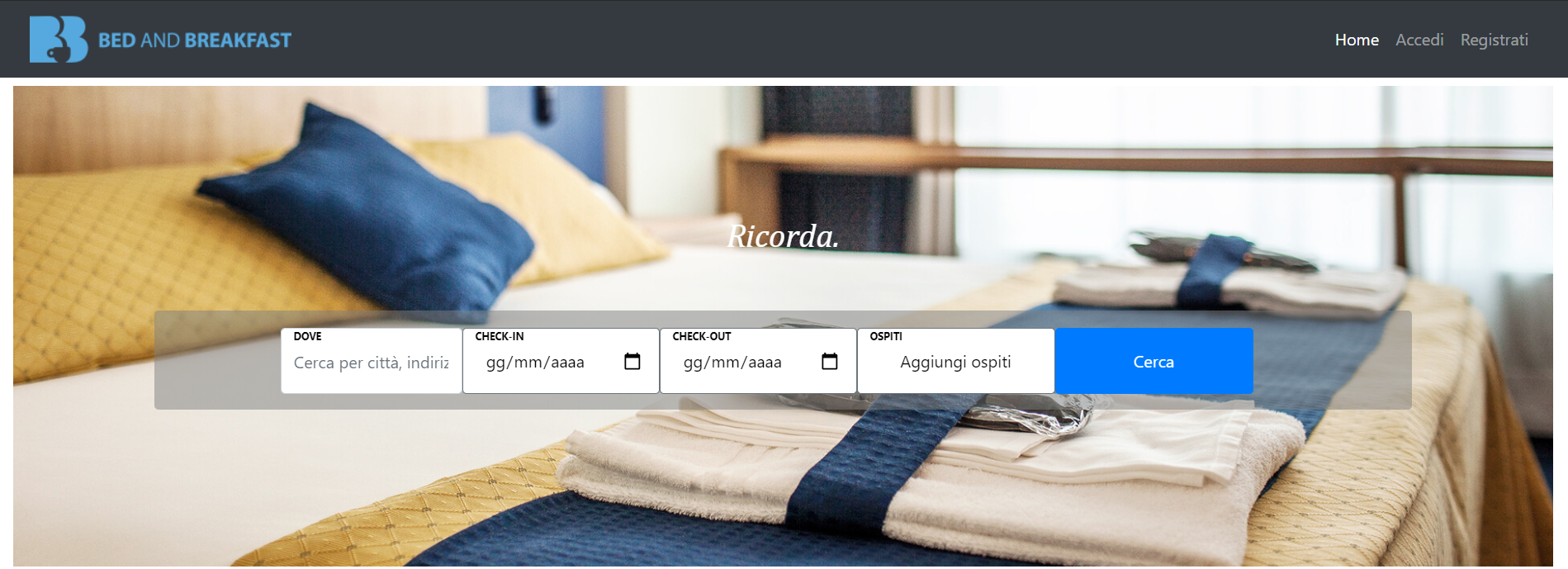
Progetto Programmazione Web e Mobile:  
B&B Web App

* Salvatore Apicella (0670120)
* Girolamo Nobile (0672406)
* Tommaso Scalici (0665134)





Indice

[Progetto Programmazione Web e Mobile: B&B Web App 1](#_Toc43849097)

[Indice 2](#_Toc43849098)

[Fase di Analisi - Introduzione 3](#_Toc43849099)

[Analisi dei Requisiti 4](#_Toc43849100)

[Glossario 6](#_Toc43849101)

[Product Backlog 7](#_Toc43849102)

[Fase di Progettazione - Architettura progettuale 9](#_Toc43849103)

[Diagrammi 10](#_Toc43849104)

[Diagramma dei casi d’uso 10](#_Toc43849105)

[Diagrammi delle classi 11](#_Toc43849106)

[Diagrammi di sequenza 13](#_Toc43849107)

[Fase di Sviluppo 21](#_Toc43849108)

[MVC (Model, View, Controller) 23](#_Toc43849109)

[Utilities e scripts di supporto 31](#_Toc43849110)

[Note finali 34](#_Toc43849111)

[Rilascio 34](#_Toc43849112)

[Note di pre-rilascio 34](#_Toc43849113)

[Software utilizzati per lo sviluppo 35](#_Toc43849114)

Fase di Analisi - Introduzione

Progettazione e sviluppo di un software di gestione per strutture turistiche come case vacanza e B&B. Il software dovrà gestire le fasi principali della prenotazione dell’appartamento o della stanza, relativamente ad entrambe le parti (cliente e proprietario).  
  
Il sistema dovrà permettere ad utente di ricercare gli appartamenti scegliendo una destinazione e selezionando opportuni filtri (durata del soggiorno, caratteristiche dell’appartamento, visualizzazione di foto e altri dettagli).  
Se l’utente invece è (anche) un host, dovrà essere in grado di gestire i propri appartamenti che mette in affitto, e dovrà avere a disposizione un’interfaccia che gli permetta di caricare i dati necessari alla pubblicazione, eventuali fotografie, oltre a poter visualizzare le richieste di prenotazioni, le prenotazioni confermate, uno storico prenotazioni e i guadagni ottenuti fino a quel momento.  
  
Il sistema deve offrire supporto per la parte economica e legale relativa all’affitto della casa o della stanza in B&B, come ad esempio la gestione del pagamento della tassa di soggiorno (che varia a seconda dell’ospite e della sua età) che può essere pagata all’atto della prenotazione oppure in loco.  
Il proprietario ha l’obbligo di comunicare alla Questura la presenza degli ospiti inviando i dati anagrafici e le foto dei documenti di riconoscimento, per questa parte si richiede che il software fornisca al proprietario la funzionalità adeguata a inserire i dati e inviarli al sistema della Questura.  
  
Il proprietario ha poi l’obbligo, ogni tre mesi, di rendicontare all’ufficio del turismo della città in cui fornisce il servizio l’ammontare delle tasse di soggiorno, le generalità degli ospiti ed il periodo, si richiede anche per questa parte il supporto del software. Inoltre, si preveda che in ogni momento il proprietario possa verificare i guadagni ottenuti. Infine, per legge non può essere affittato un appartamento alla stessa persona per un ammontare cumulativo annuo di 28 giorni, si richiede quindi che il sistema effettui questo controllo.

Analisi dei Requisiti

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisiti Funzionali** | **MoSCoW** |
| 0 | L’utente deve potersi registrare sulla piattaforma e successivamente poter accedere con le proprie credenziali e gestire il proprio profilo. | Must Have |
| 1 | Il proprietario di un appartamento deve poter vedere e gestire le prenotazioni degli utenti | Must Have |
| 2 | L’utente deve poter effettuare ricerche e applicare filtri per: destinazione, tipo di alloggio desiderato, costo per notte, periodo di soggiorno etc. | Must Have |
| 3 | All’atto della prenotazione il sistema deve inviare una mail di riepilogo al cliente e al proprietario, che può accettare o rifiutare la prenotazione. | Must Have |
| 4 | Il proprietario di un B&B deve poter inserire il suo appartamento con tutte le informazioni che possono essere necessarie per una buona pubblicizzazione dello stesso, comprese eventualmente delle fotografie. Tra le info che possono essere utili: descrizione delle forniture (tipo Wi-Fi, ascensore disabili, parcheggio, etc.) e descrizione del luogo con attenzione ai monumenti o ai luoghi di interesse turistico più vicini. | Must Have |
| 5 | Per legge non può essere affittato un appartamento alla stessa persona per un ammontare cumulativo annuo di **28 giorni**, si richiede quindi che il sistemaeffettui questo controllo. | Must Have |
| 6 | Il proprietario deve comunicare alla Questura la presenza degli ospiti inviando i dati anagrafici e le foto dei documenti di riconoscimento, per questa parte si richiede che il software fornisca al proprietario la funzionalità per inserire i dati e inviarli al sistema della Questura. | Must Have |
| 7 | Il proprietario ha l’obbligo, **ogni tre mesi**, di rendicontare all’ufficio del turismo della città in cui fornisce il servizio l’ammontare delle tasse di soggiorno, le generalità degli ospiti ed il periodo, si richiede anche per questa parte il supporto del software (*ad esempio ricorda una settimana prima della scadenza del rendiconto*) | Must Have |
| 8 | Il proprietario deve poter verificare i guadagni ottenuti. | Must Have |
| 9 | L’utente può lasciare una recensione per un **appartamento** in cui ha alloggiato. | Should Have |

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisiti Non Funzionali** | **MoSCoW** |
| 0 | I dati sensibili (come ad esempio le password) degli utenti devono essere salvati sul sistema previa cifratura per questioni di sicurezza. | Must have |
| 1 | Si utilizzerà il protocollo HTTPS per la comunicazione sicura dei dati trasferiti. | Must have |

Glossario

* **User:** Un utente iscritto sulla piattaforma. Può effettuare ricerche di appartamenti, prenotazioni e visualizzare uno storico delle prenotazioni effettuate.
* **Host:** Un utente iscritto sulla piattaforma (come **User**) e anche proprietario di uno o più appartamenti, che può pubblicare annunci di affitto per la propria casa vacanza o B&B. Può visualizzare le prenotazioni ricevute dai clienti (accettarle o rifiutarle) e lo storico dei guadagni fino a quel momento.
* **Apartment:** Un appartamento con i relativi dati (descrizione, numero posti letto, fotografie, prezzo per notte, forniture, servizi, ecc.).
* **Reservation:** Una prenotazione che il cliente effettua per un appartamento in uno specifico intervallo di date, che il proprietario può accettare o rifiutare.

