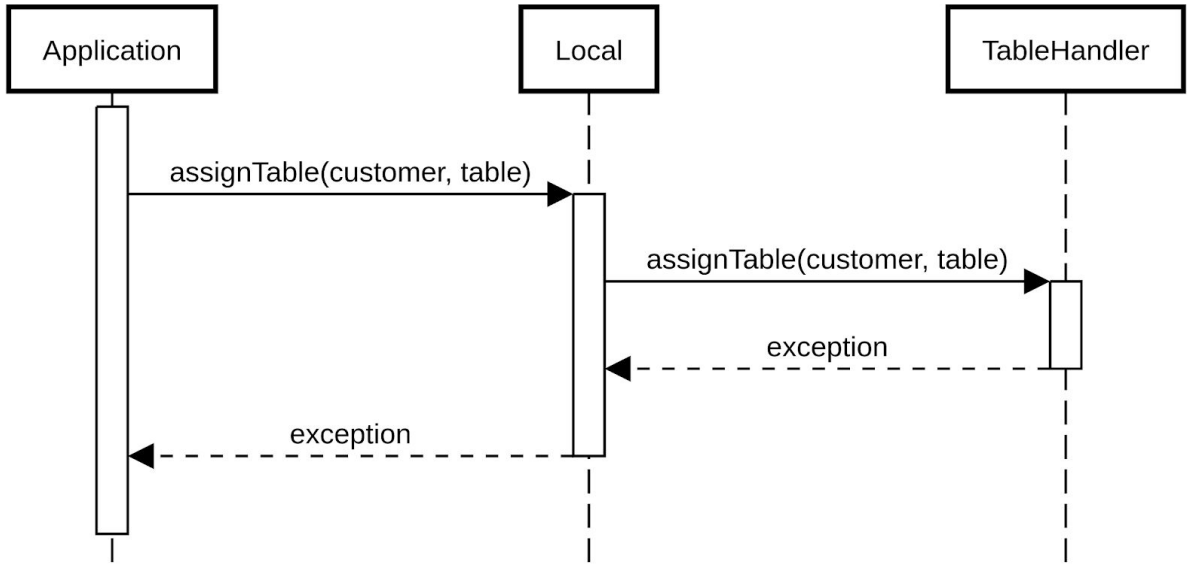
Classi usate ma non specificate : `WaiterNotification`

CD-24: Chiudi stanza



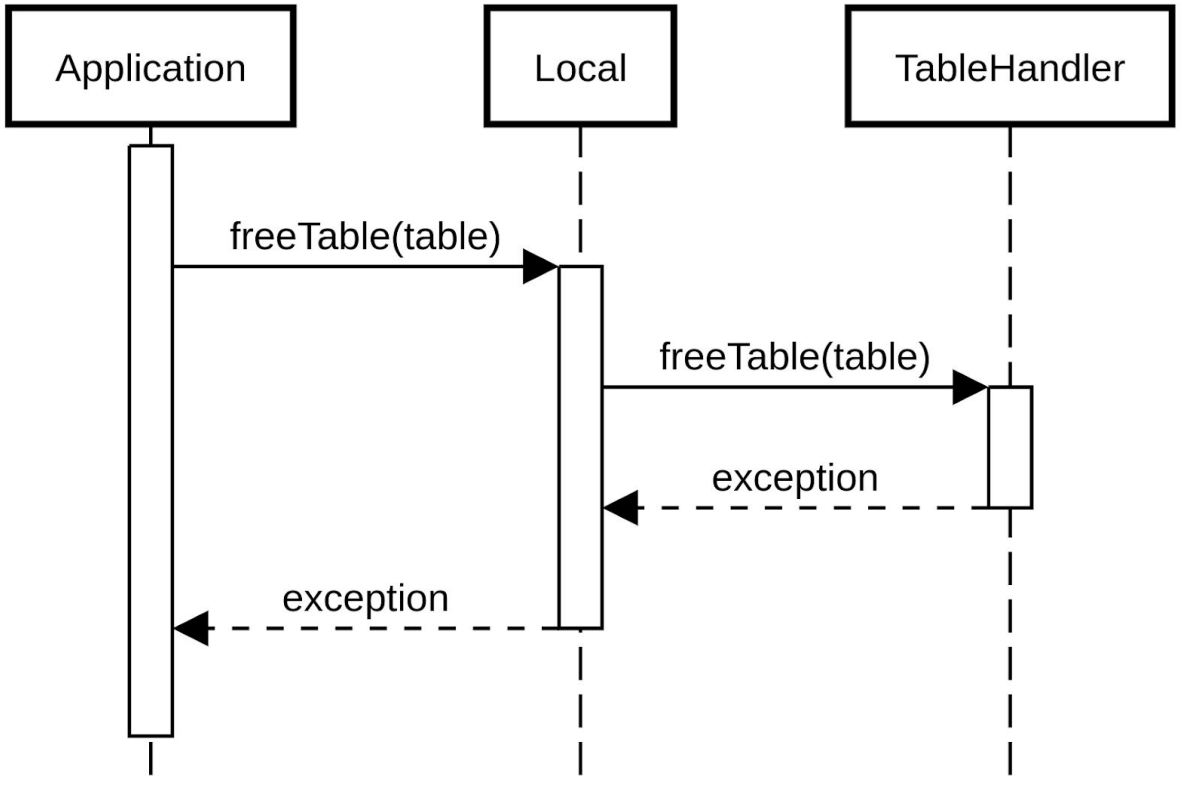
Classi usate ma non specificate: `Message`

