Documento di Progettazione

Termini glossario

* Shake
* Nearby (magari citare versione)

## 

## 2. Architettura del Sistema

**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.

**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.

**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.

## **3. 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.

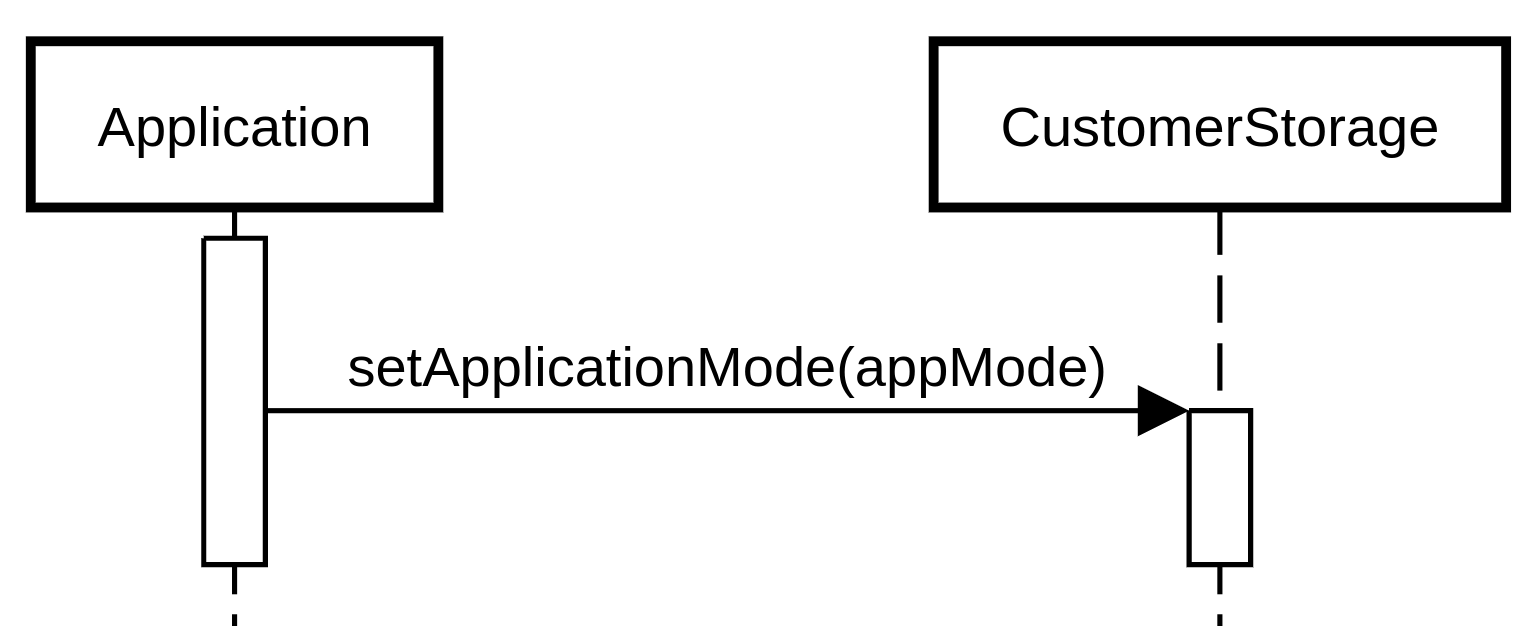
Per quanto riguarda il modello di controllo abbiamo deciso di adottare il modello basato su eventi in quanto il più naturale per lo sviluppo di un’app in Android dove sono richiesti diversi input da parte degli utenti e dove le callback sono ormai uno standard.

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.