Product Backlog

|  |
| --- |
| Storie Utente relative a **User** |
| *Come* ***User*** *posso fare login/logout, registrarmi e gestire i dati dell’account ed eventualmente eliminare l’account.* |
| *Come* ***User*** *posso effettuare ricerche di appartamenti, selezionando una destinazione, le date di checkin e checkout, e utilizzando i diversi filtri di ricerca.* |
| *Come* ***User*** *posso effettuare una prenotazione di un appartamento. Al completamento del processo l’utente riceverà una mail riepilogativa con le informazioni della prenotazione.* |
| *Come* ***User*** *posso scegliere se diventare anche* ***Host****.* |

|  |
| --- |
| Storie Utente relative a **Host** |
| *Come* ***Host*** *posso pubblicare annunci per* ***Apartment*** *da affittare, e caricare sul sistema tutti i dati, le descrizioni e fotografie necessarie per il completamento della pubblicazione.* |
| *Come* ***Host*** *posso gestire le* ***Reservation*** *in corso e visualizzare lo storico di quelle passate.* |
| *Come* ***Host*** *posso decidere di accettare o rifiutare le prenotazioni che sono ancora in uno stato sospeso.* |
| *Come* ***Host*** *posso avere a disposizione un’interfaccia che mi permetta di inviare alla Questura i dati richiesti del cliente.* |
| *Come* ***Host*** *posso avere a disposizione un’interfaccia che mi permetta di rendicontare le tasse di soggiorno all’ufficio del turismo.* |
| *Come* ***Host*** *posso visualizzare i guadagni ottenuti.* |

Fase di Progettazione - Architettura progettuale

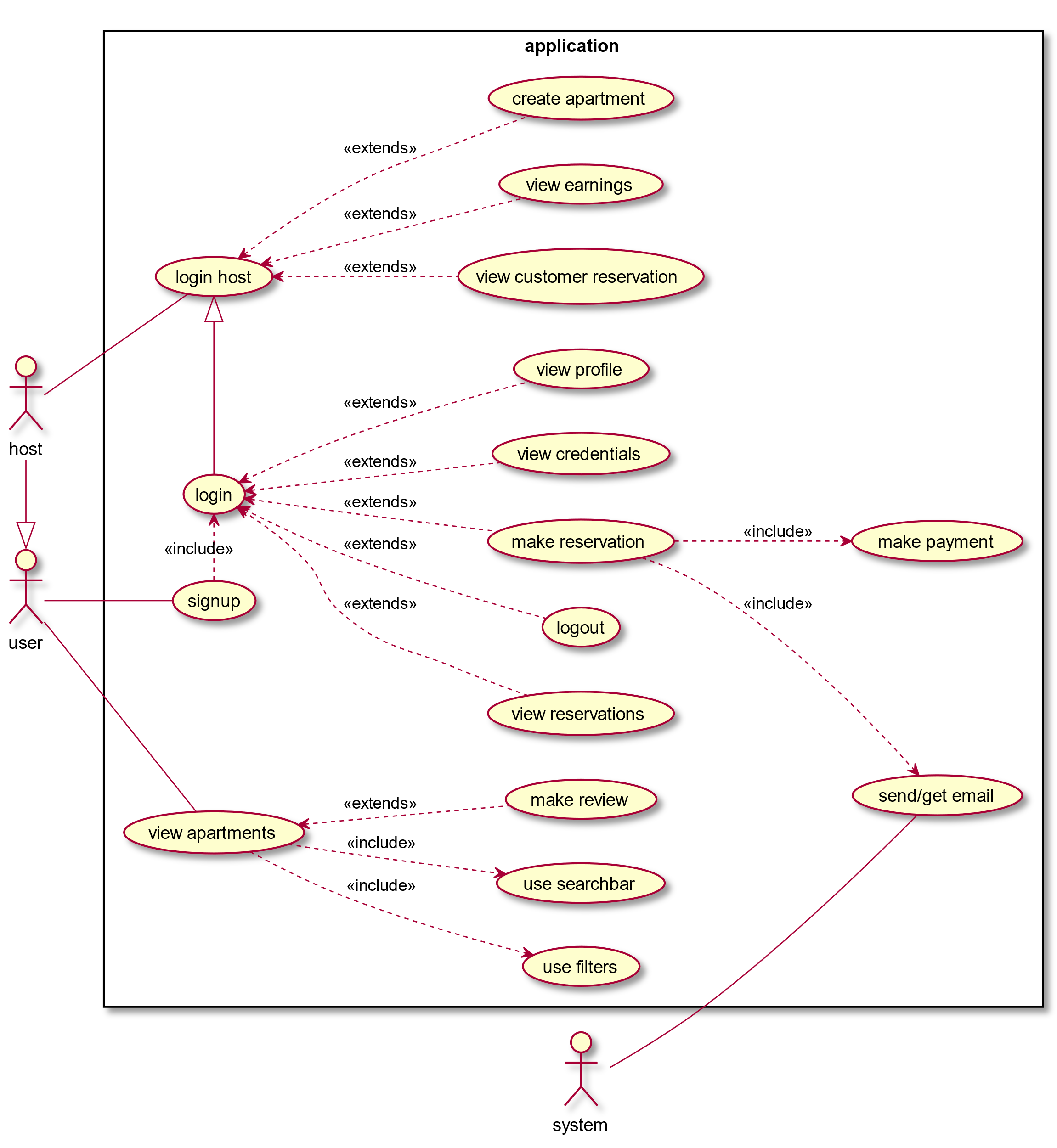
Per questo progetto si è deciso di realizzare una Web Application, utilizzando come stack architetturale lato back-end le seguenti tecnologie: Node.js, Express e MongoDB. Per il front-end si è deciso di utilizzare EJS come template engine ed HTML5+CSS3+JS con Bootstrap e jQuery.

Per quanto riguarda il design architetturale abbiamo deciso di utilizzare il pattern MVC.

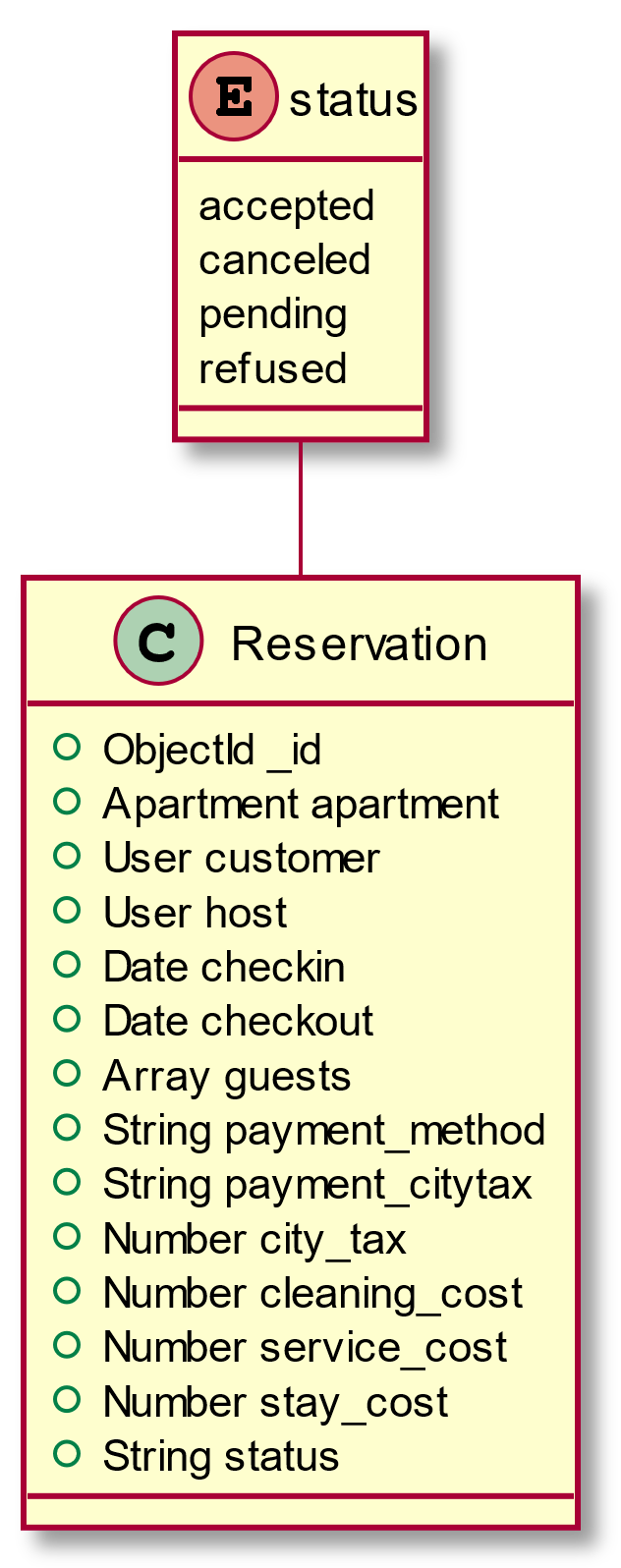
La web application verrà sviluppata come SPA (Single Page Application), quindi tutte le risorse principali (HTML, JS, CSS) verranno caricate sul browser con un unico caricamento di pagina, mentre le altre risorse verranno caricate dinamicamente su richiesta utilizzando la tecnica AJAX.