CD-25: Aggiungi tavolo



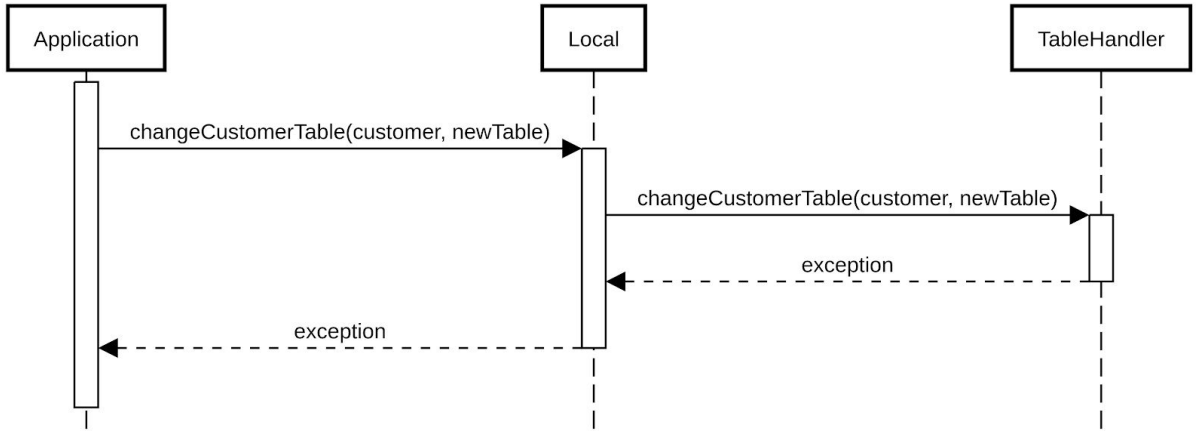
Classi usate ma non specificate: `Table, ManagerTable`

CD-26: Rimuovi tavolo



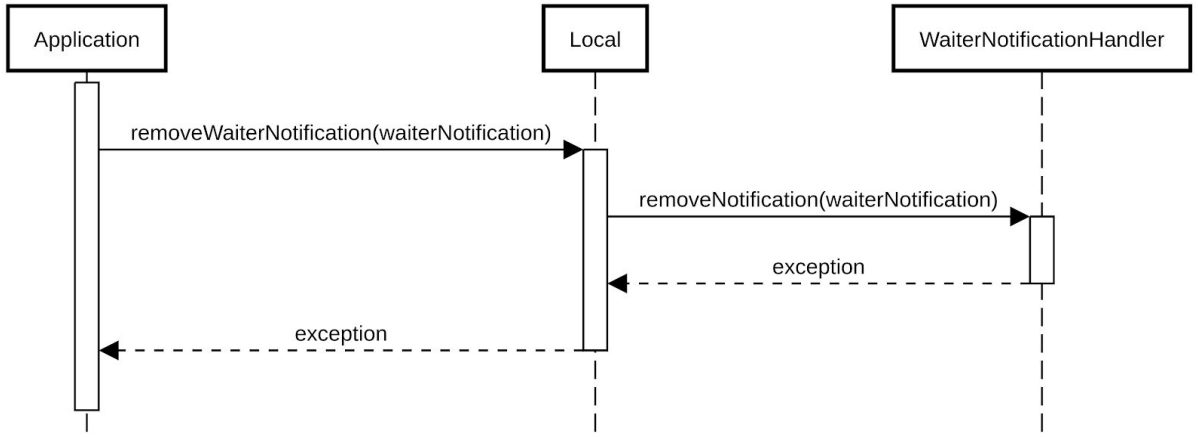
Classi usate ma non specificate: `Table, ManagerTable`