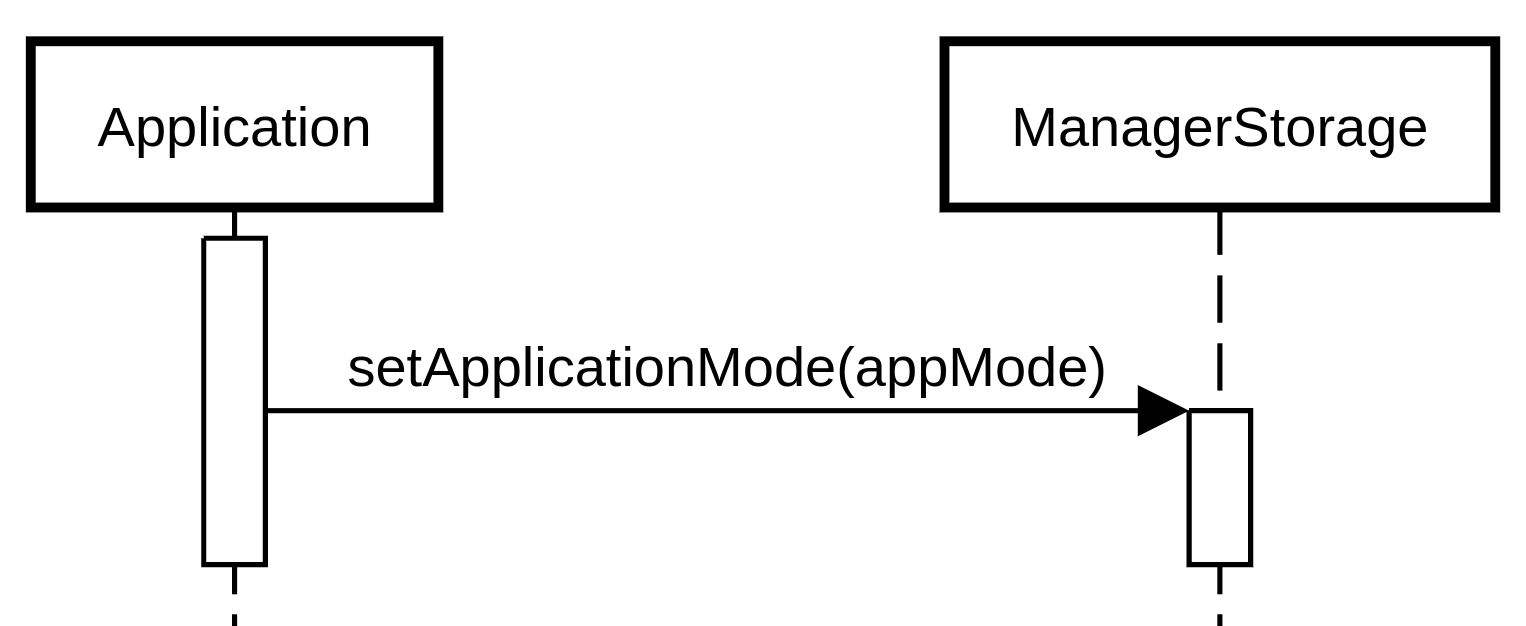
## 

## **4. Modelli UML: diagrammi di sequenza**

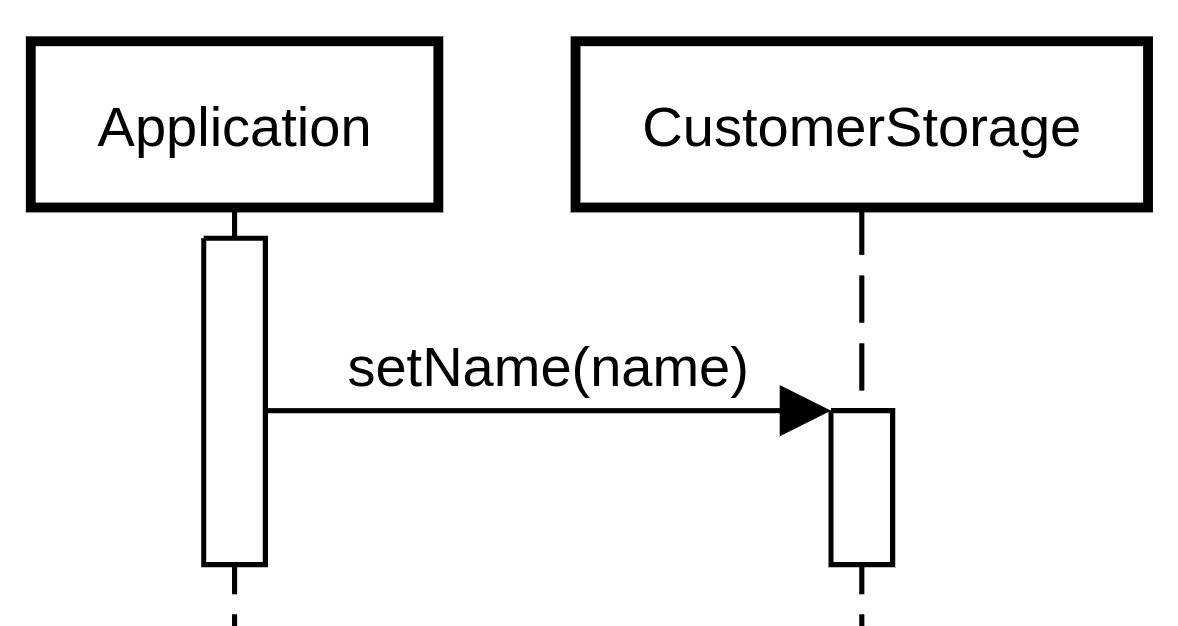
**CD-01: Scegli cliente**



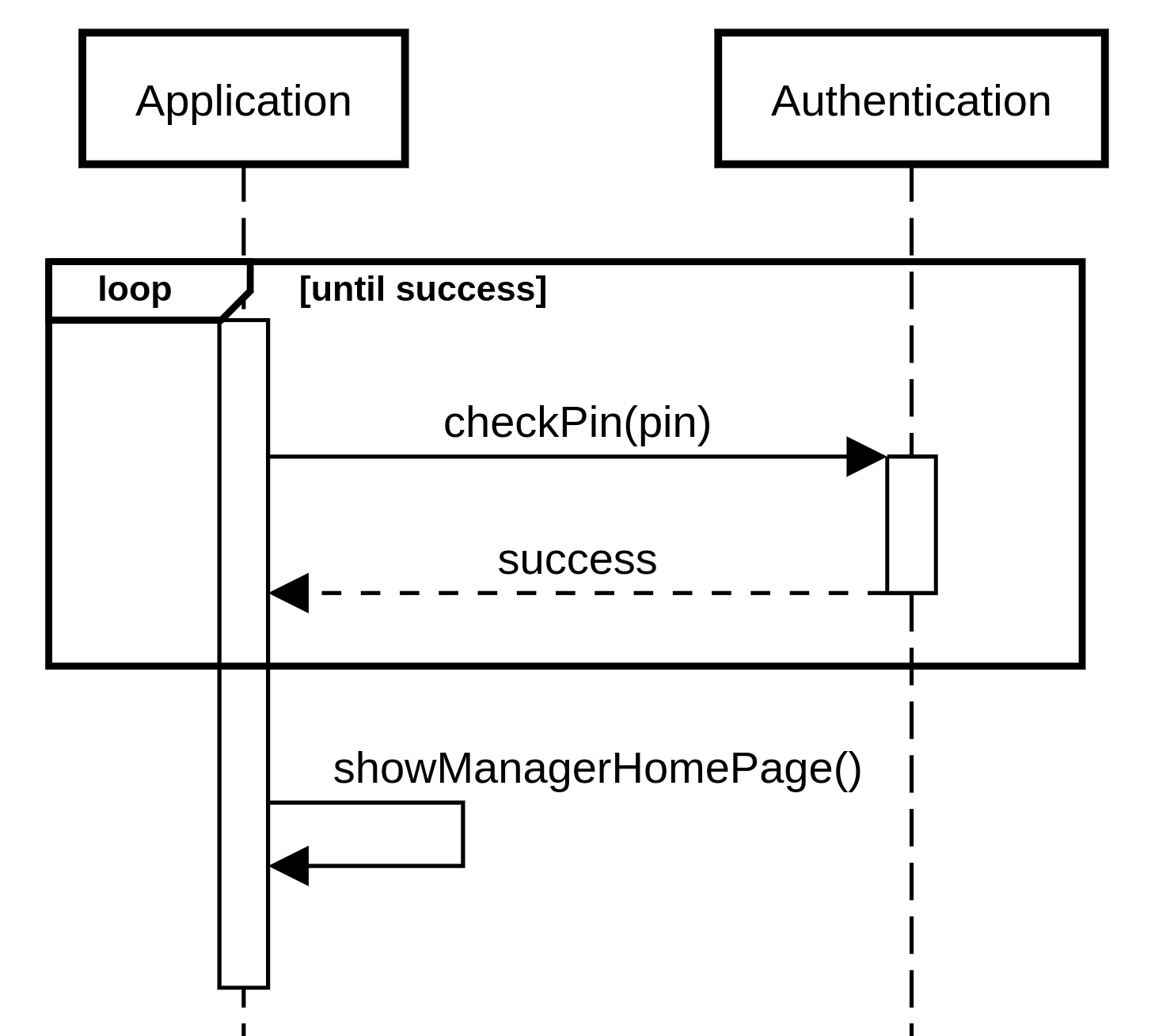
**CD-02: Scegli gestore**



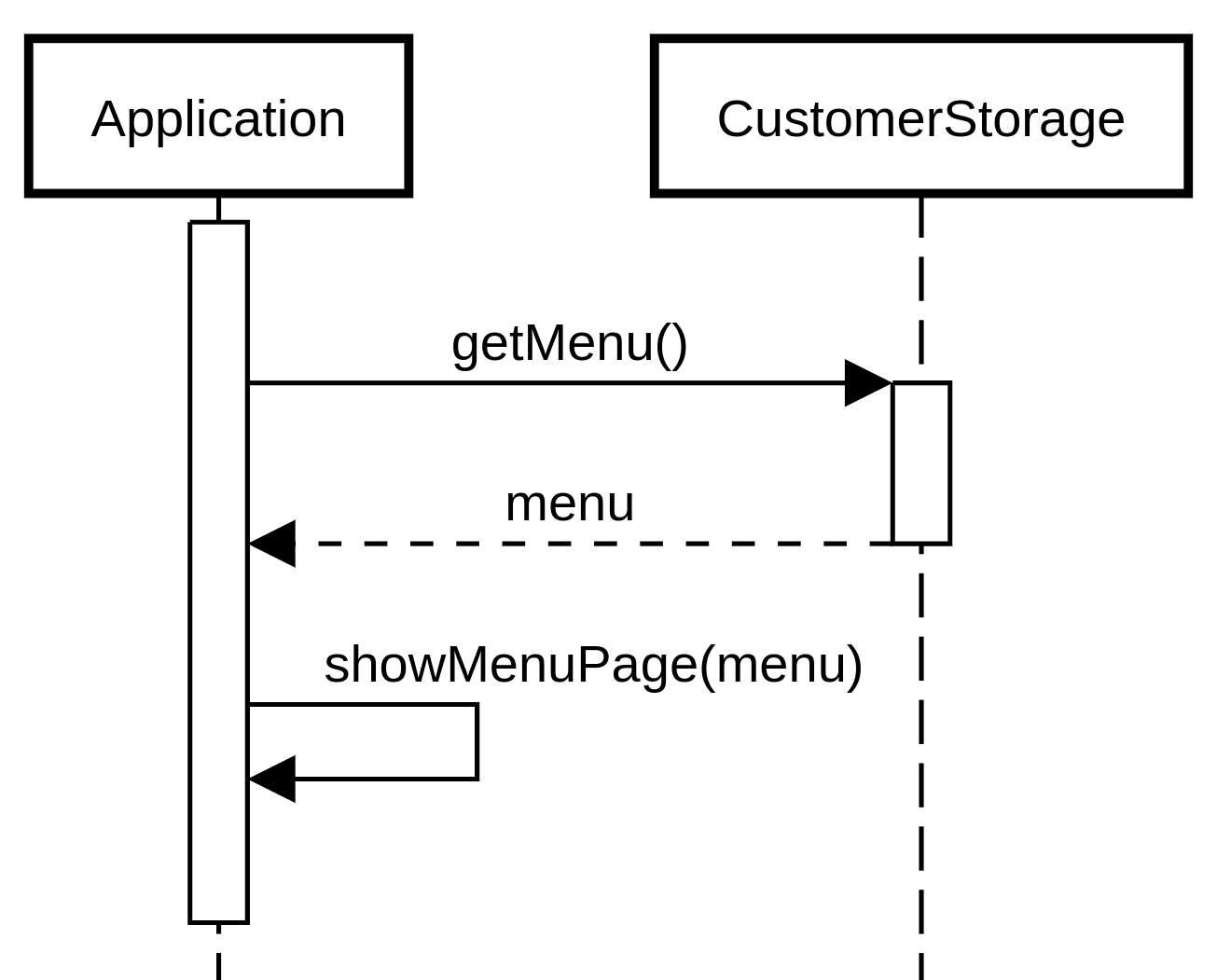
**CD-03: Inserimento nome**



**CD-04: Inserimento pin**

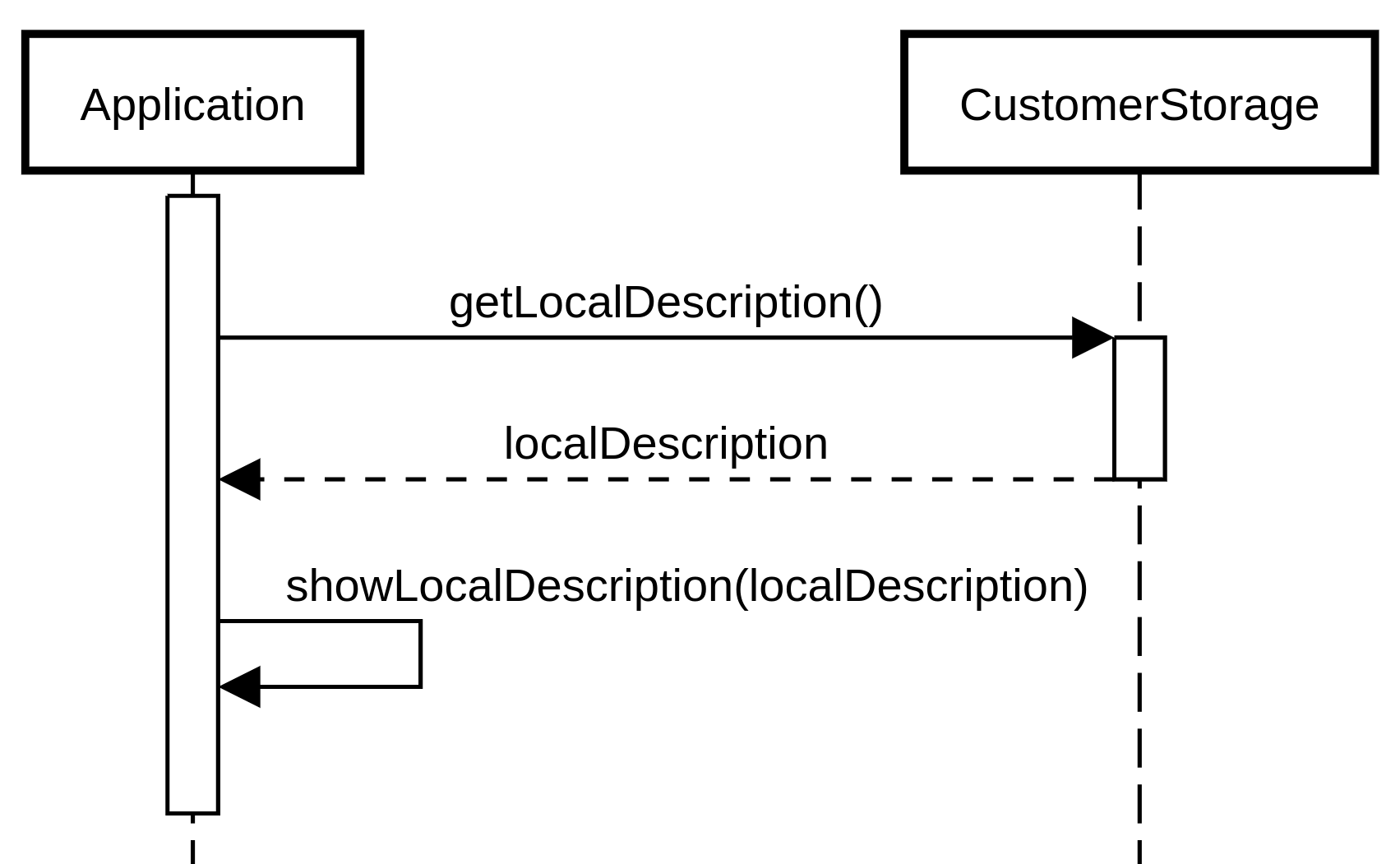


**CD-05: Visualizza menù**



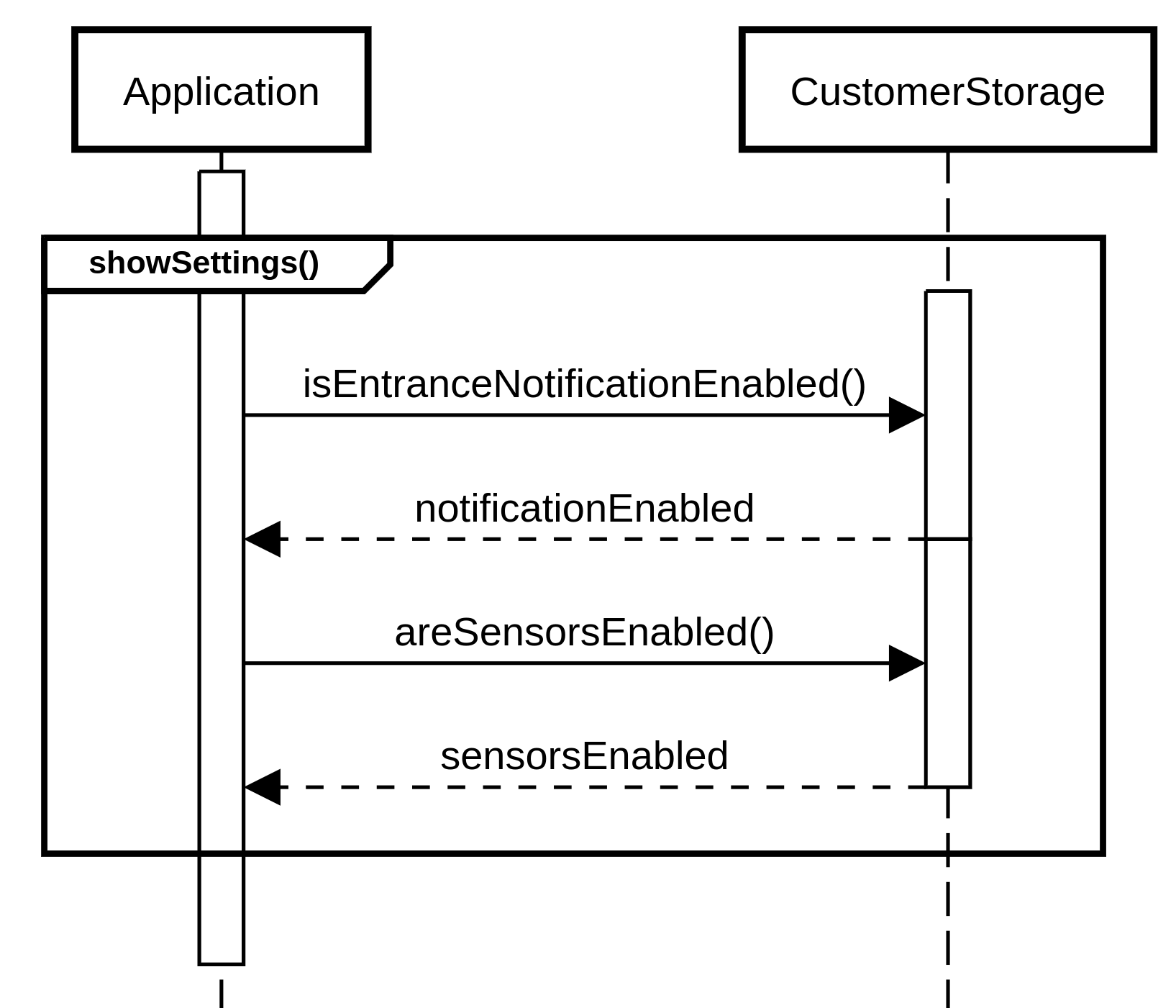
Classi usate ma non specificate: LocalDescription, Menu