Diagrammi

Diagramma dei casi d’uso



Diagrammi delle classi

Apartment Reservation User

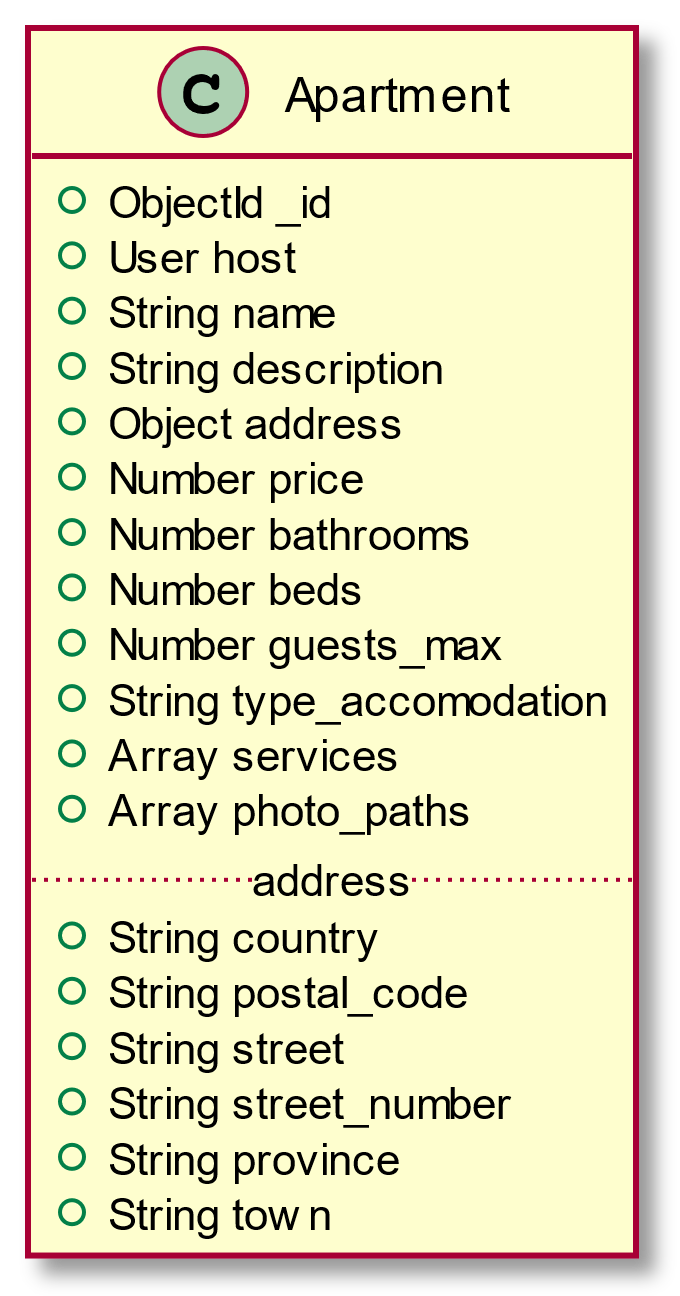
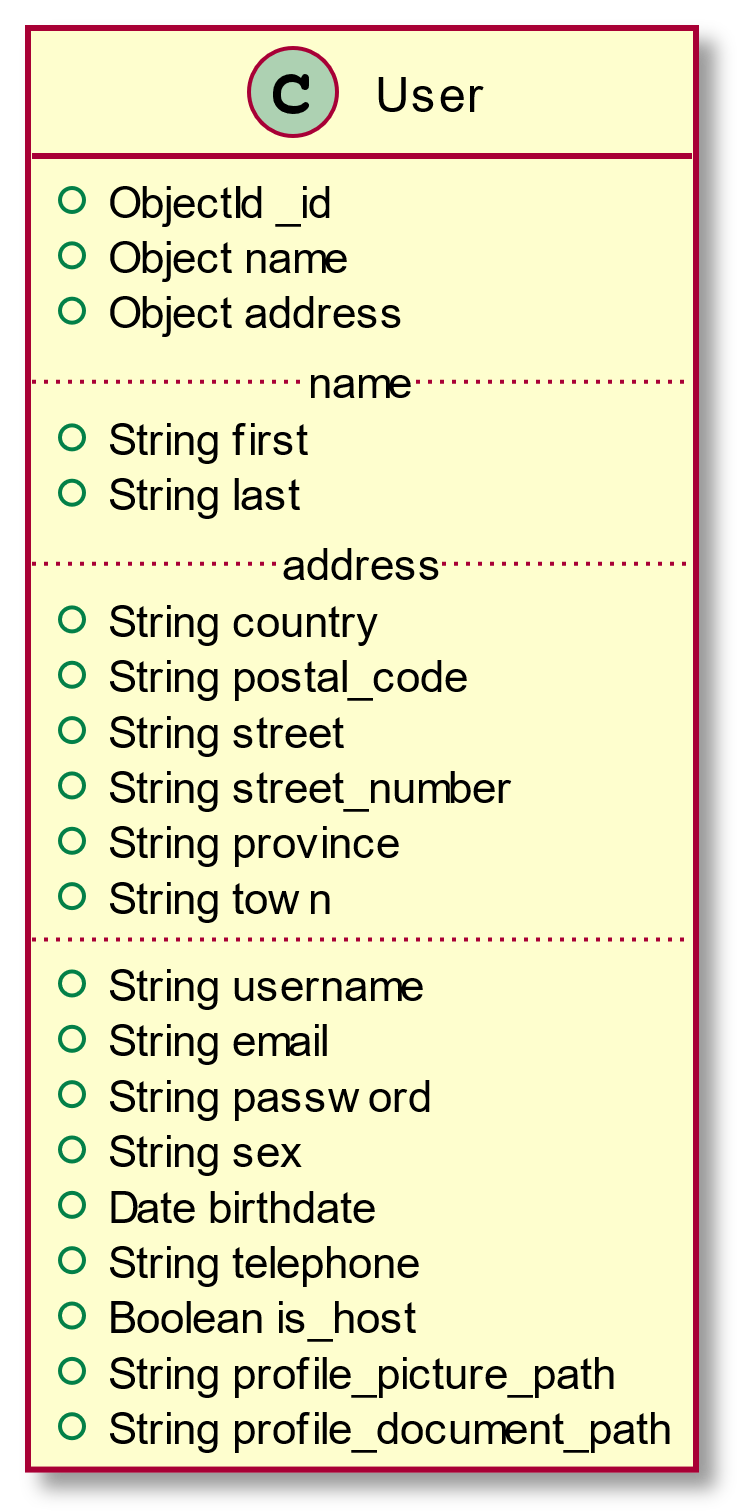
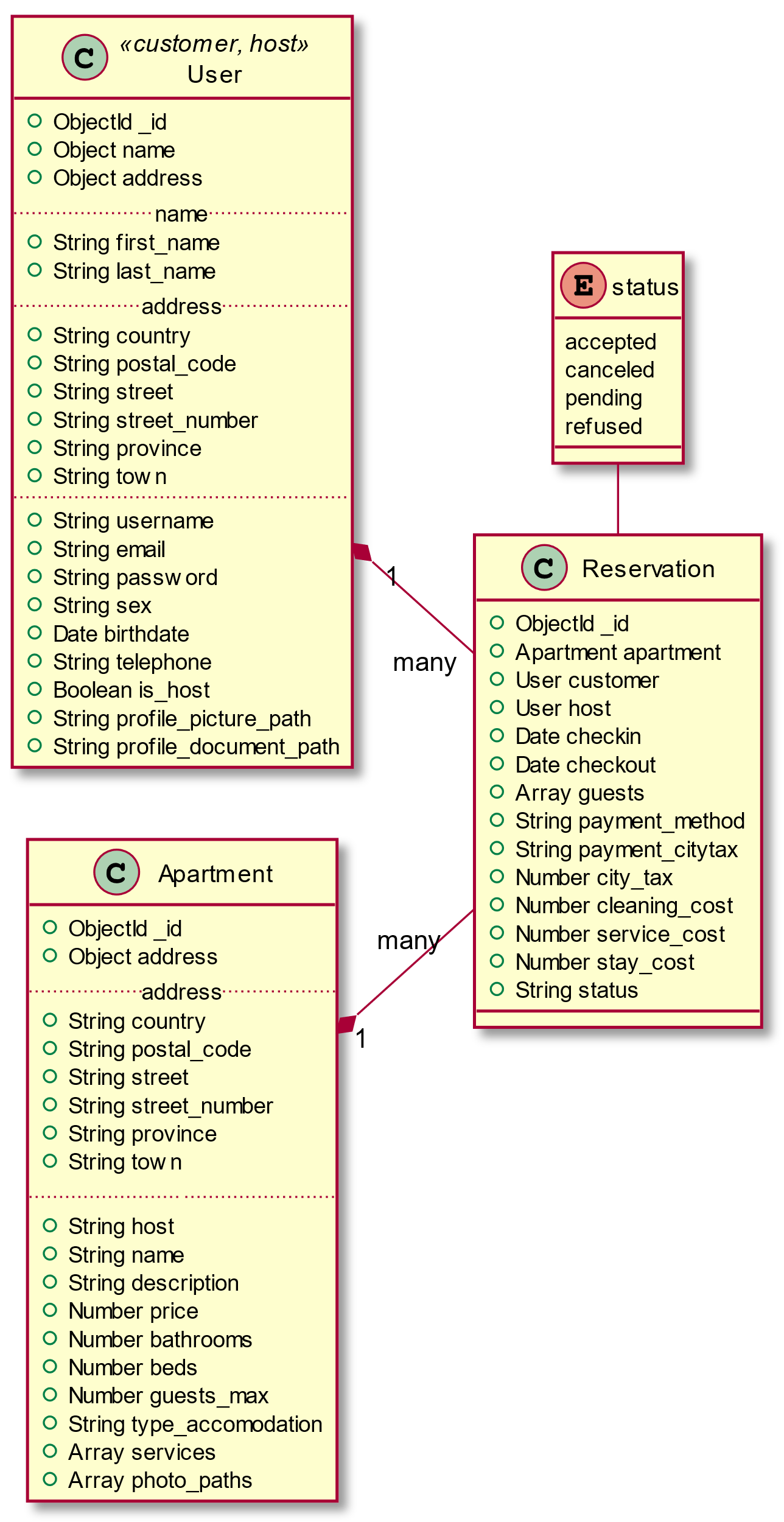
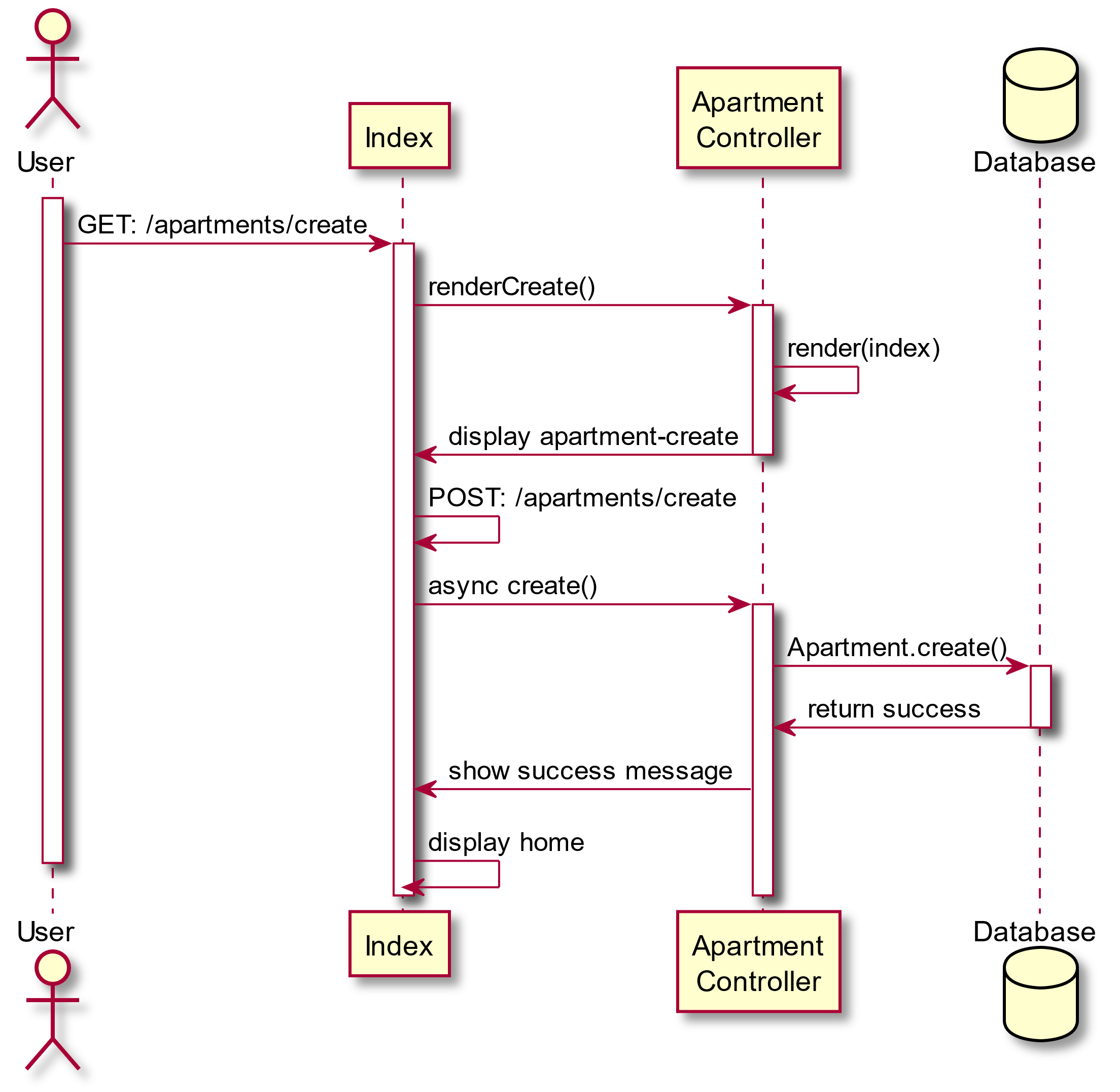