CD-27: Modifica tavolo



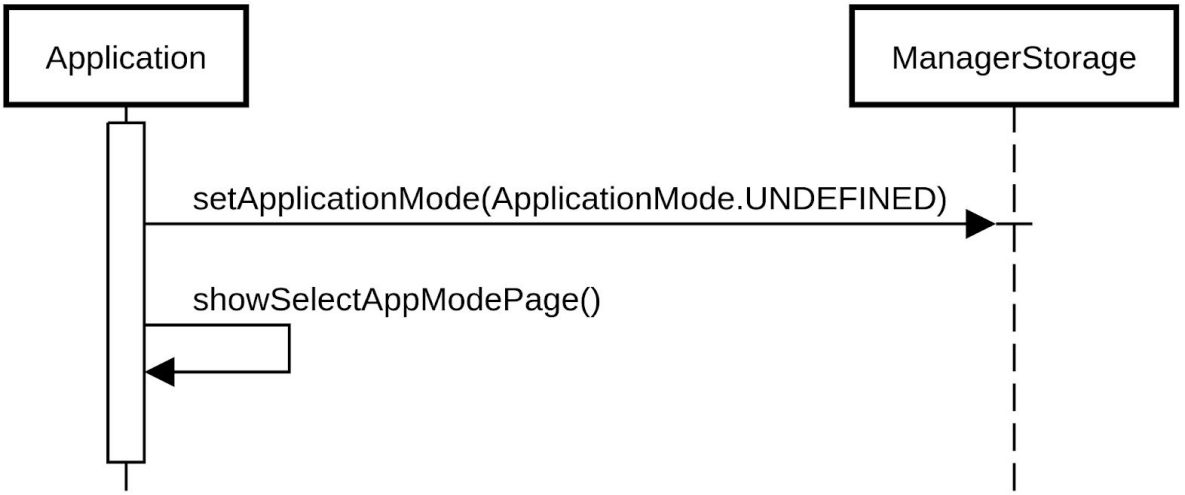
Classi usate ma non specificate: `Table, ManagerTable`