**CD-06: Visualizza descrizione locale**

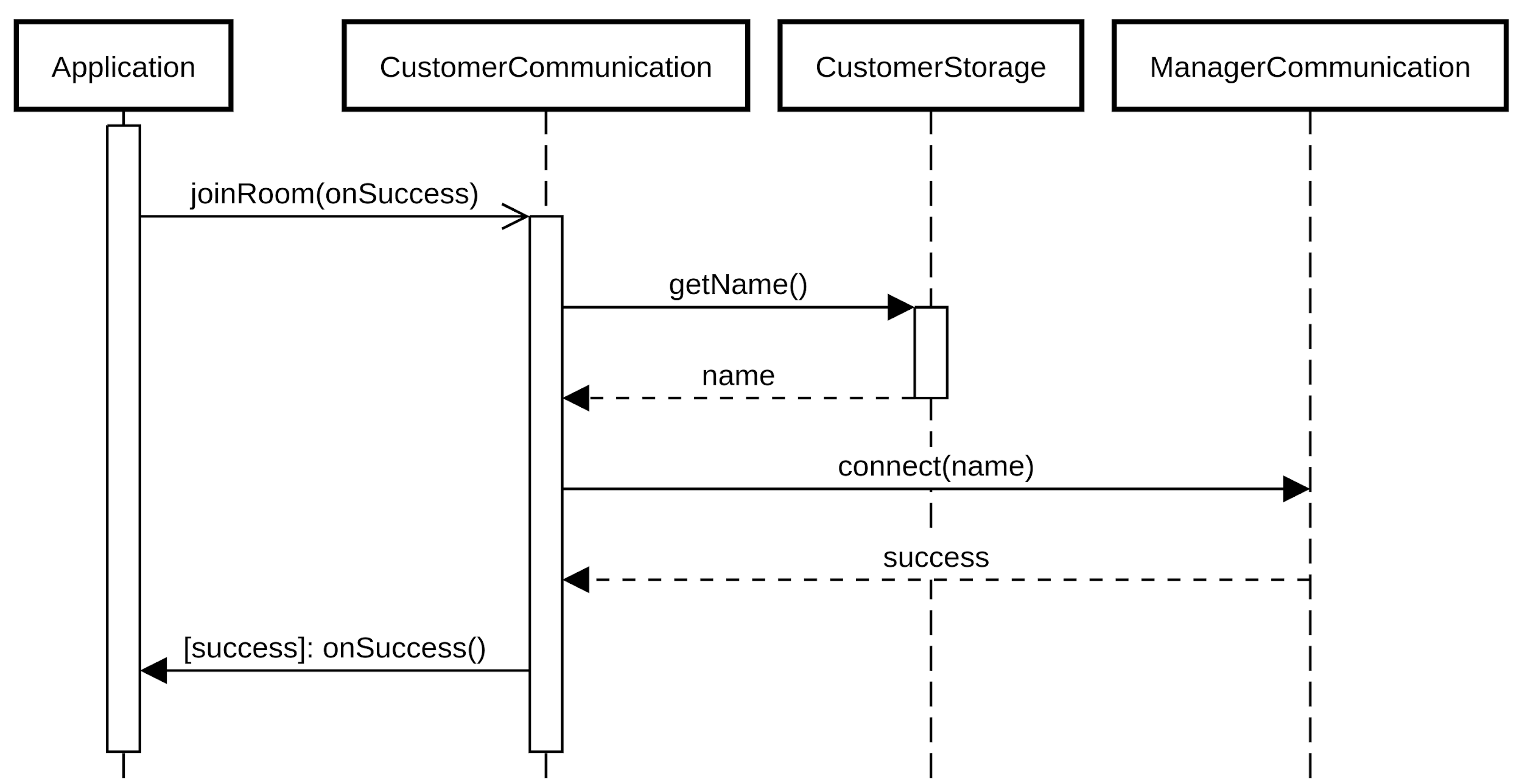


Classi usate ma non specificate: LocalDescription

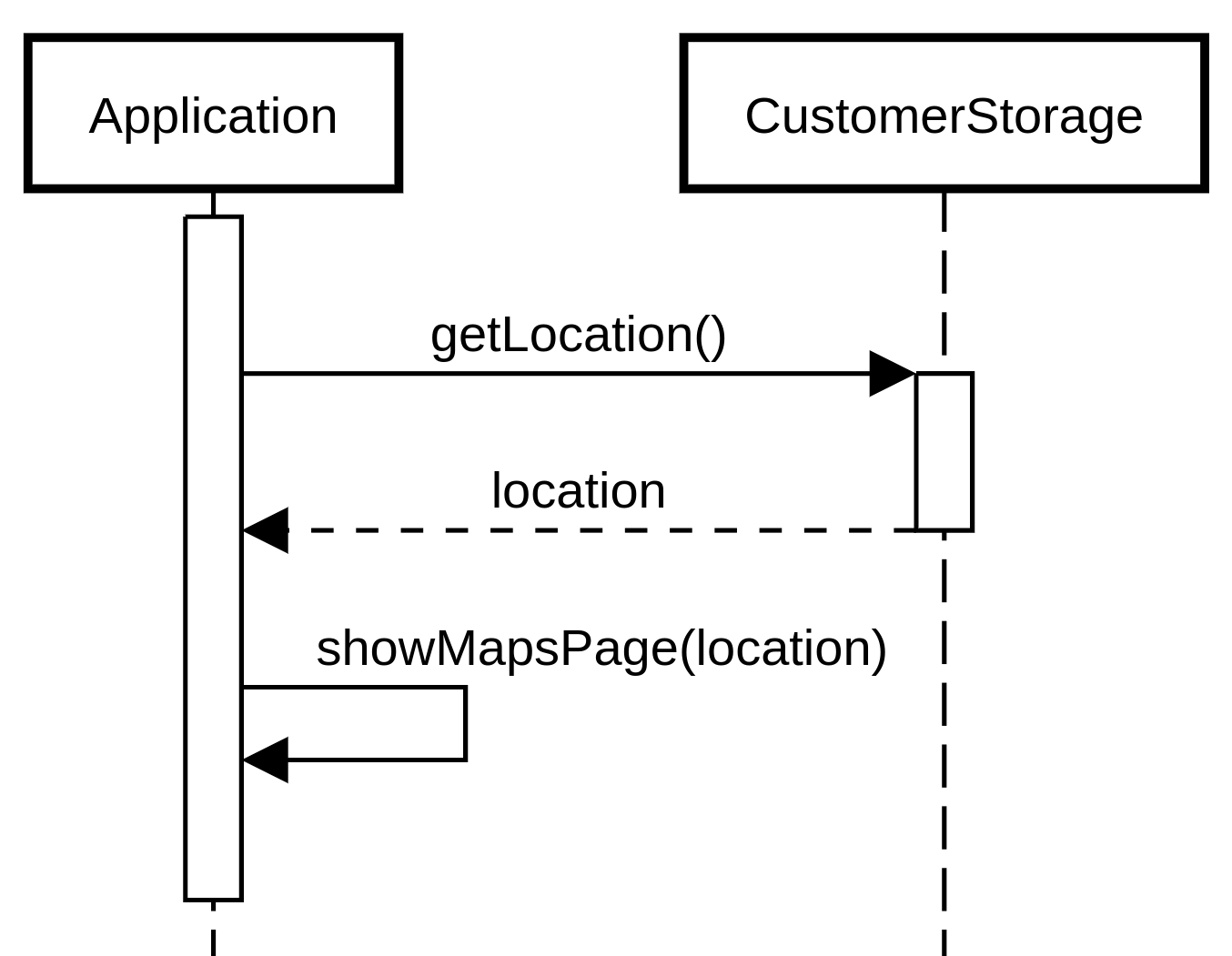
**CD-07: Impostazioni**



**CD-08: Accedi alla stanza virtuale**

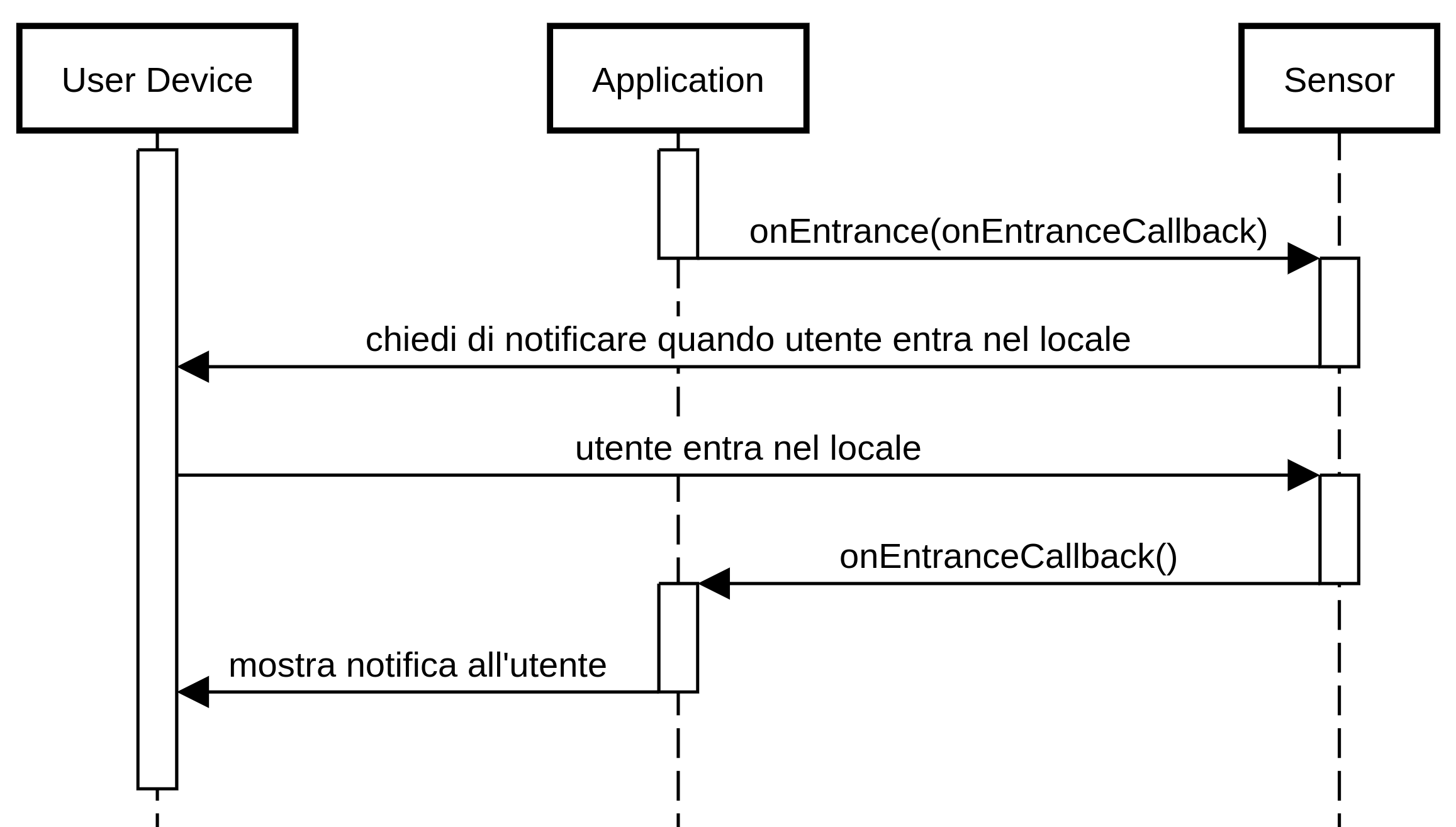