Diagramma delle classi completo

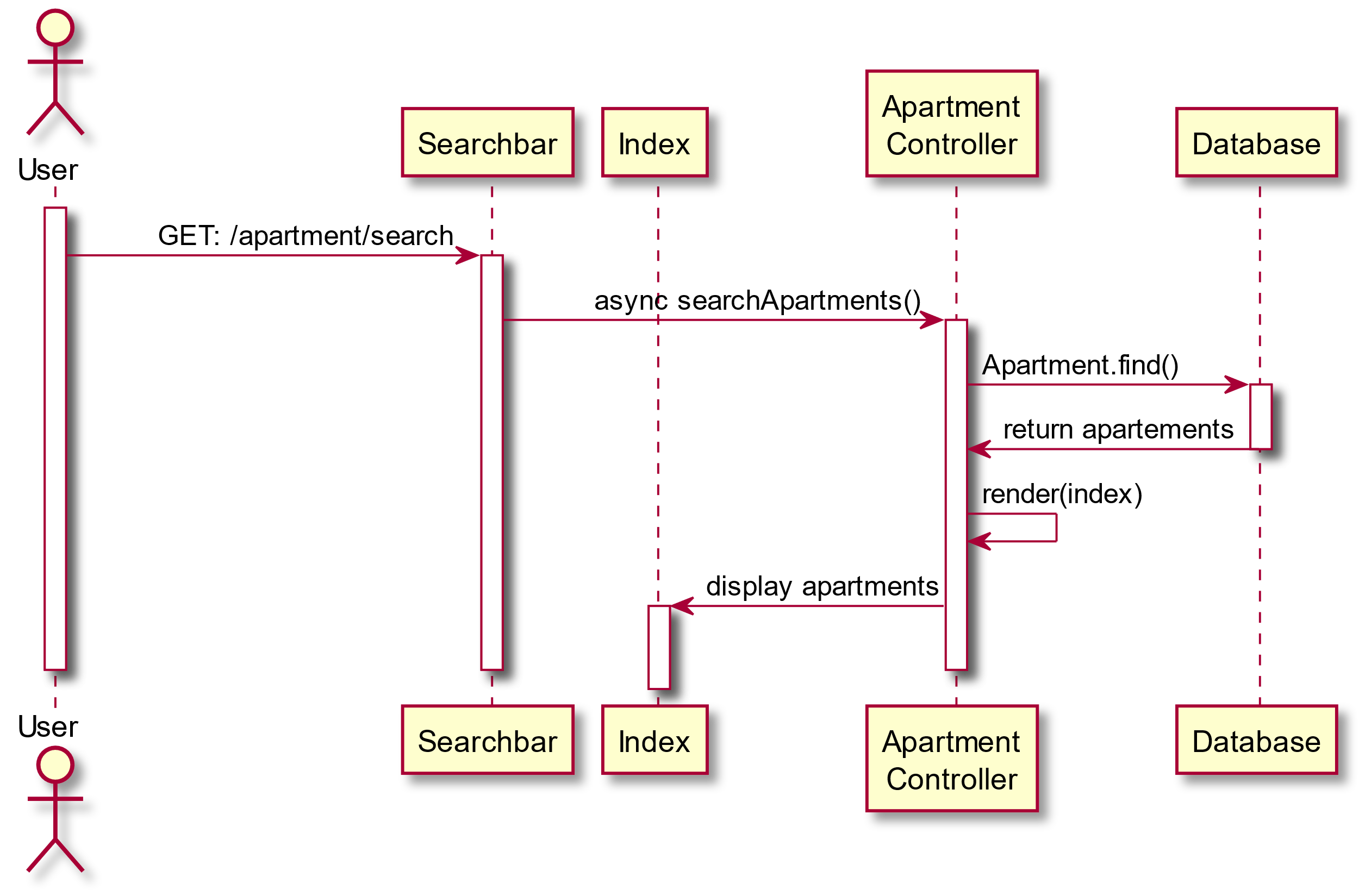


Diagrammi di sequenza

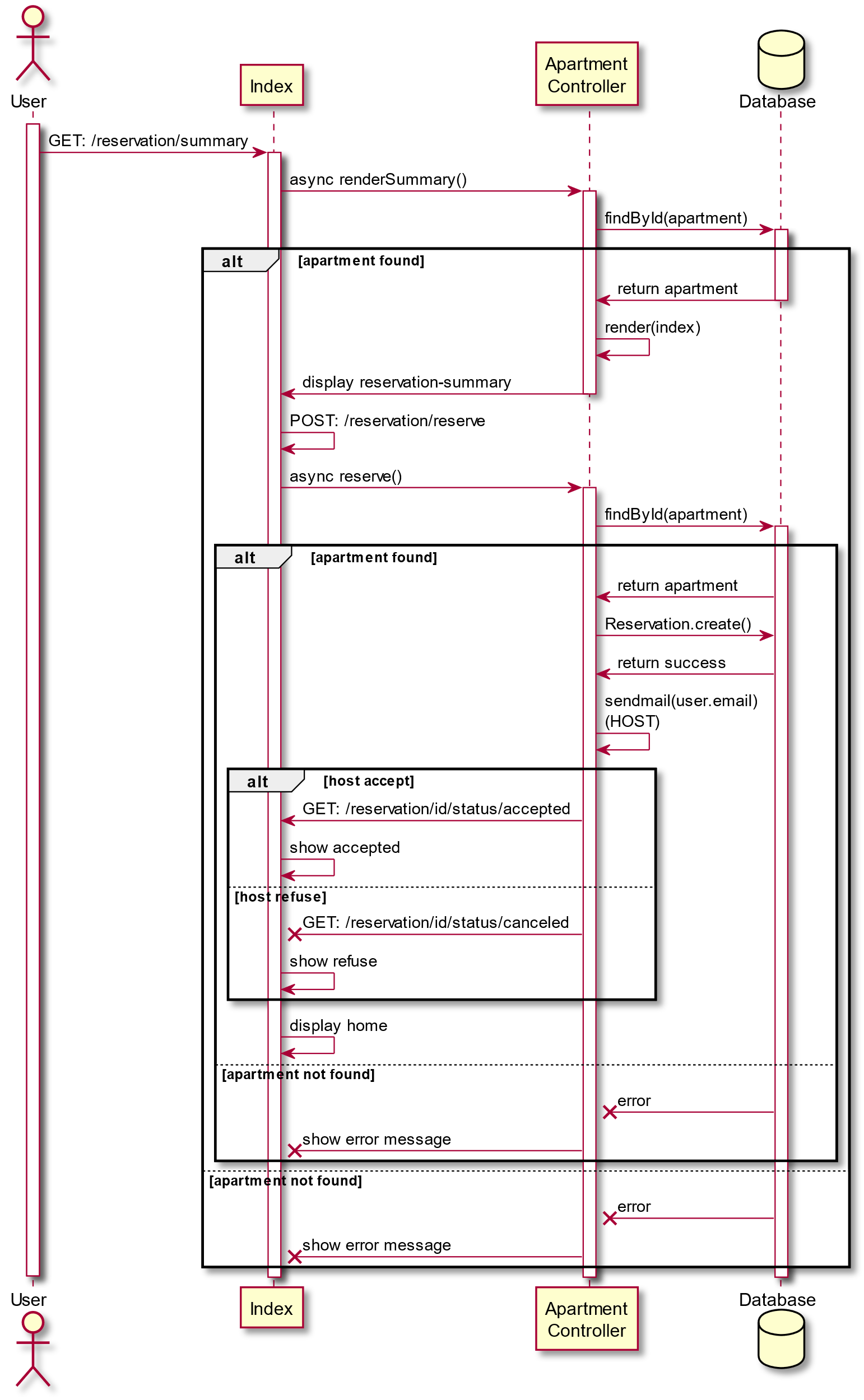
Inserimento di un annuncio per un appartamento