CD-28: Prendi in carico



Classi usate ma non specificate: `WaiterNotification`

CD-29: Esci da modalità gestore

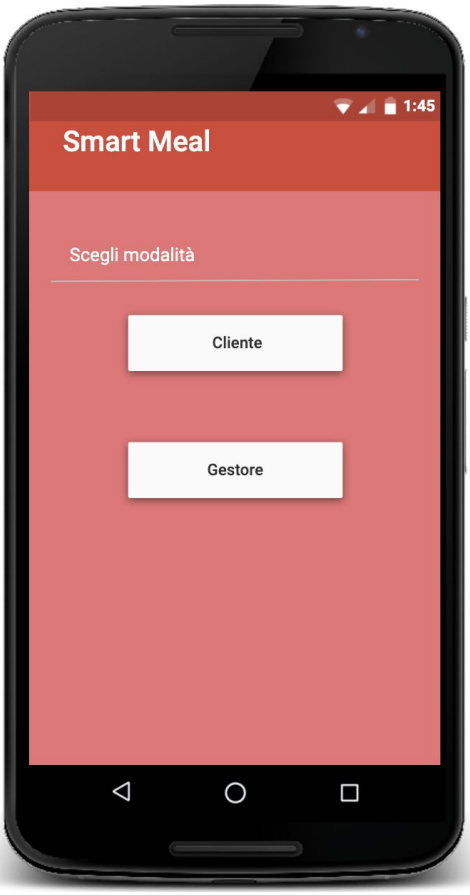


Classi usate ma non specificate: `ApplicationMode`

7. Progettazione dell’interfaccia Utente

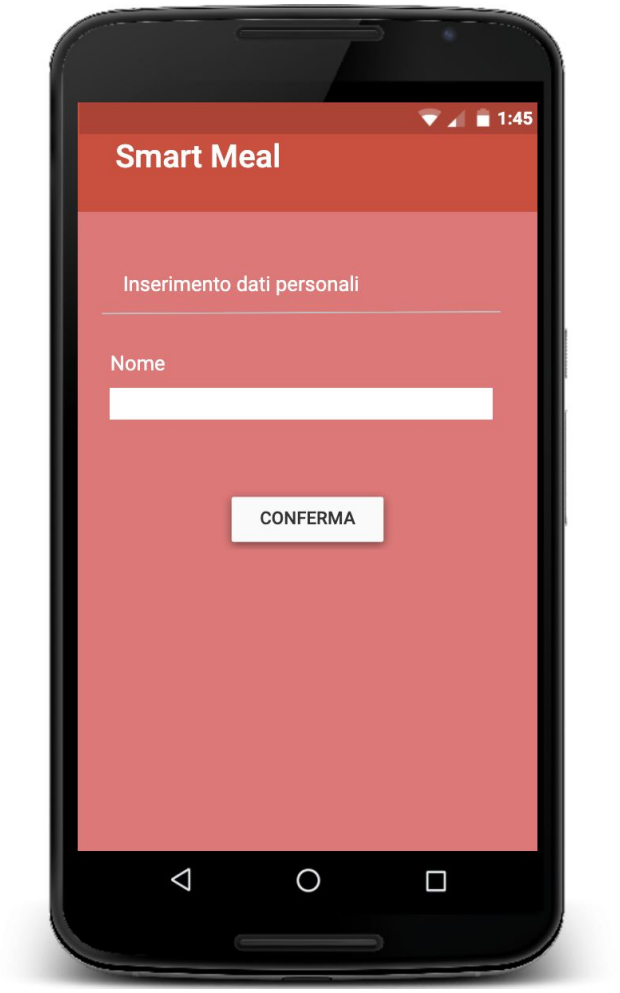
7.1 Selezione modalità applicazione

Al primo accesso è richiesto all’utente di scegliere la modalità dell’applicazione.
Selezionando “cliente” si passa alla schermata 7.2 (selezione nome) ; selezionando “gestore” si passa alla schermata 7.6 (inserimento pin).



7.2 Inserimento dati personali

Se si sceglie come modalità dell'applicazione "cliente" allora verrà chiesto di inserire il nome. Confermando si passa alla schermata 7.3 (schermata principale cliente).

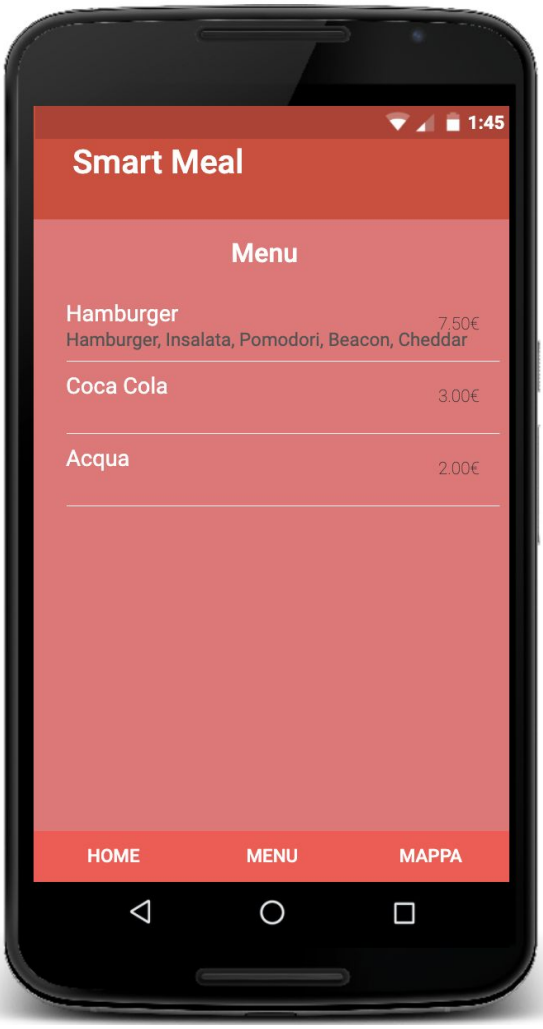


7.3 Schermata principale cliente

E’ la schermata principale (home) dell’applicazione modalità “cliente”. Ci si arriva nel primo accesso dopo aver confermato il nome. In tutti gli accessi successivi dell’applicazione modalità “cliente” questa è la prima schermata che si visualizza.
Si visualizzano nome, descrizione e foto del locale. Inoltre si possono visualizzare menù (schermata 7.4) e posizione del locale (schermata 7.6) con gli appositi bottoni.
Cliccando sul floating button si tenta di accedere alla stanza virtuale (schermata 7.7).
Infine cliccando la rotellina in alto a destra si accede alle impostazioni (schermata 7.5).