**CD-09: Visualizza mappa**

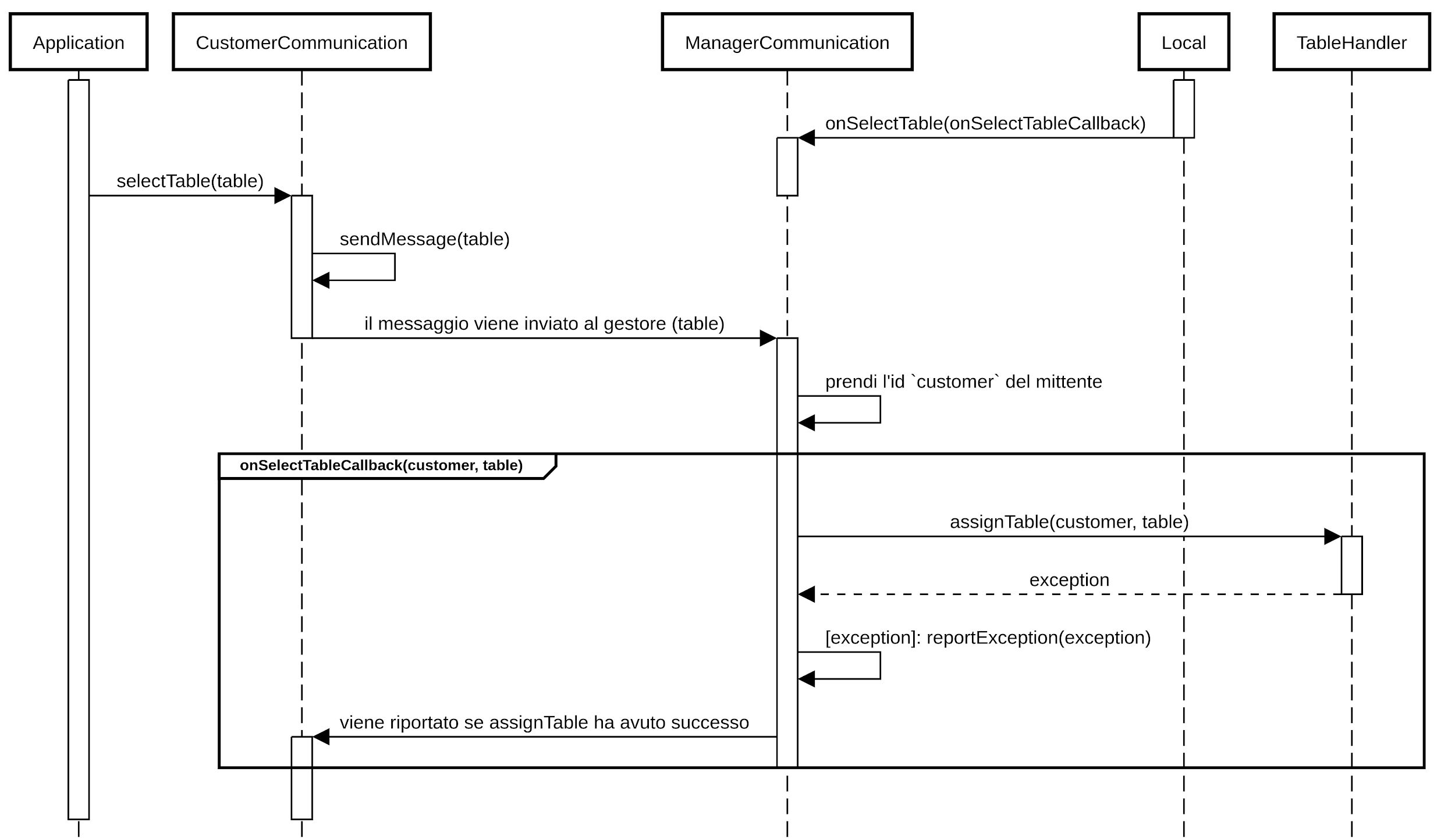


Classi usate ma non specificate: LocalDescription

**CD-10: Notifica di benvenuto**

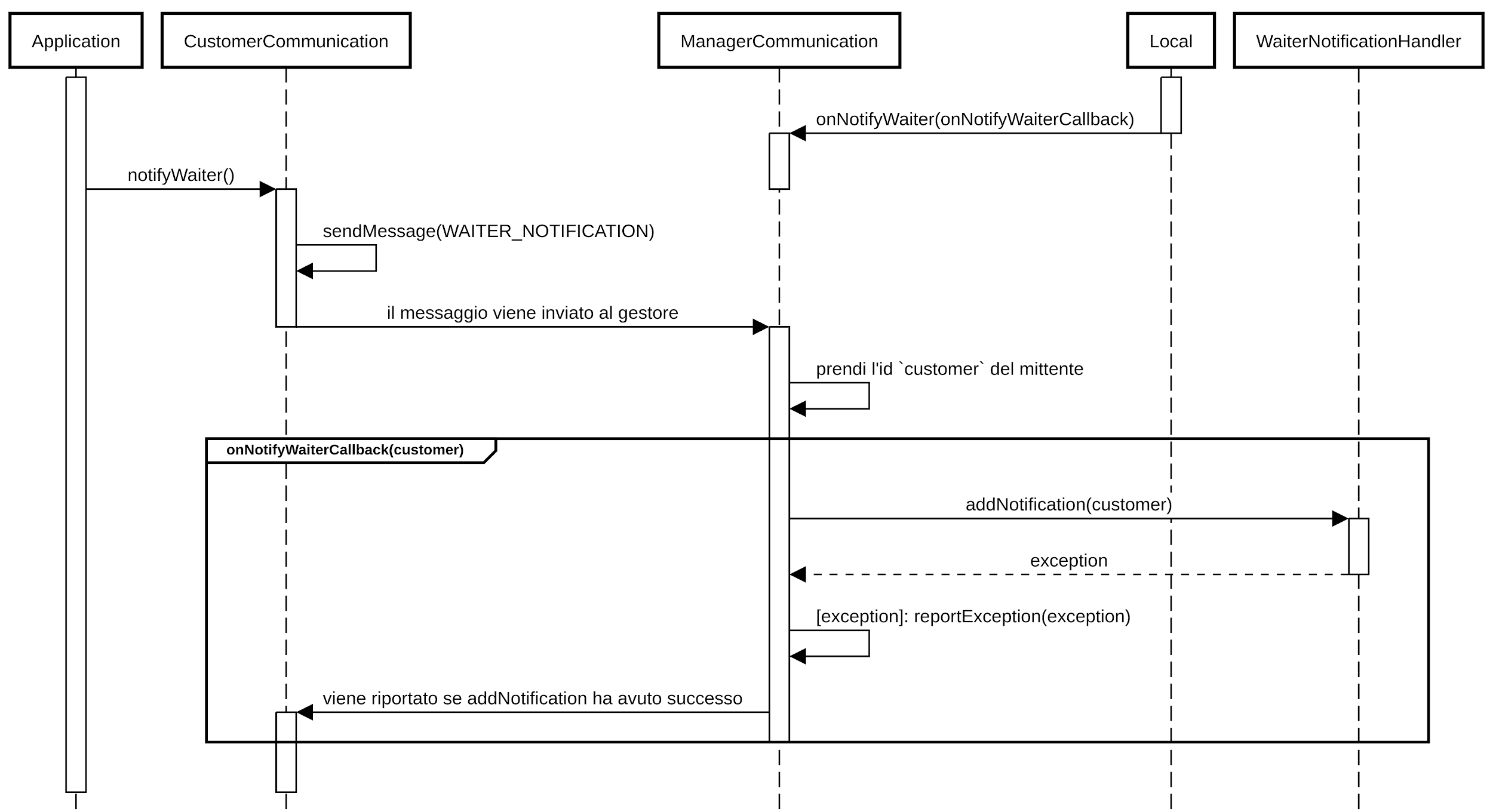


**CD-11: Immetti numero tavolo**



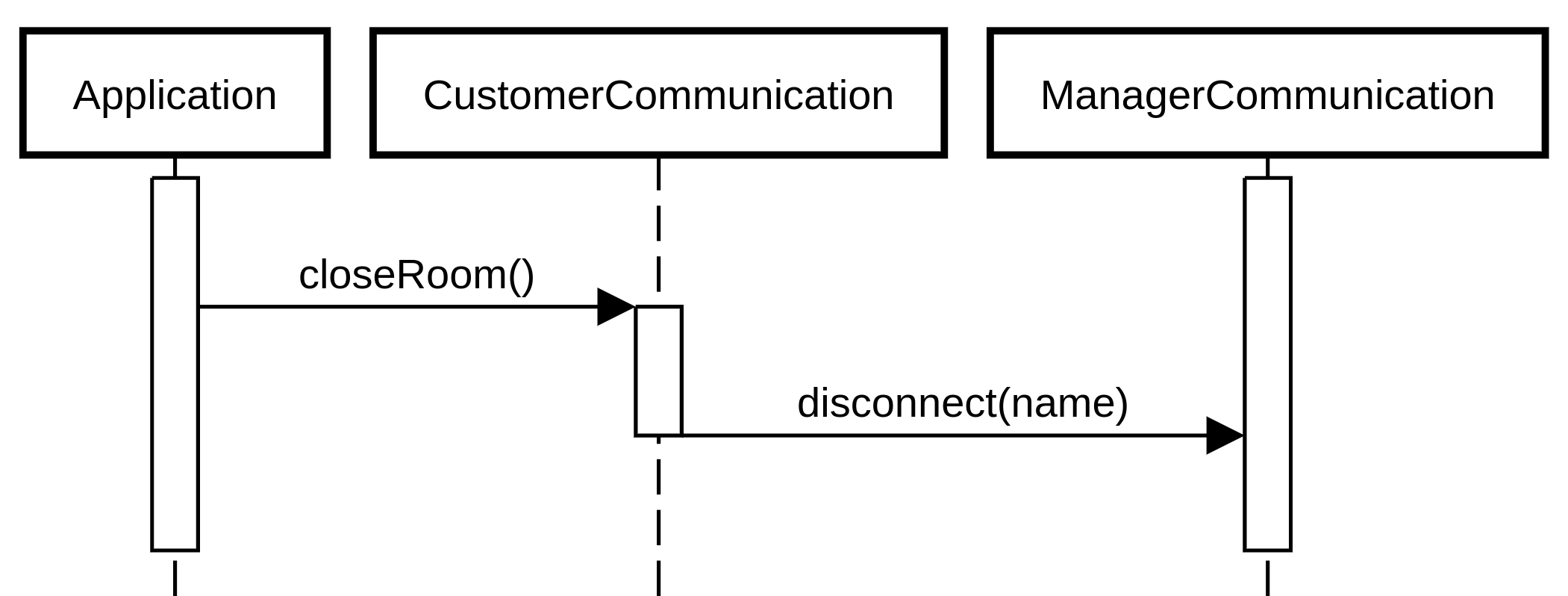
Classi usate ma non specificate: Table, ManagerTable, Message