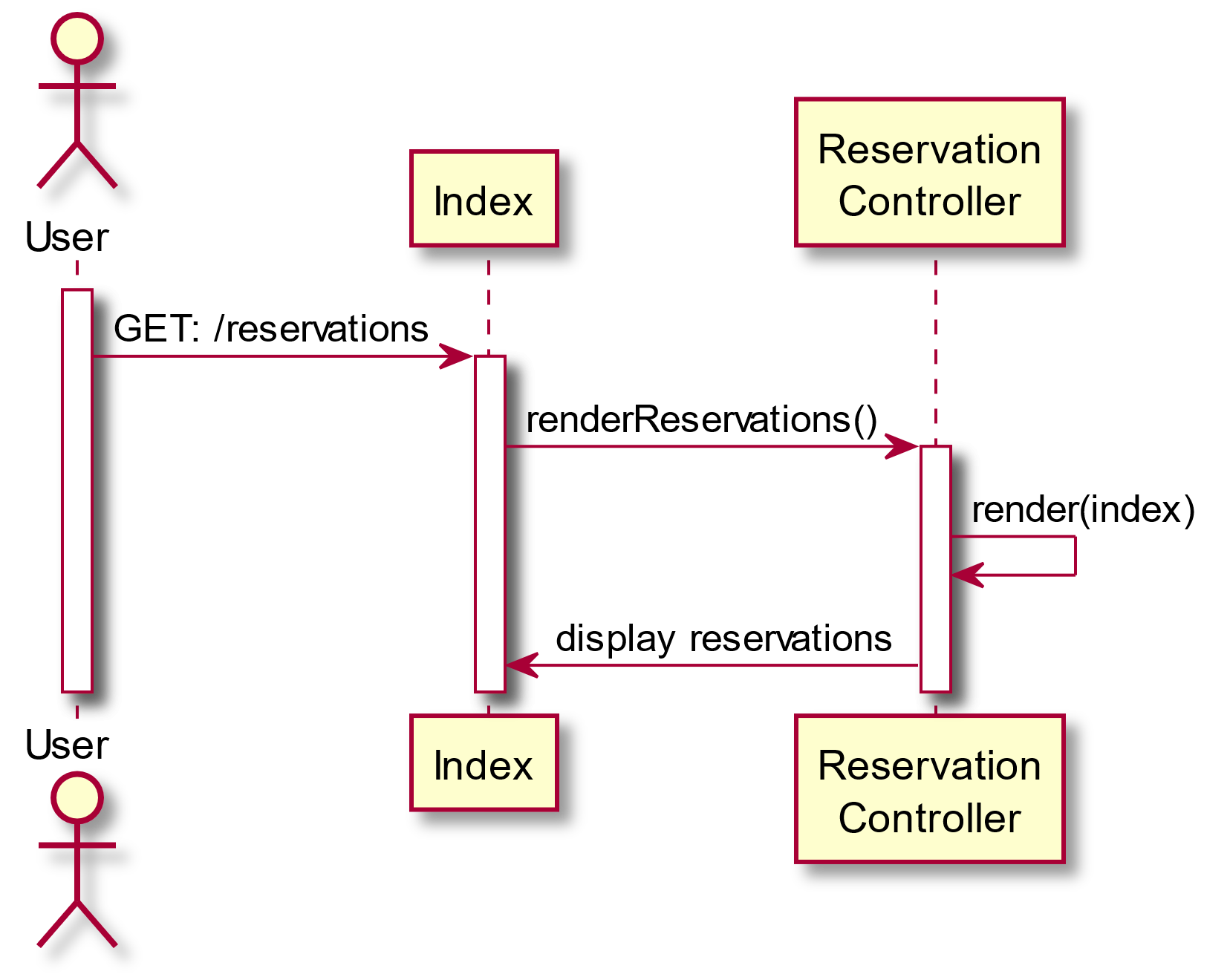
Ricerca di appartamenti



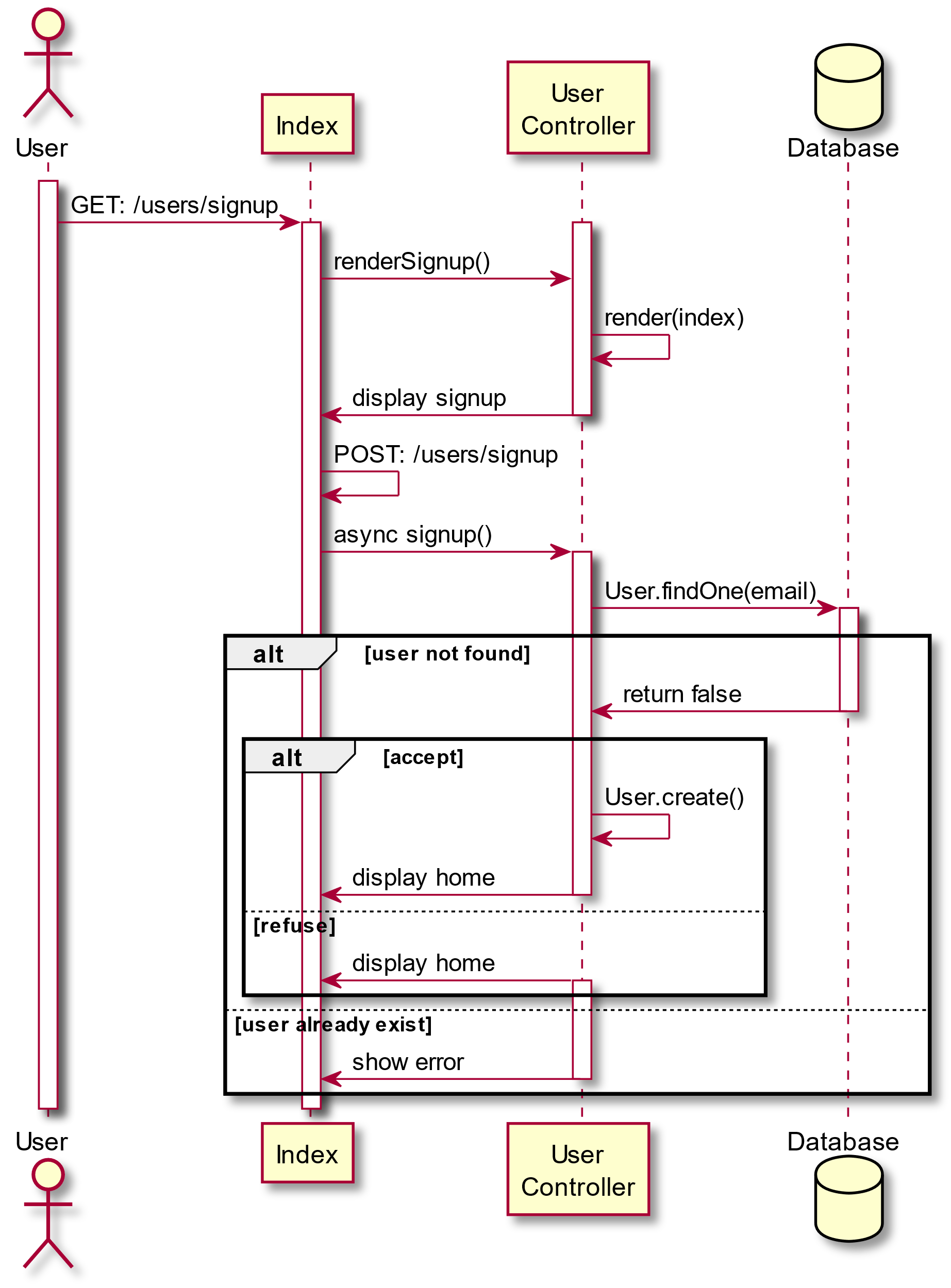
Prenotazione di un appartamento da parte di un customer