7.4 Visualizzazione menù

Schermata in cui si visualizza il menù. Ci si arriva cliccando il bottone “menù”.

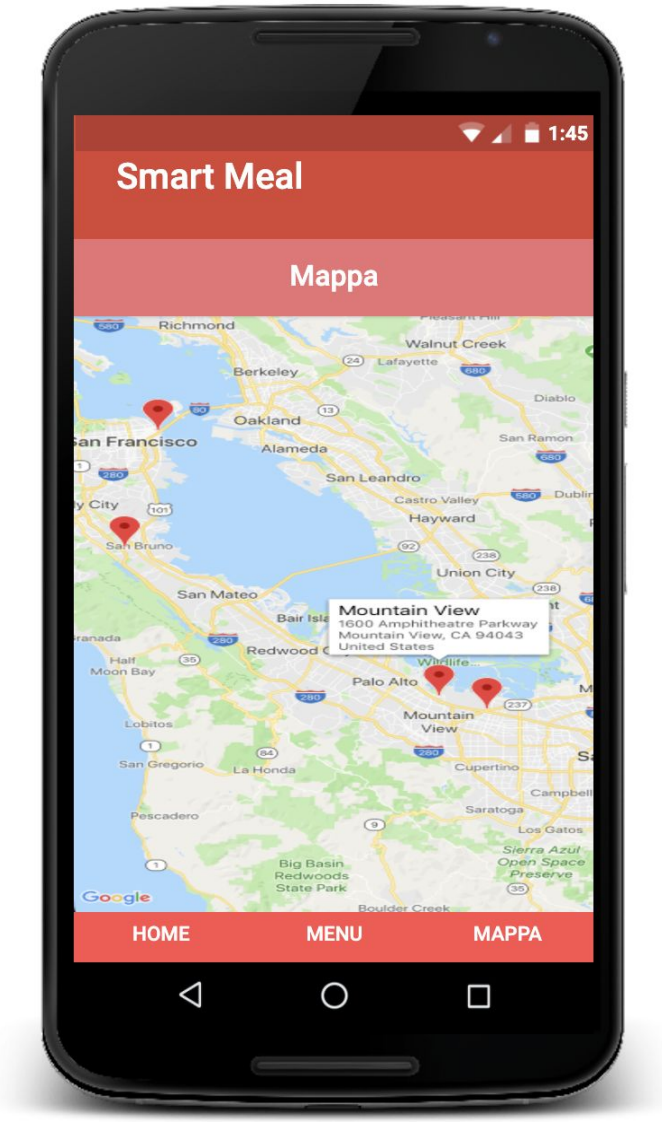


7.5 Impostazioni applicazione modalità “cliente”

Schermata delle impostazioni dell’applicazione modalità “cliente”. Ci si arriva cliccando la rotellina in alto a destra della schermata principale (schermata 7.3),con applicazione in modalità “cliente”.

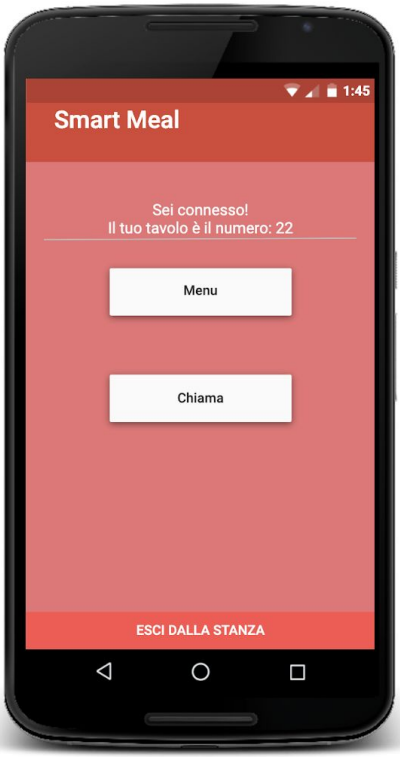
7.6 Visualizzazione mappa

Schermata di visualizzazione della mappa. Ci si arriva cliccando il bottone “mappa”.