**CD-12: Chiama cameriere**

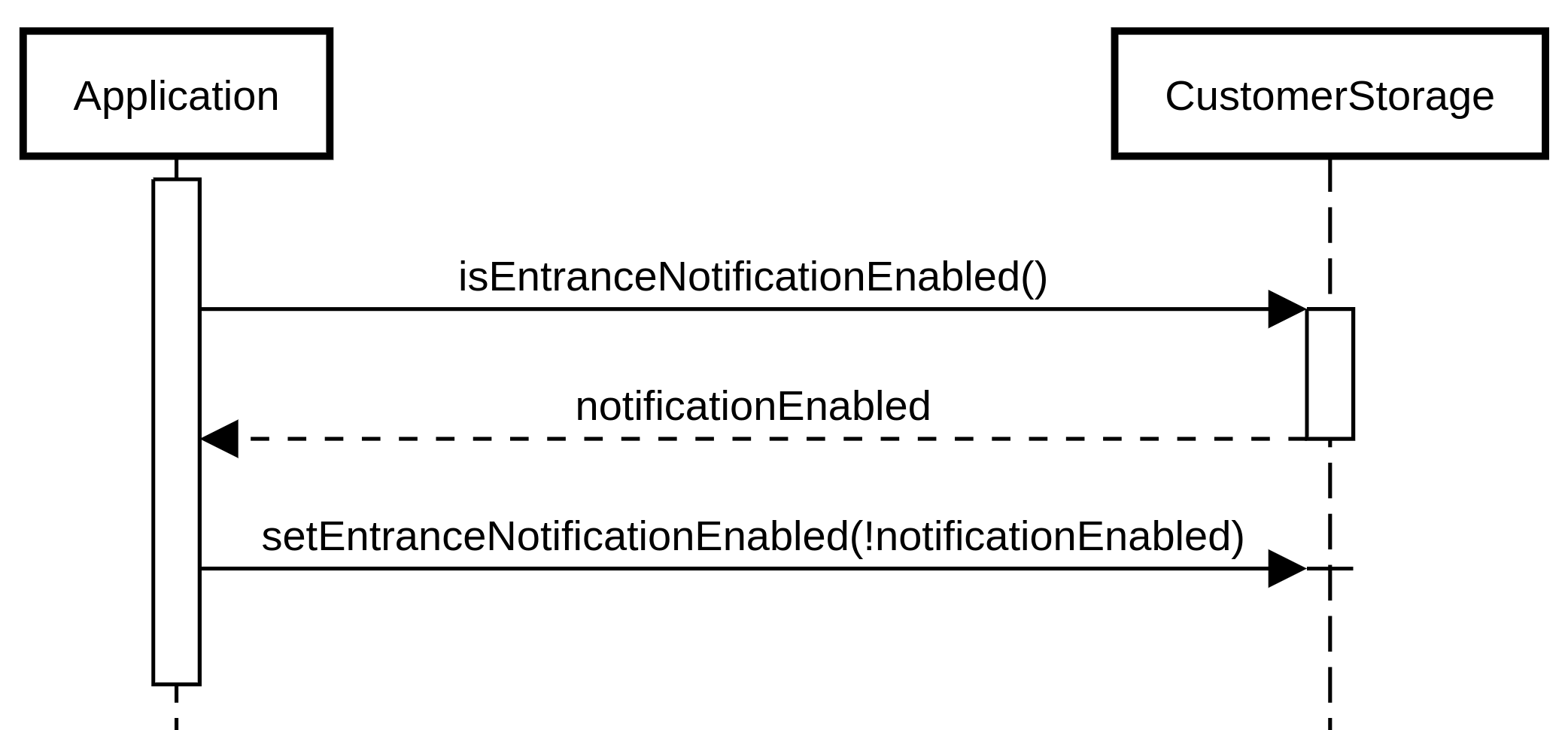


Classi usate ma non specificate: WaiterNotification, Message

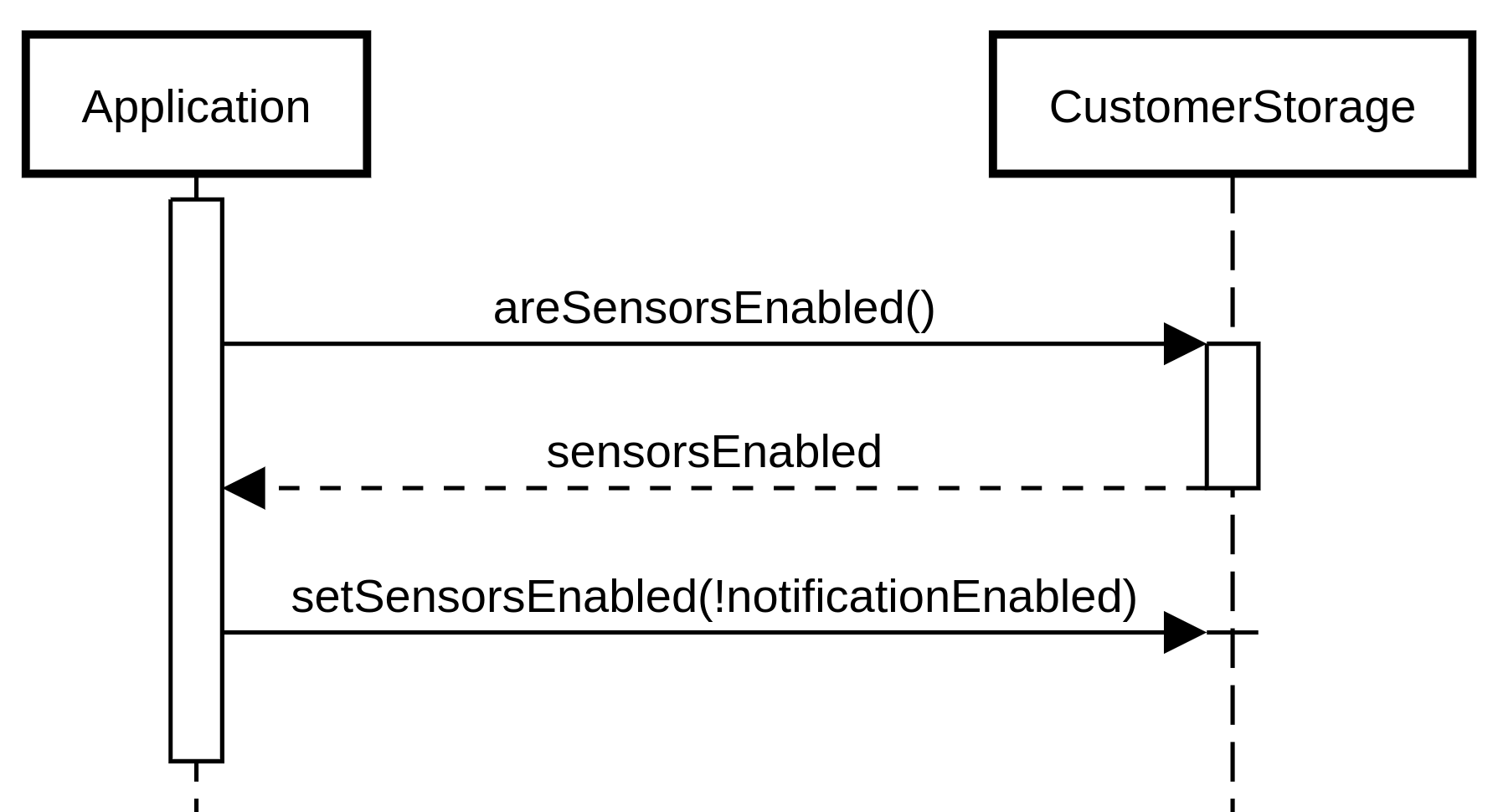
**CD-13: Esci**



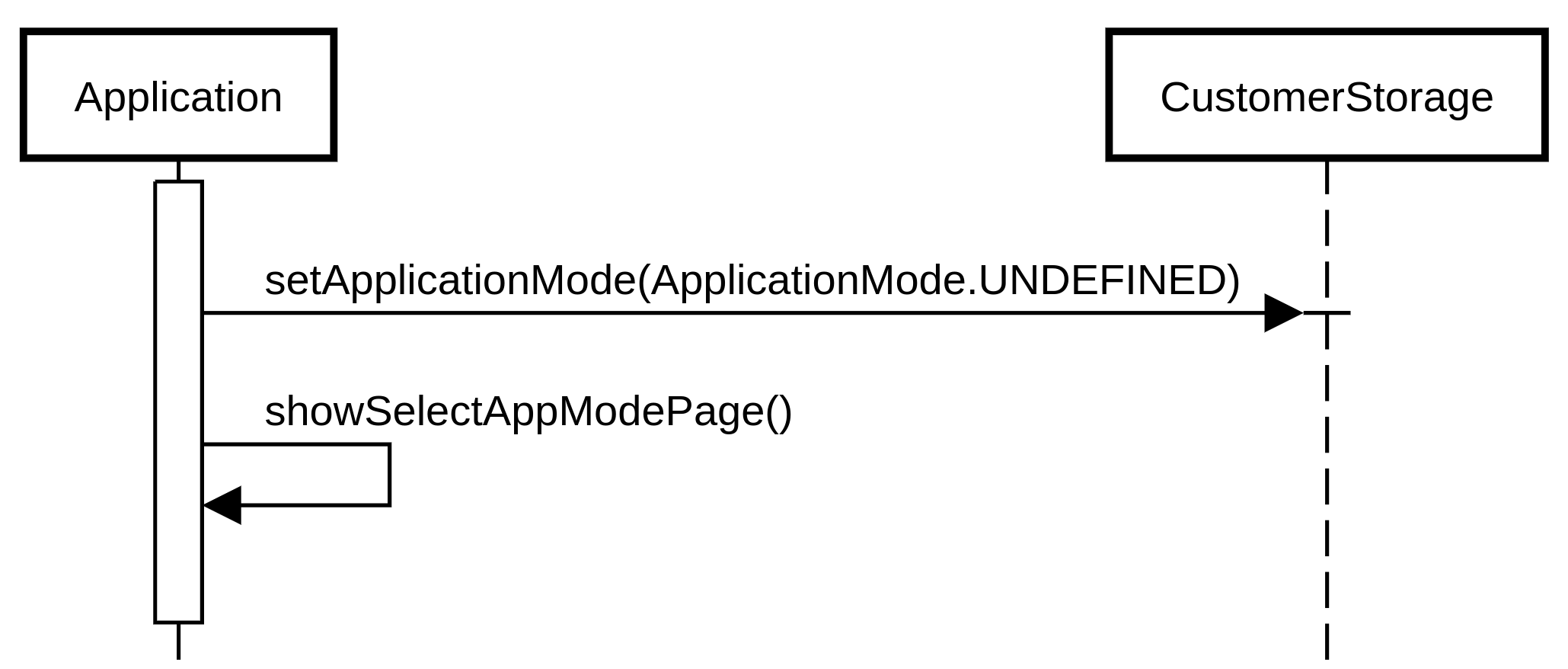
**CD-14: Attiva/disattiva notifica benvenuto**