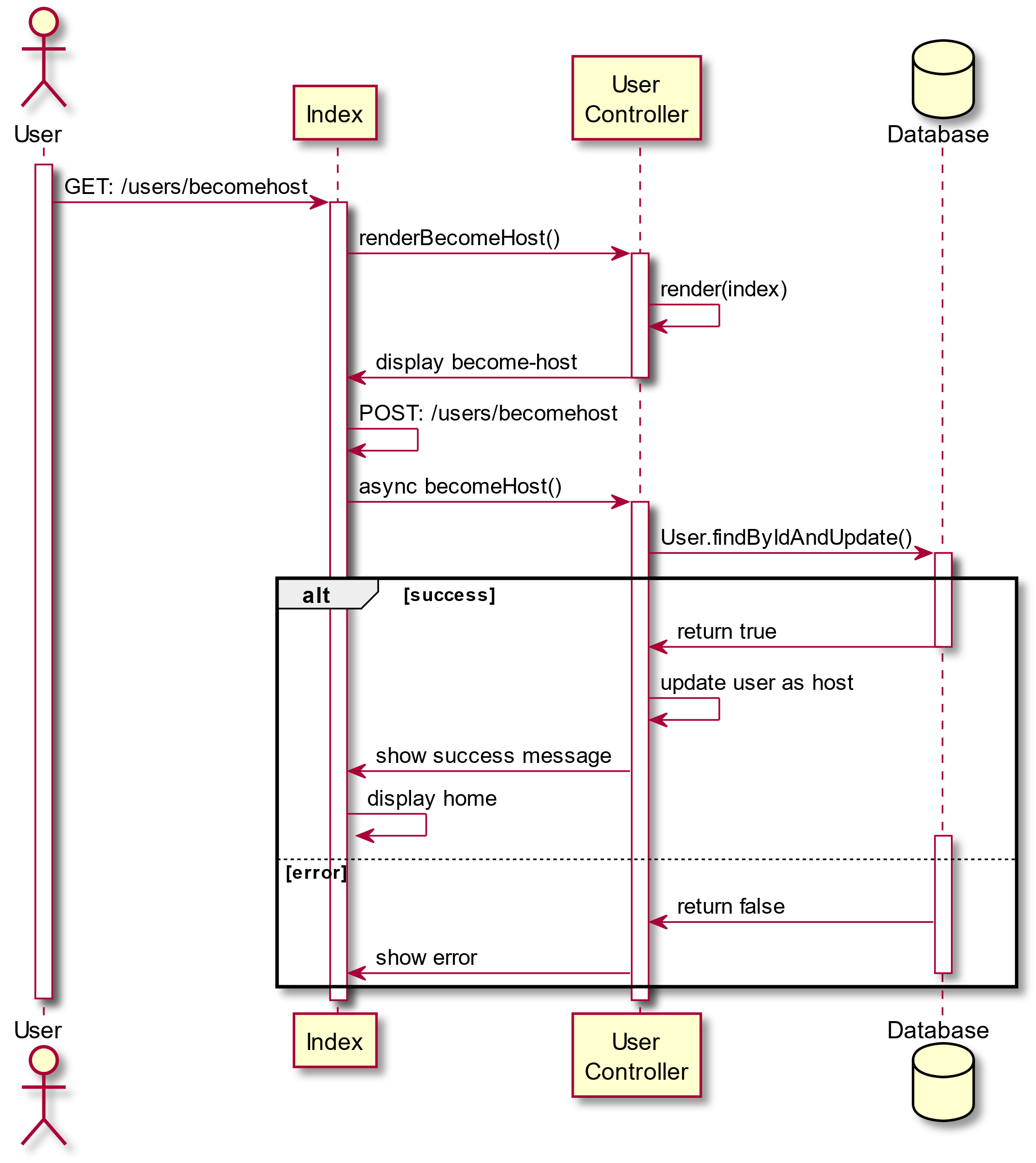
Visualizzazione dello storico prenotazioni



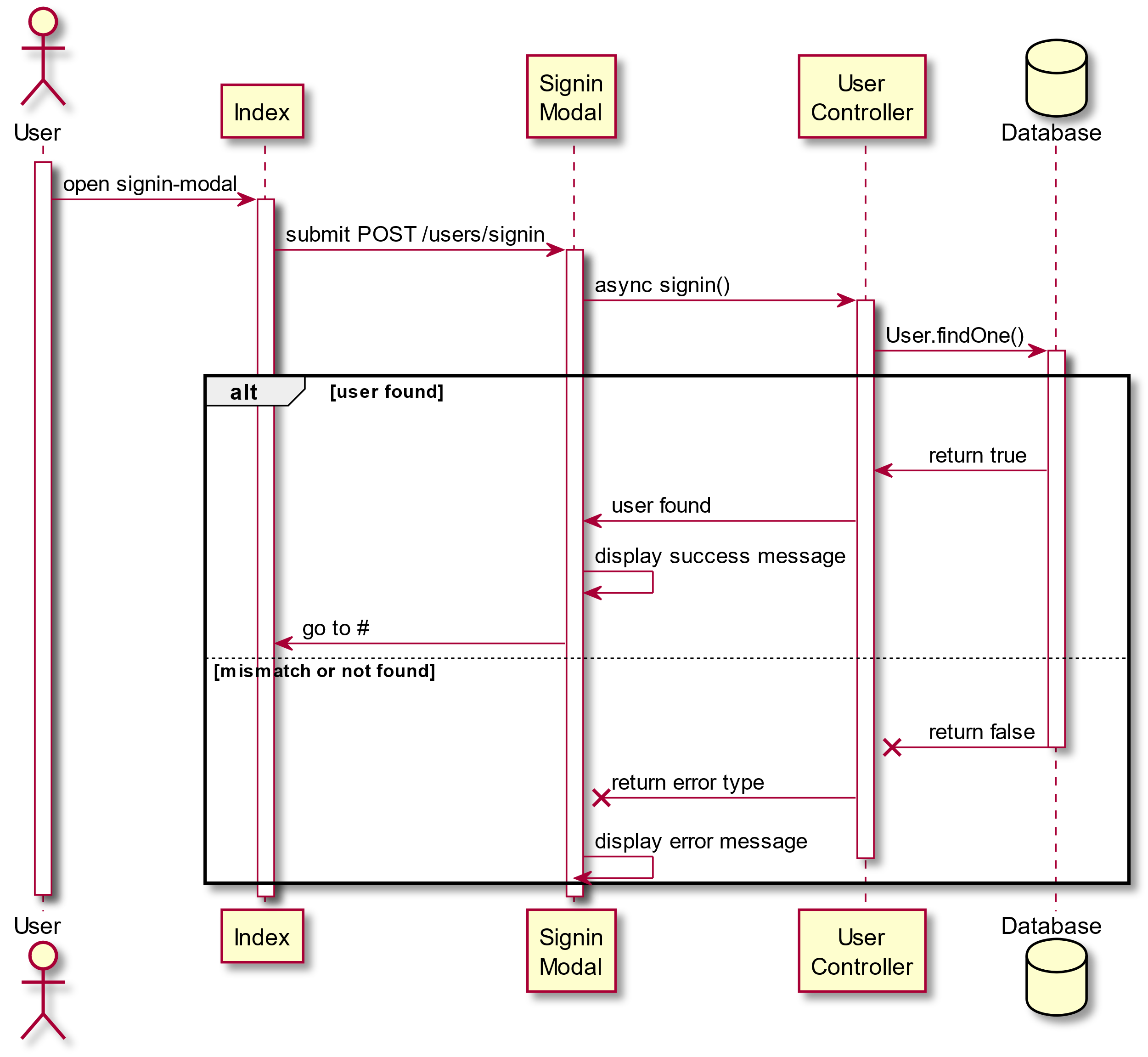
Processo di registrazione di un nuovo utente