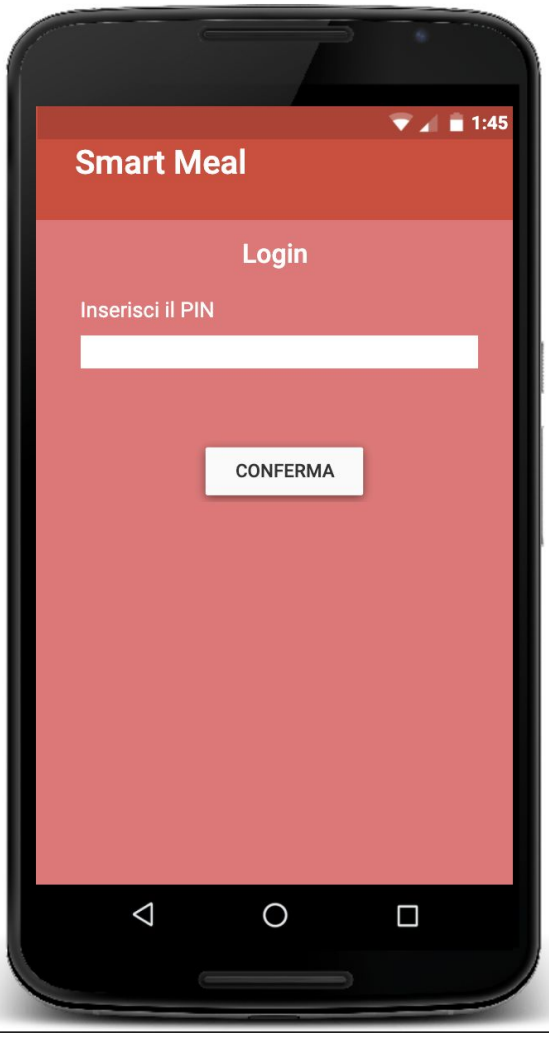
7.7 Stanza virtuale lato cliente

Dalla schermata a sinistra il cliente ha la possibilità, dopo essersi connesso alla stanza virtuale (creata dal gestore) tramite il floating button nella Home (schermata 7.3), di selezionare un tavolo tra quelli disponibili.
Se l'operazione viene annullata ritorna alla Home, mentre se conferma entra nella stanza virtuale e viene portato nella pagina a destra in cui ha la possibilità di effettuare le operazioni quali la visualizzazione del menu del ristorante e chiamare il cameriere (schermata corrente, figura a destra).



7.8 Inserimento pin

Scegliendo come modalità dell'applicazione la modalità "gestore" si chiede di digitare il pin. Digitando il pin corretto si passa alla schermata principale gestore (schermata 7.9).

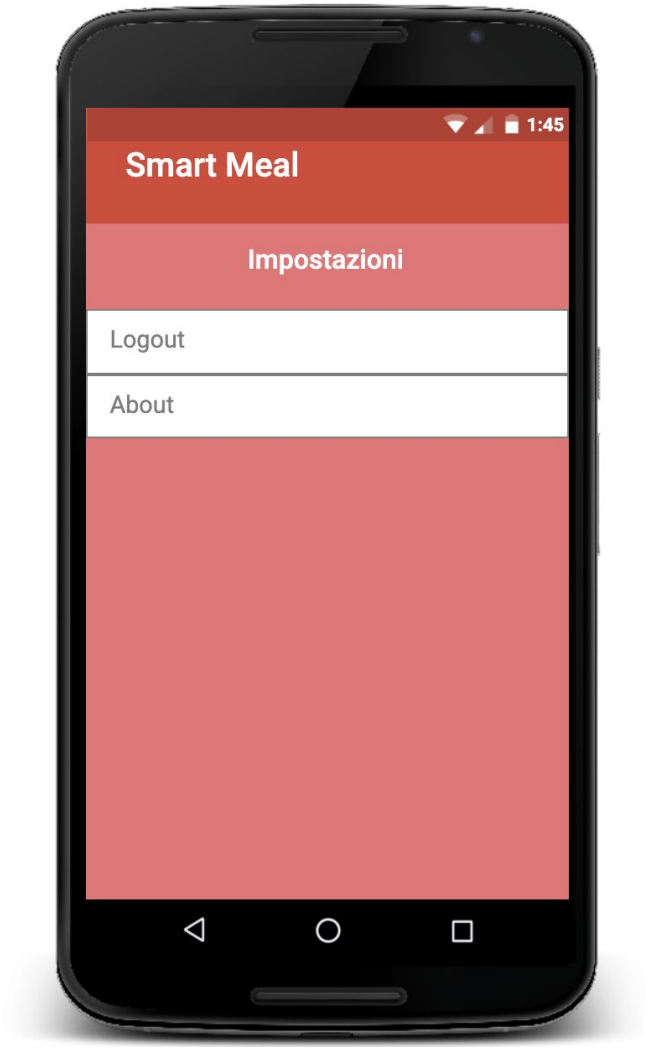


7.9 Schermata principale gestore

E’ la schermata principale (home) dell’applicazione modalità “gestore”. Ci si arriva nel primo accesso dopo aver digitato il pin corretto. In tutti gli accessi successivi dell’applicazione modalità “gestore” questa è la prima schermata che si visualizza.
Si possono visualizzare menu (schermata 7.4) e descrizione del locale. E si può creare la stanza virtuale (schermata 7.11).
Infine si possono accedere le impostazioni (schermata 7.10).