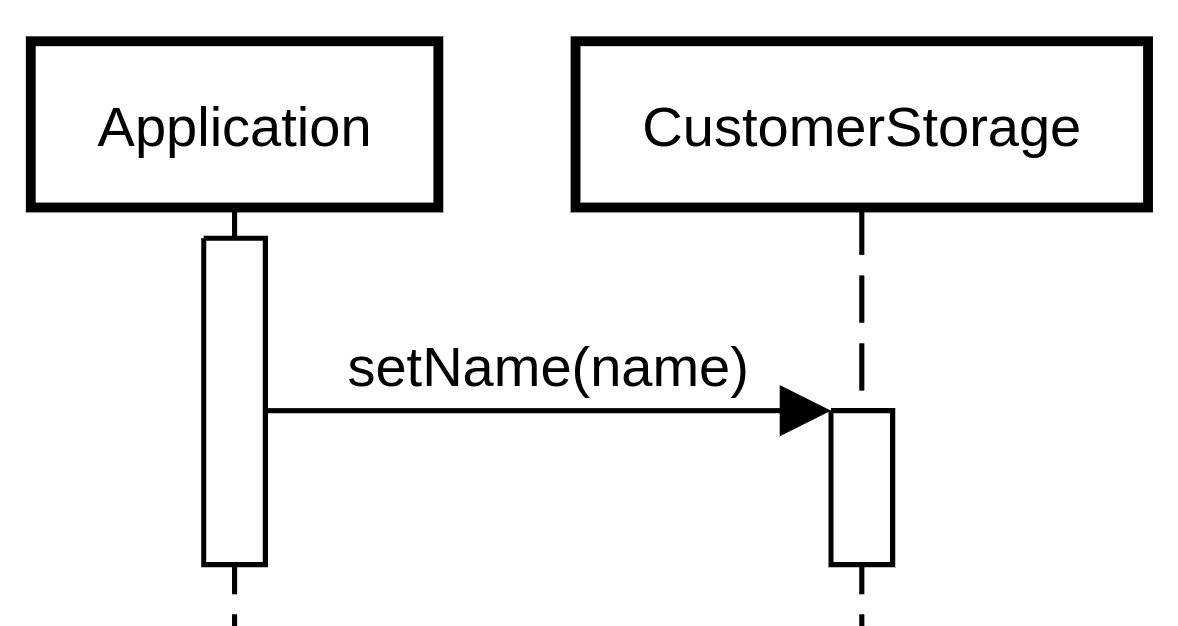
**CD-15:Attiva/disattiva uso sensori**



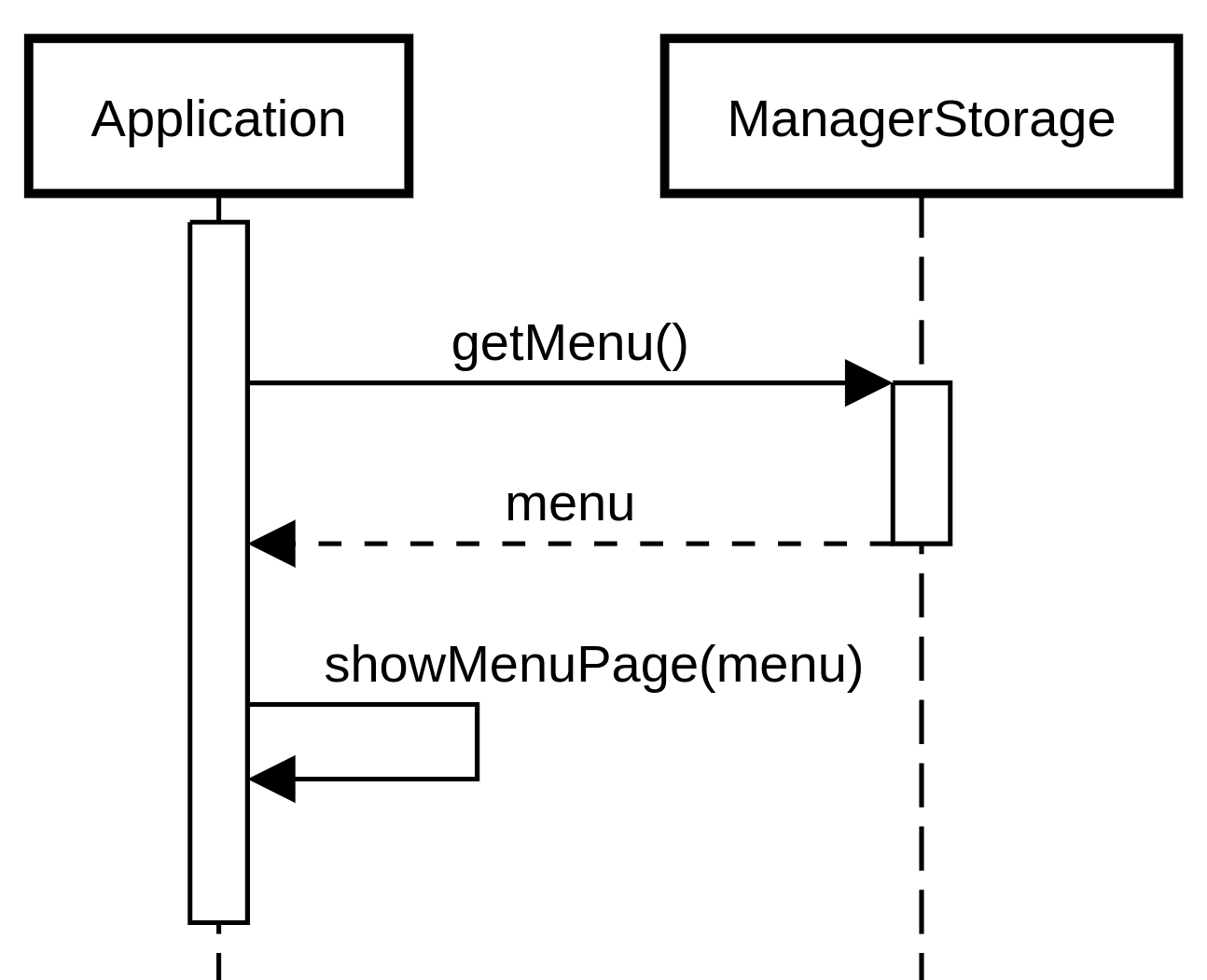
**CD-16: Esci da modalità cliente**



**CD-17: Modifica nome**

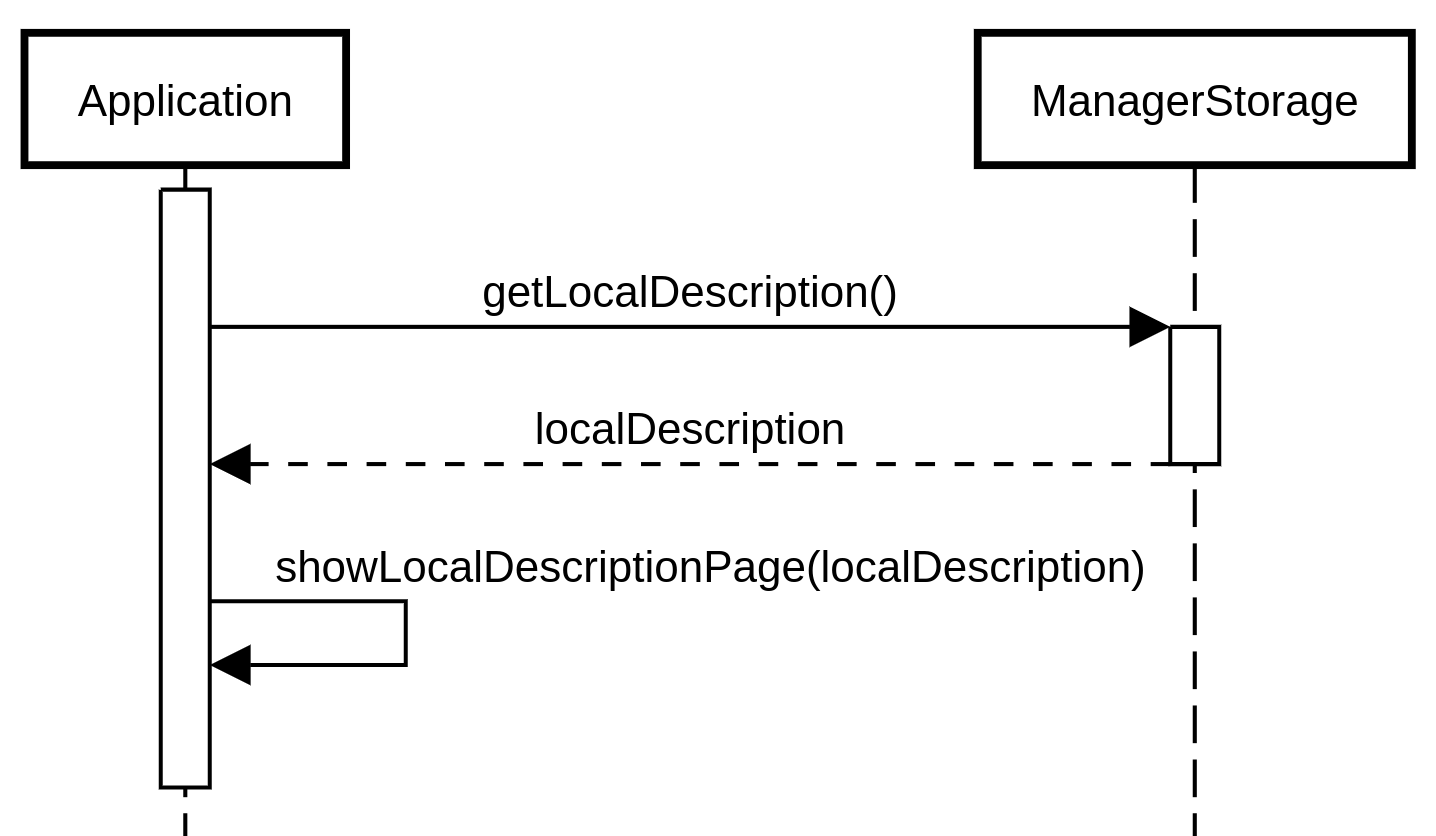


**CD-18: Visualizza menù**



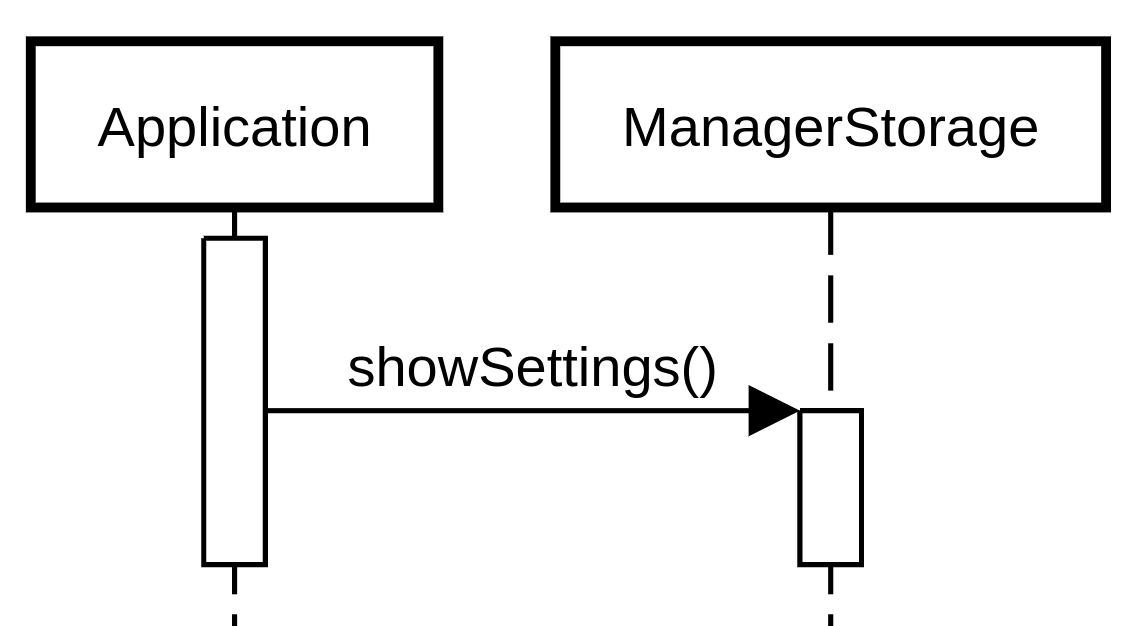
Classi usate ma non specificate: LocalDescription