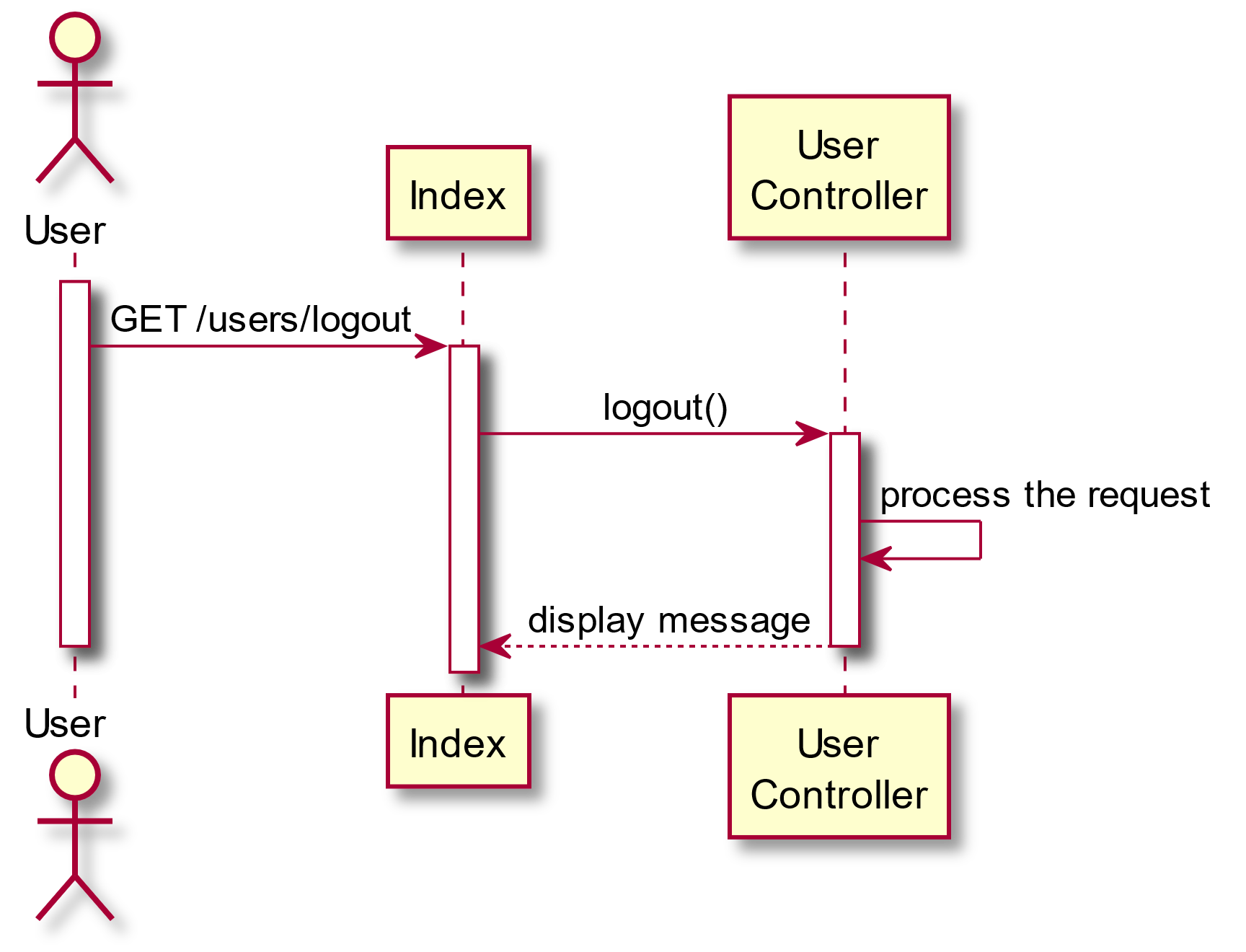


Processo per diventare host e poter pubblicare annunci



Processi di login e logout



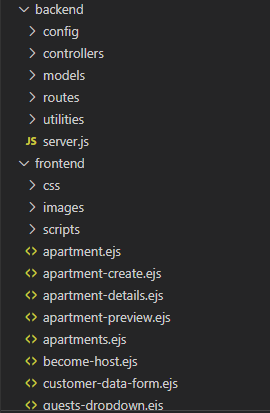


Visualizzazione dei dati del profilo utente ed eventuale aggiornamento dei dati

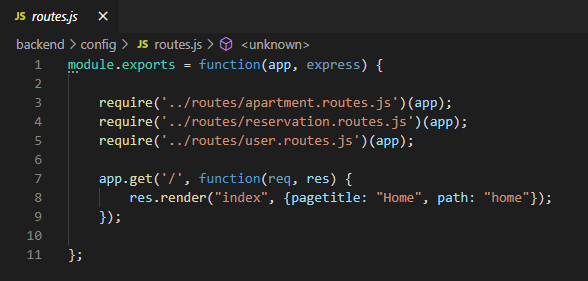


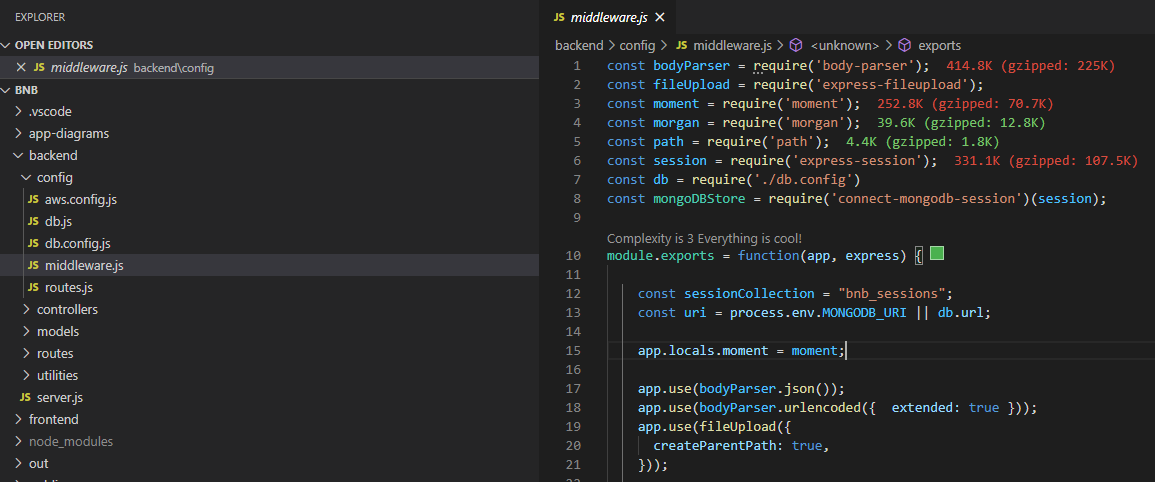
Fase di Sviluppo

Per la fase di sviluppo ci siamo innanzitutto preoccupati di strutturare il progetto in modo da separare logicamente il codice backend da quello frontend nell’ottica di seguire il pattern MVC. Nella cartella frontend abbiamo dunque inserito tutte le views, mentre nella cartella backend si trova il codice JS relativo ai controller, alle routes e al modello.



Nella cartella **config** abbiamo il codice di configurazione per il database, le routes principali, AWS (per l’upload dei file su cloud) e la configurazione dei vari middleware di express utilizzati.



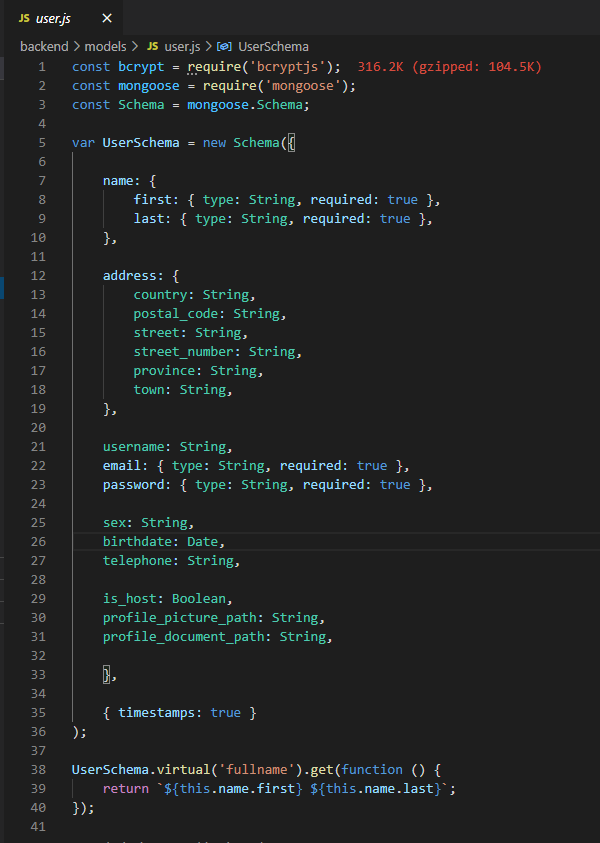


MVC (Model, View, Controller)

Modello

Per la definizione delle classi di modello abbiamo utilizzato gli **Schema** di Mongoose (un ORM per MongoDB). Di seguito i modelli definiti:

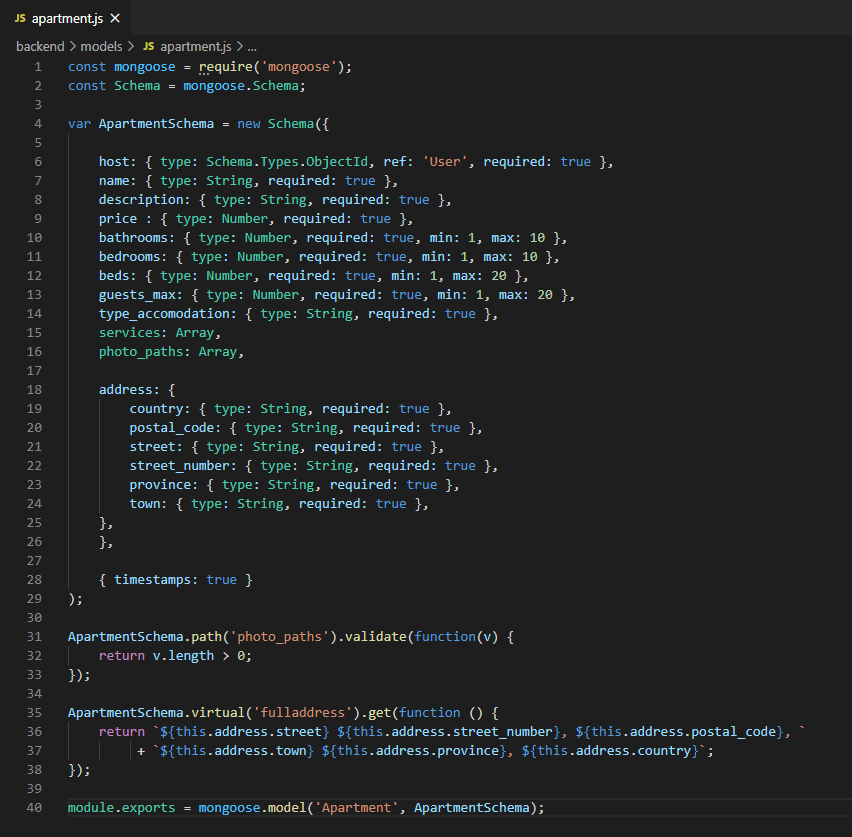
User



Per proteggere le password degli utenti abbiamo utilizzato bcrypt come libreria per la crittografia di quest’ultime, crittografando le password prima dell’effettivo salvataggio sul database, utilizzando il metodo “**pre**” su save dello **Schema**.