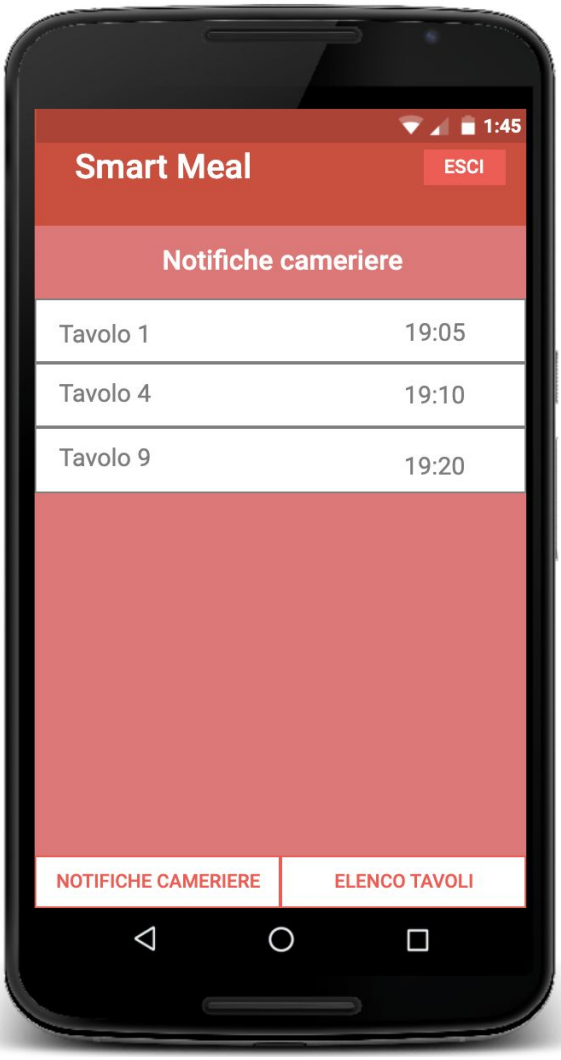
7.10 Impostazioni applicazione modalità applicazione

Schermata delle impostazioni dell'applicazione modalità "gestore". Ci si arriva cliccando la rotellina in alto a destra della schermata principale applicazione modalità "gestore" (schermata 7.9).