**CD-19: Visualizza descrizione**

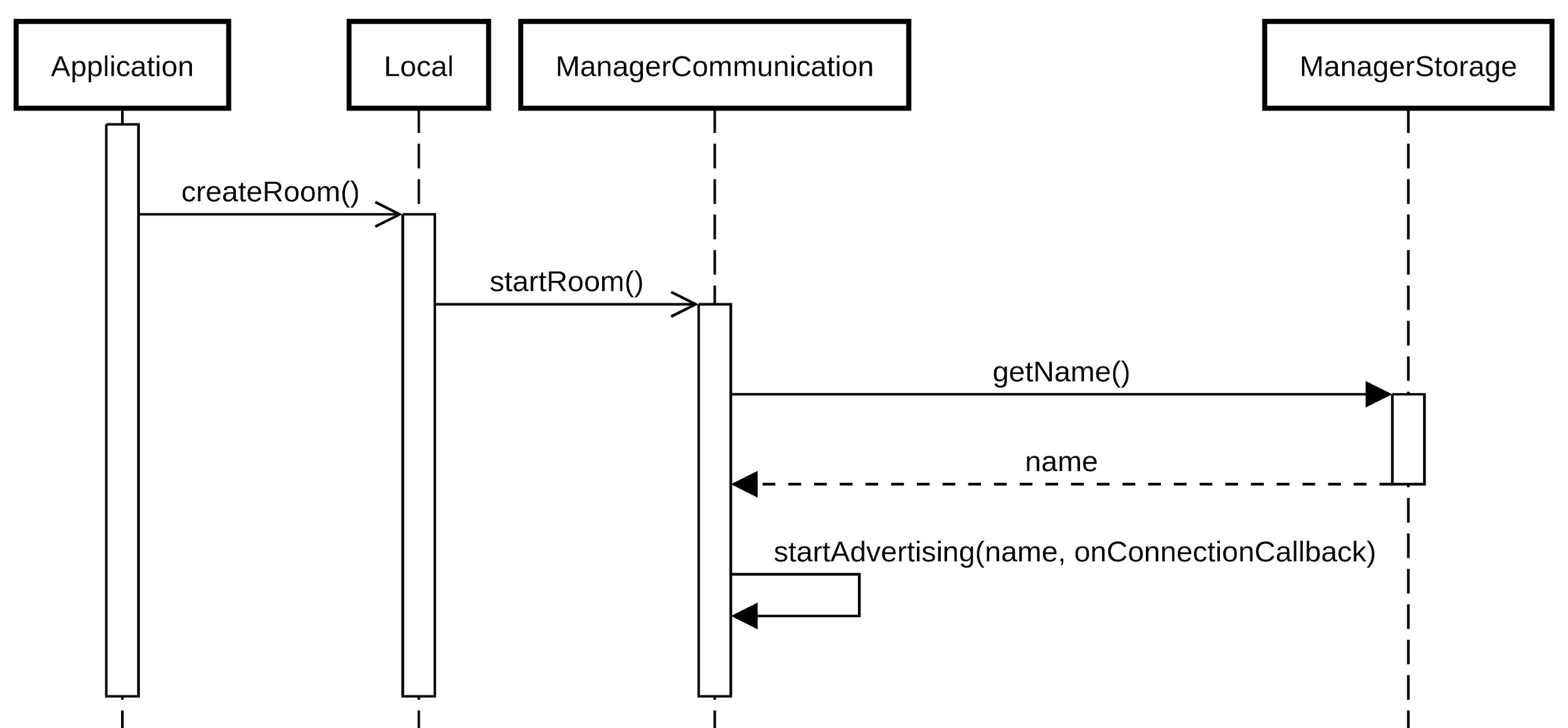


Classi usate ma non specificate: LocalDescription

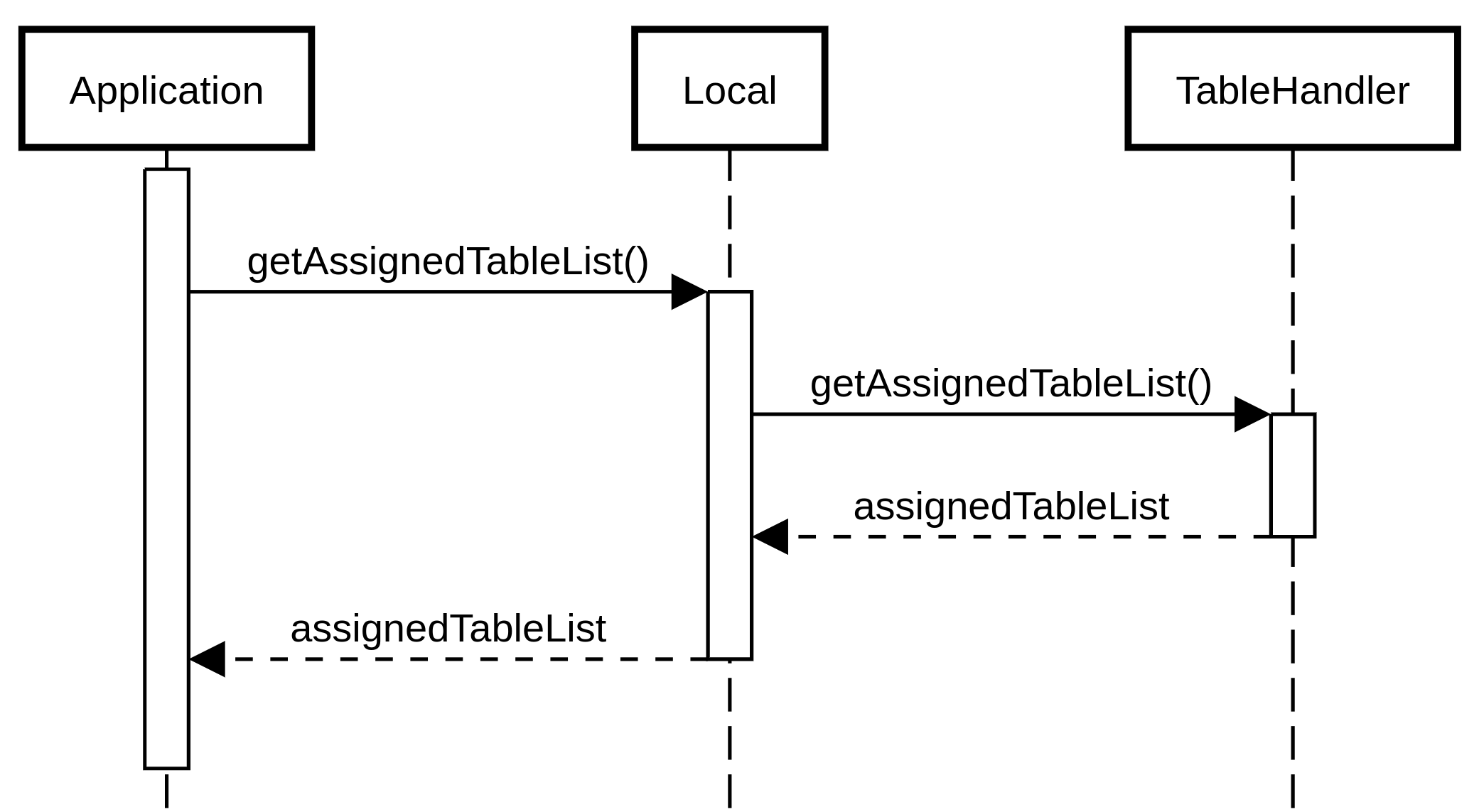
**CD-20: Impostazioni**



**CD-21: Crea stanza virtuale**

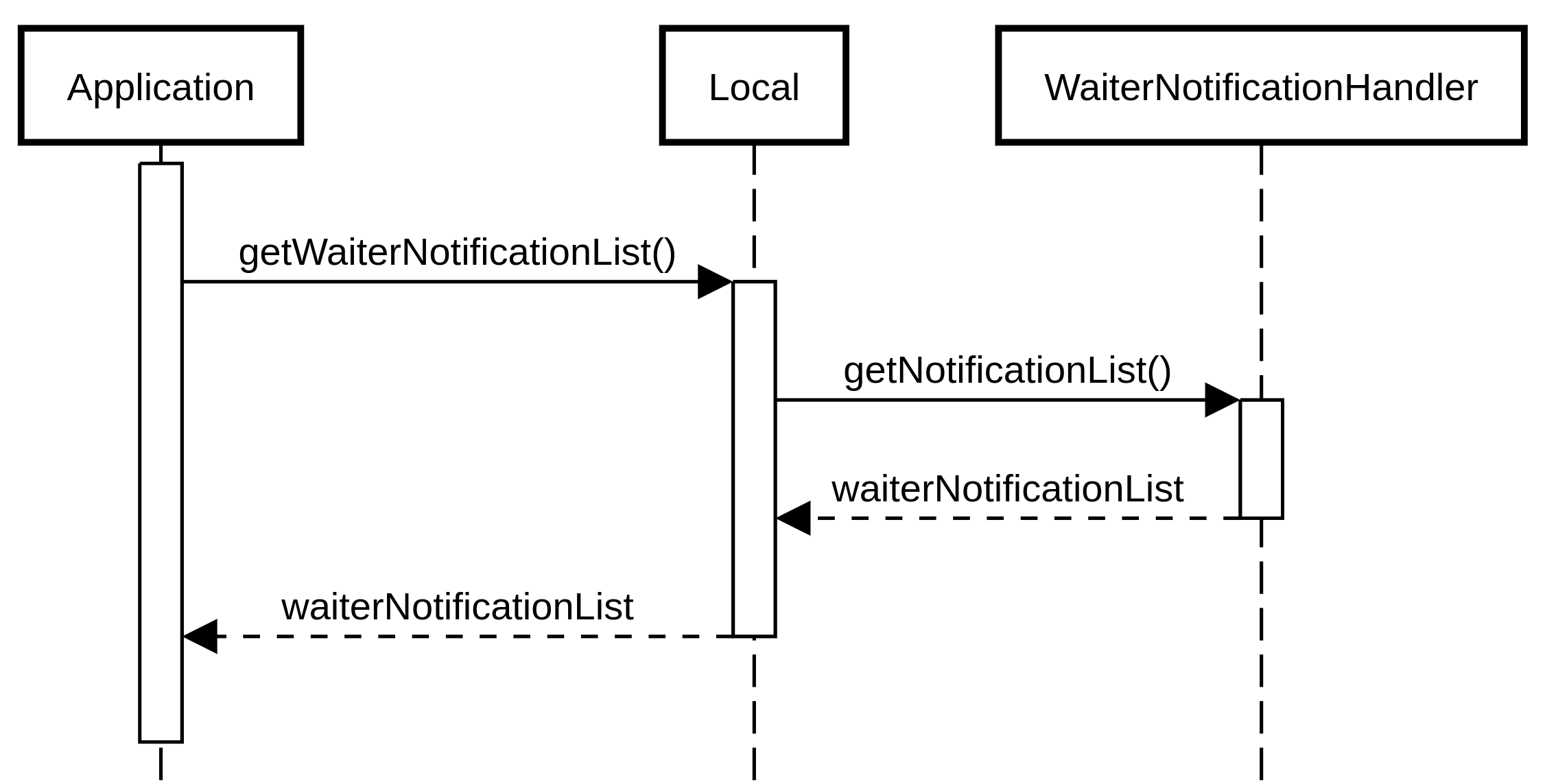


**CD-22: Lista tavoli occupati**



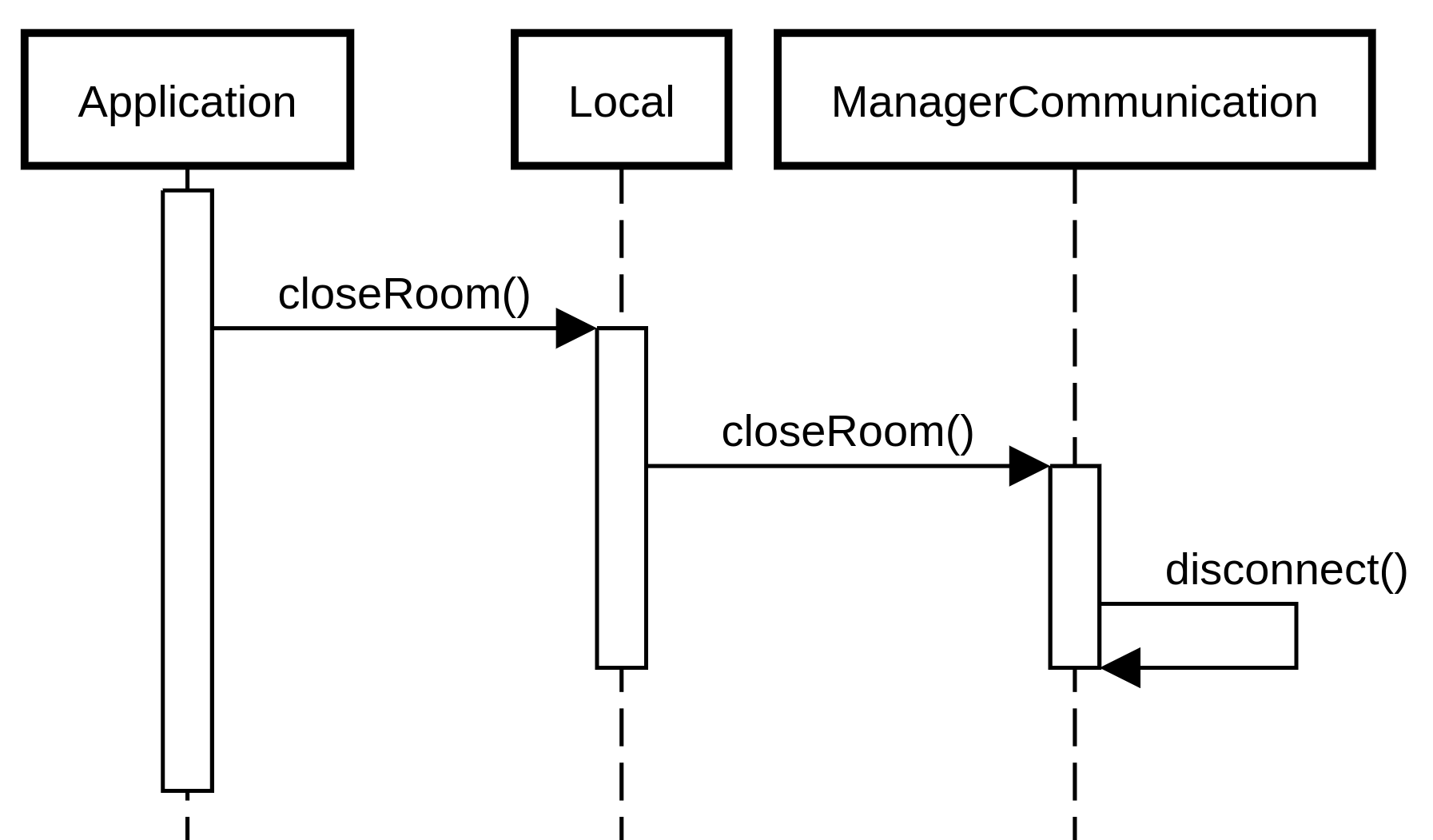
Classi usate ma non specificate: Table, ManagerTable

**CD-23: Lista notifiche cameriere**



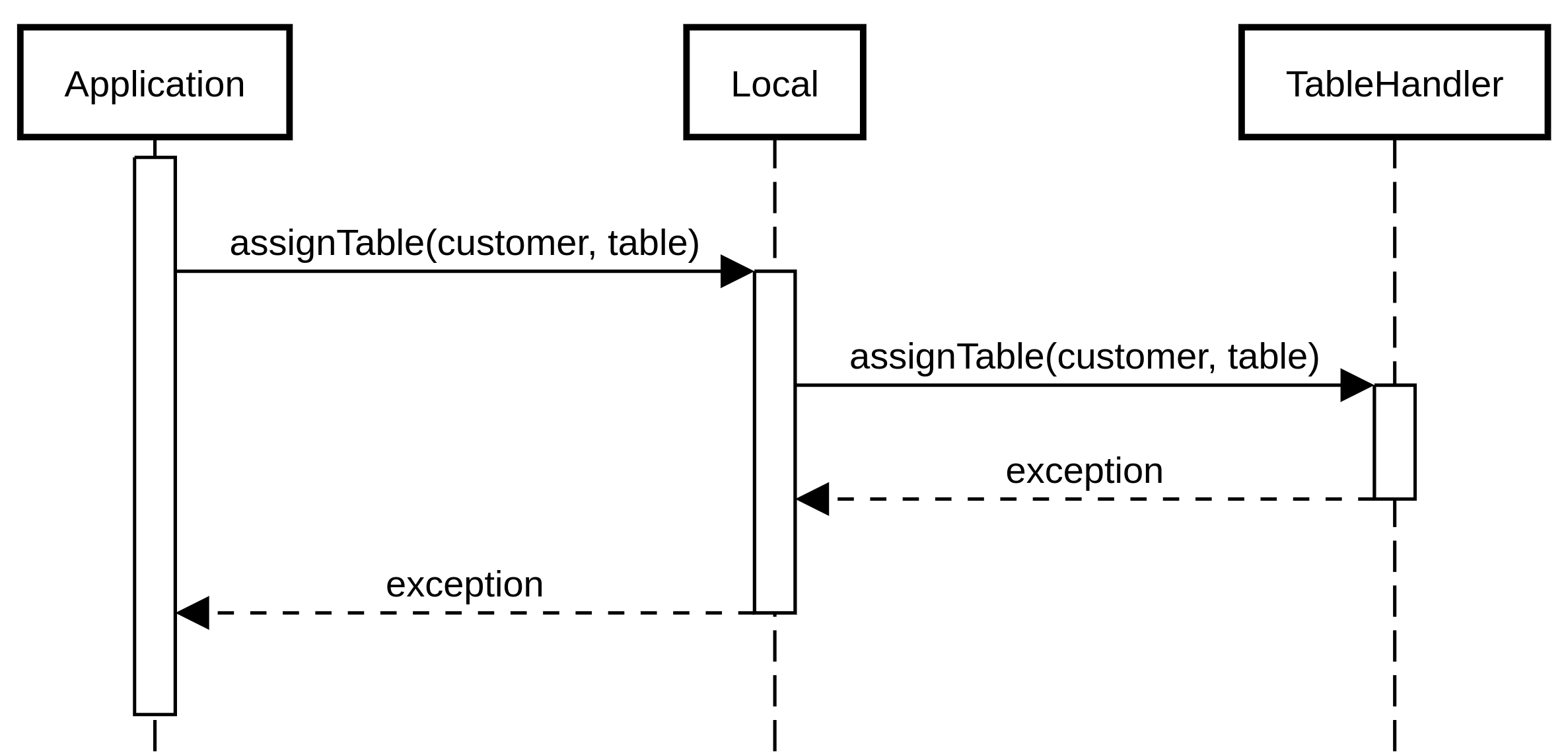
Classi usate ma non specificate : WaiterNotification