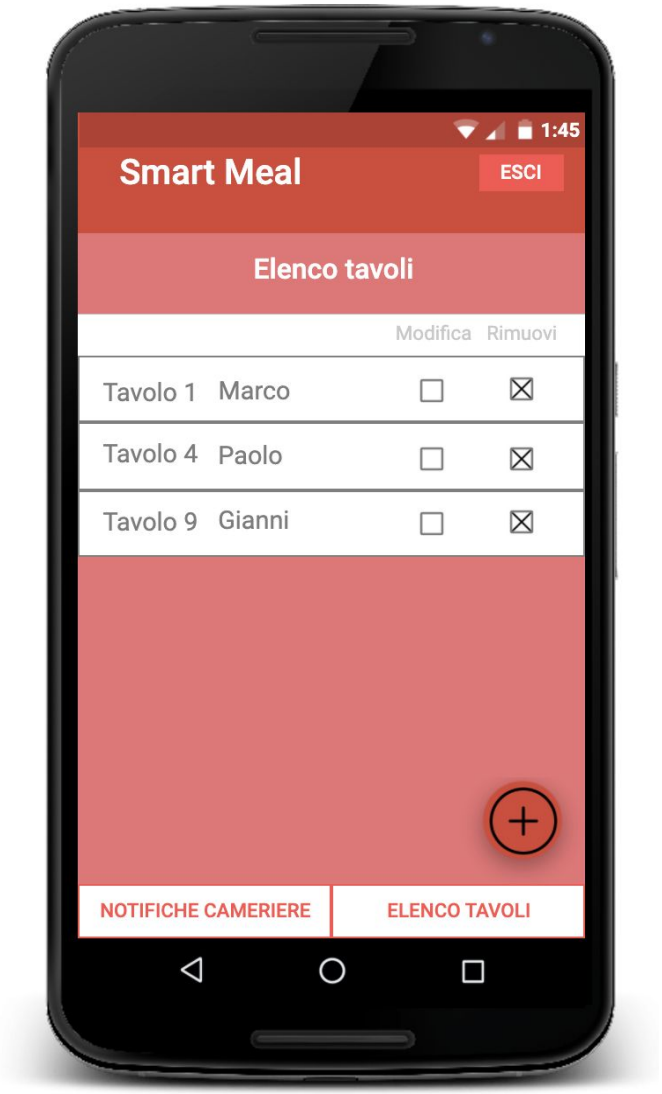
7.11 Stanza virtuale lato gestore

Schermata della stanza virtuale lato gestore. Si visualizza la lista di notifiche cameriere, ordinate per l'orario.
Cliccando il bottone "elenco tavoli" si passa alla gestione dei tavoli occupati (schermata 7.12). E cliccando su "esci" si chiude la stanza virtuale, ritornando alla schermata principale.



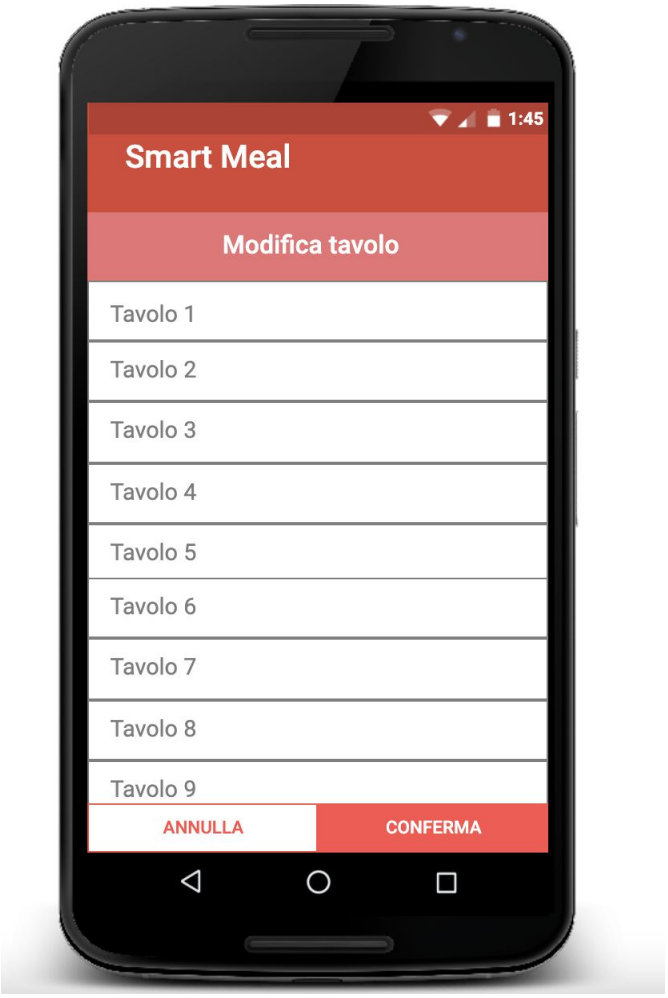
7.12 Gestione elenco tavoli

Nella seguente schermata, accessibile cliccando “ELENCO TAVOLI”, il gestore può gestire i vari tavoli del locale.
Vengono visualizzati tutti i tavoli occupati dai clienti e sono presenti le opzioni “Modifica” e “Rimuovi”. In modifica viene aperta la schermata “Modifica tavolo” (schermata 7.13) ed in rimuovi abbiamo la possibilità di rimuovere il cliente dal tavolo, facendo tornare libero il tavolo.
Infine il gestore, cliccando sul floating button, può aggiungere un tavolo occupato.
(Quest’ultima funzionalità serve per segnare i tavoli occupati da clienti che non hanno l’applicazione)



7.13 Modifica di un tavolo

La schermata sottostante è usata per sostituire il tavolo al cliente che ne aveva già uno: verranno visualizzati tutti i tavoli liberi ed il gestore potrà selezionare quello che vorrà assegnare al cliente.



7.14 Aggiunta tavolo

Schermata in cui il gestore aggiunge un tavolo occupato. Inserisce il nome del cliente e seleziona il tavolo libero da occupare.