**CD-24: Chiudi stanza**



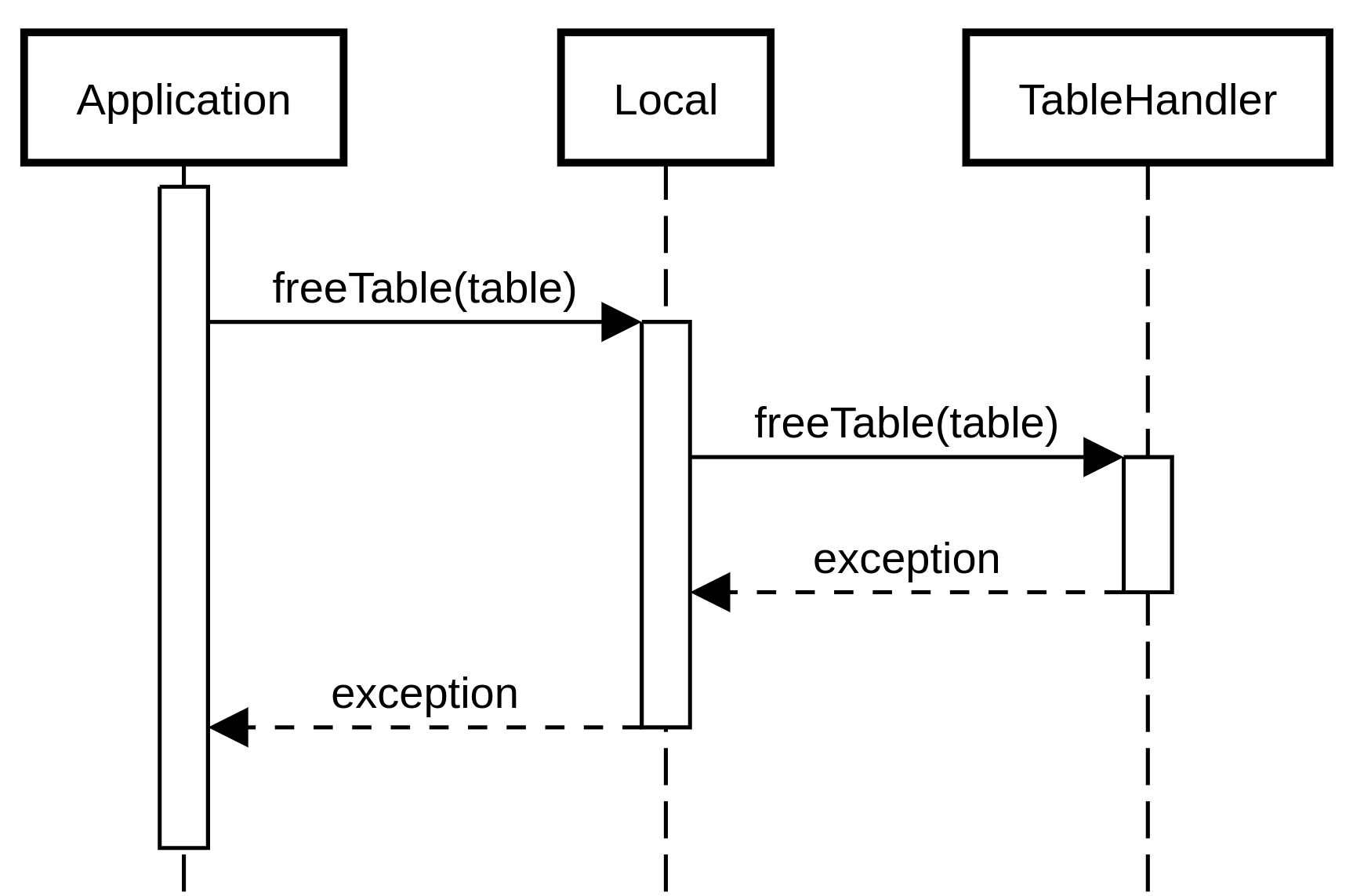
Classi usate ma non specificate: Message

**CD-25: Aggiungi tavolo**



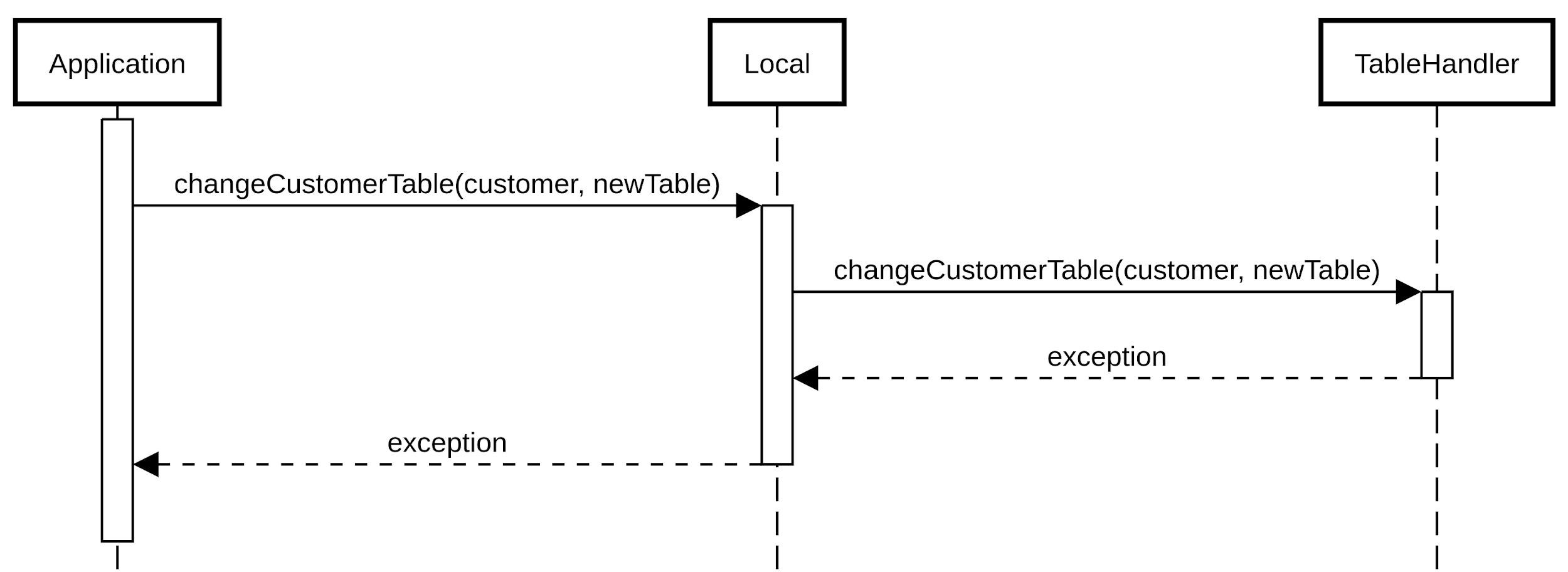
Classi usate ma non specificate: Table, ManagerTable

**CD-26: Rimuovi tavolo**



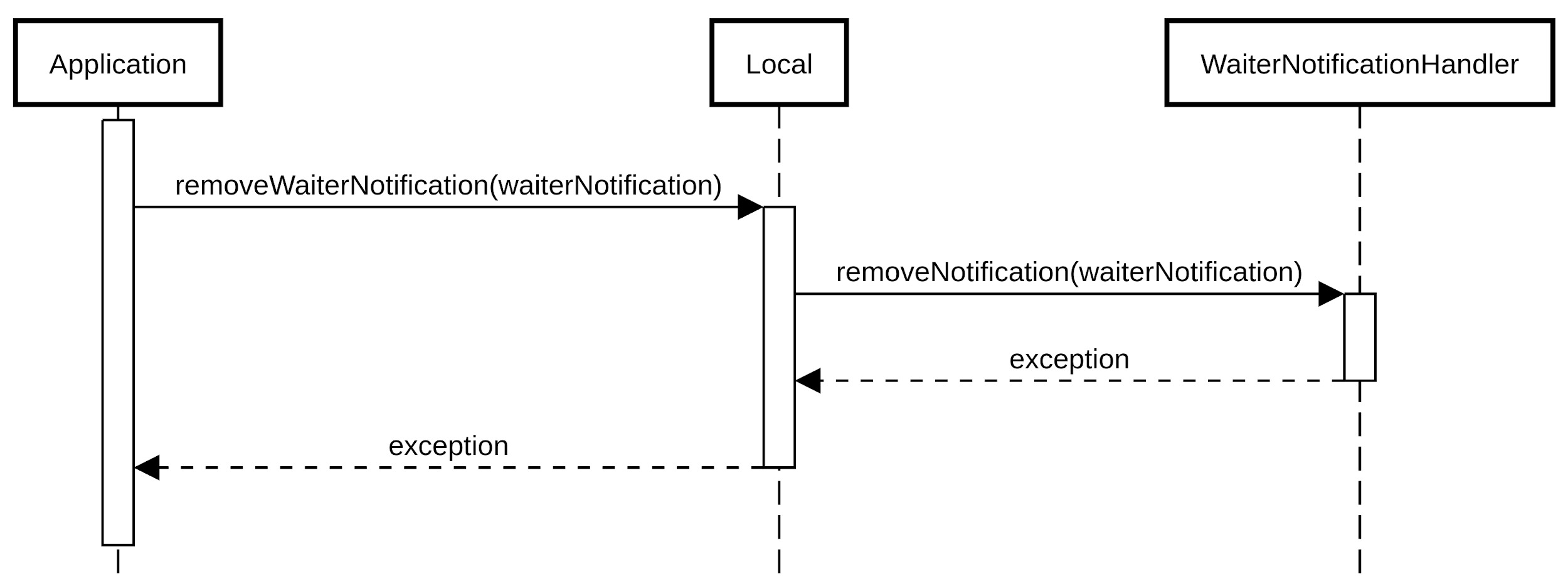
Classi usate ma non specificate: Table, ManagerTable

**CD-27: Modifica tavolo**



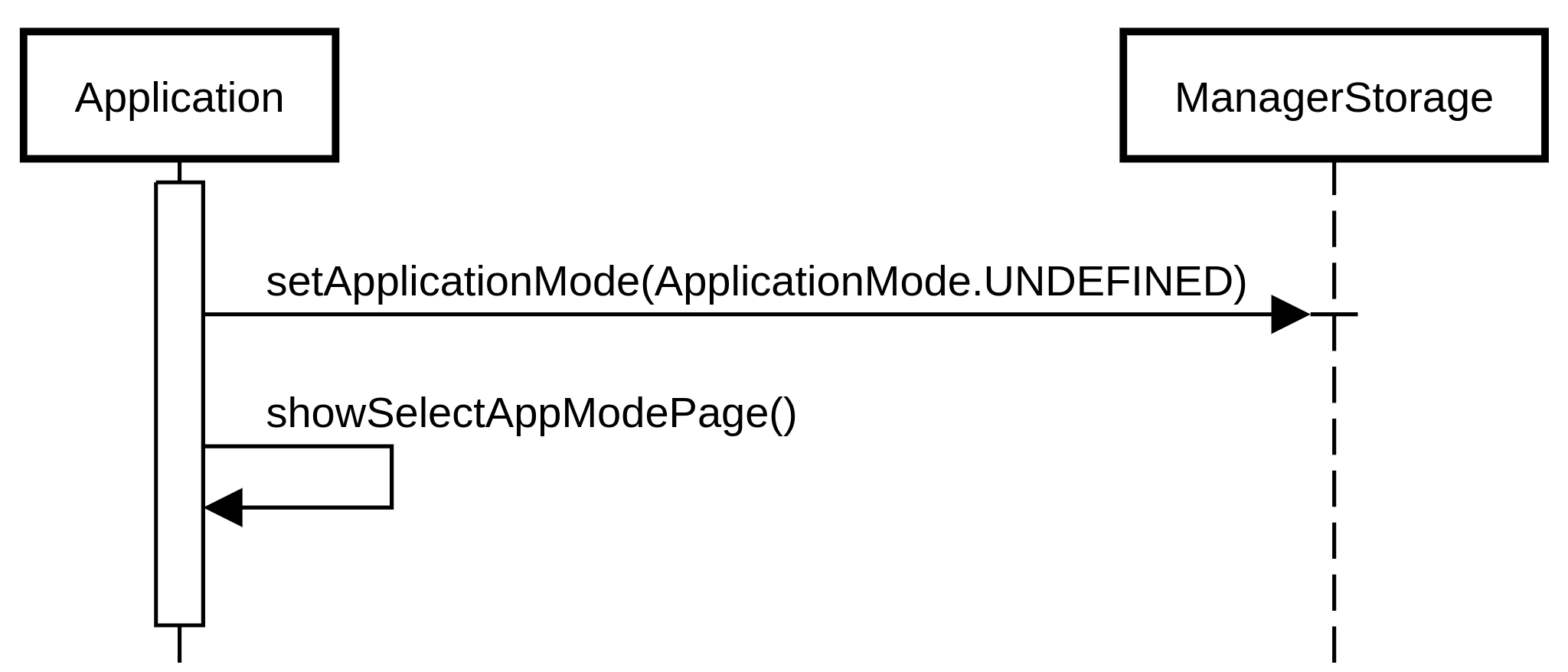
Classi usate ma non specificate: Table, ManagerTable

**CD-28: Prendi in carico**



Classi usate ma non specificate: WaiterNotification

**CD-29: Esci da modalità gestore**



Classi usate ma non specificate: ApplicationMode