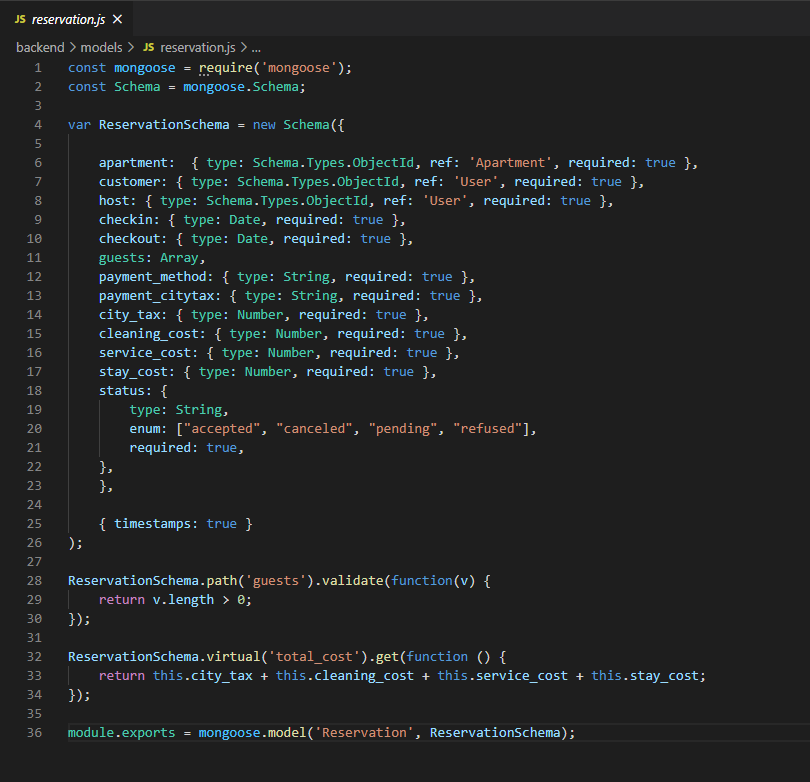


Apartment



Per Apartment oltre agli attributi descrittivi dell’appartamento stesso (come nome, descrizione, posti letto) abbiamo un array per conservare i percorsi delle foto caricate da chi pubblica l’annuncio, ed un riferimento ObjectId all’utente (host) che ha pubblicato l’annuncio.

Reservation



Per Reservation, abbiamo tre riferimenti esterni che rappresentano l’appartamento (**Apartment**) che è stato prenotato, il customer (**User**) che l’ha prenotato e l’host (**User**) che sta affittando l’appartamento. Inoltre abbiamo un array “guests” dove memorizziamo tutti i dati degli ospiti che verranno inseriti dal customer all’atto della prenotazione. Infine abbiamo un attributo status (**enum**) per indicare in quale stato si trova la prenotazione (confermata, rifiutata, in attesa o annullata dal customer).

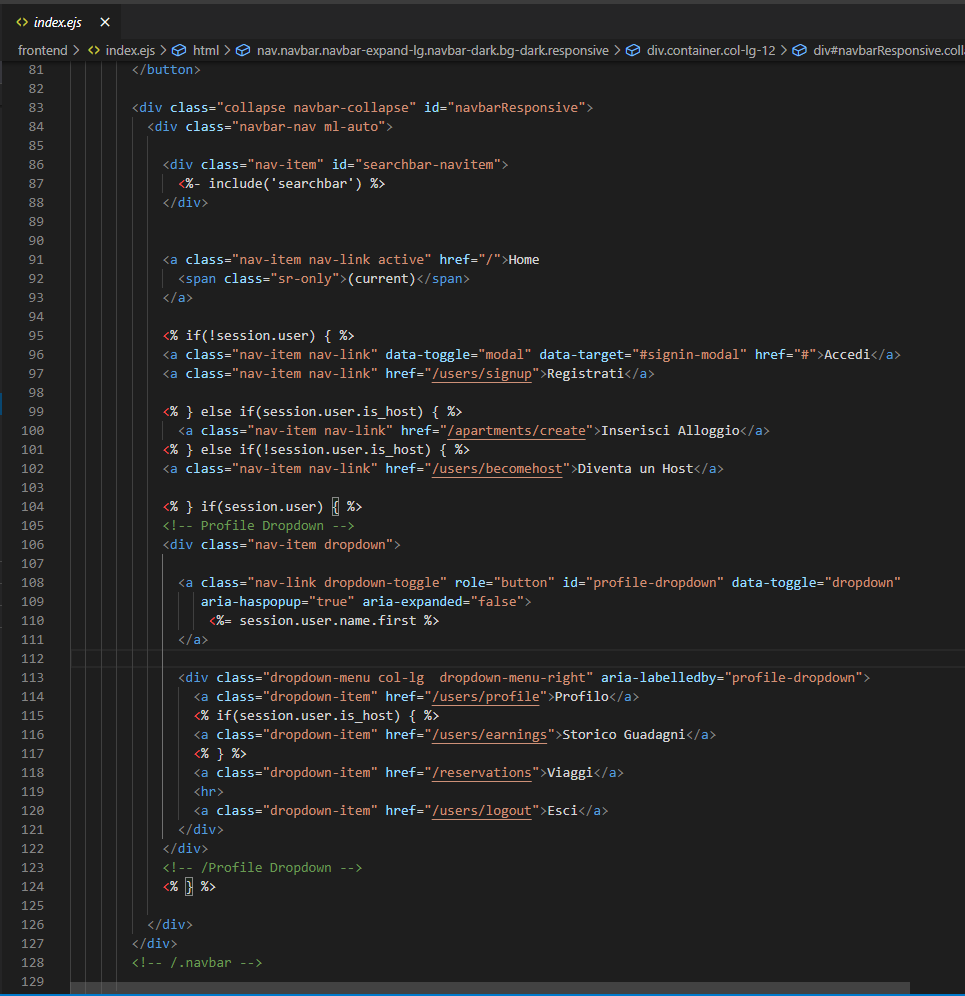
View

Utilizzando EJS abbiamo sfruttato molto il sistema di view parziali, impostando la web app secondo uno schema SPA (Single Page Application), caricando quindi tutto il necessario sempre nella stessa pagina (index.ejs) la quale poi attraverso il sistema di include di EJS e una variabile “path” che viene settata ogni volta dal controller in base alla route, carica la view corretta all’interno di index.ejs.

index.ejs

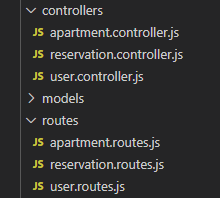


Inoltre sempre in index.ejs abbiamo sfruttato EJS per far comparire condizionalmente i diversi menu a seconda dello stato dell’utente (logged, not logged, host, not host, ecc.)

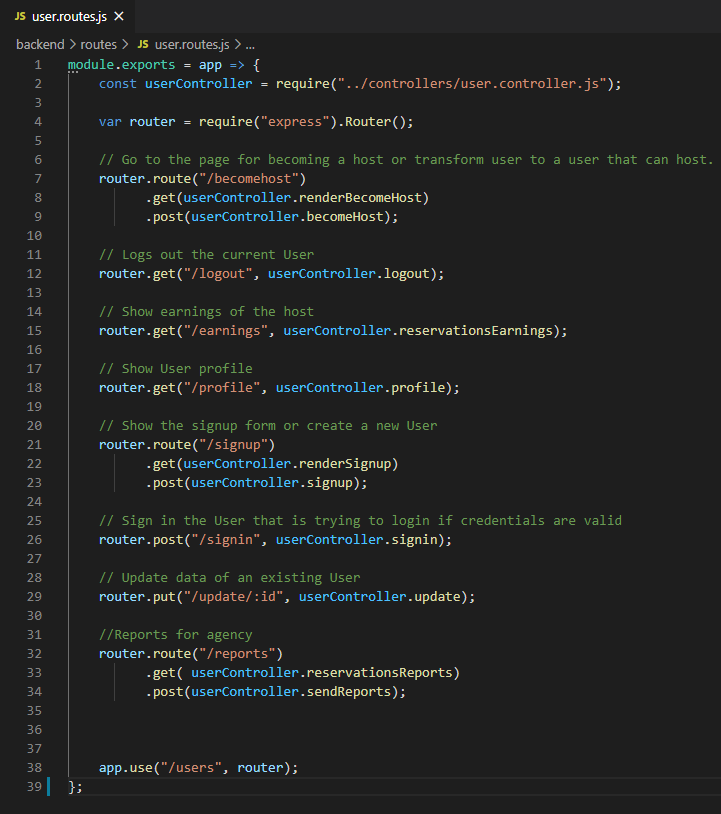


Controller e Routes

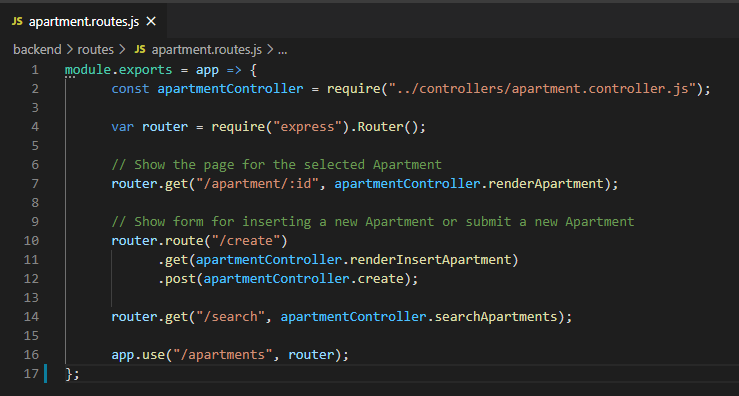
Abbiamo creato un file di controller e uno di route per la gestione di ciascuna entità (Apartment, Reservation e User).



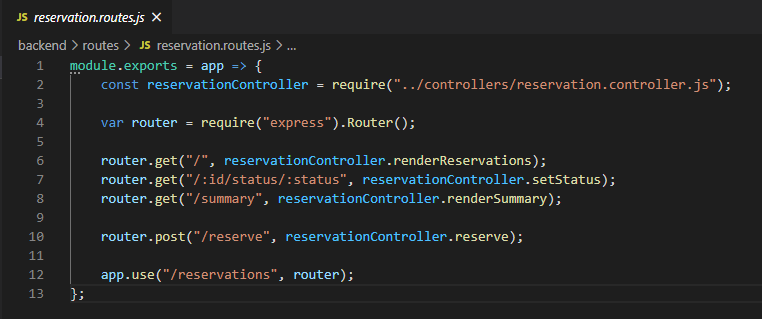
User



Apartment

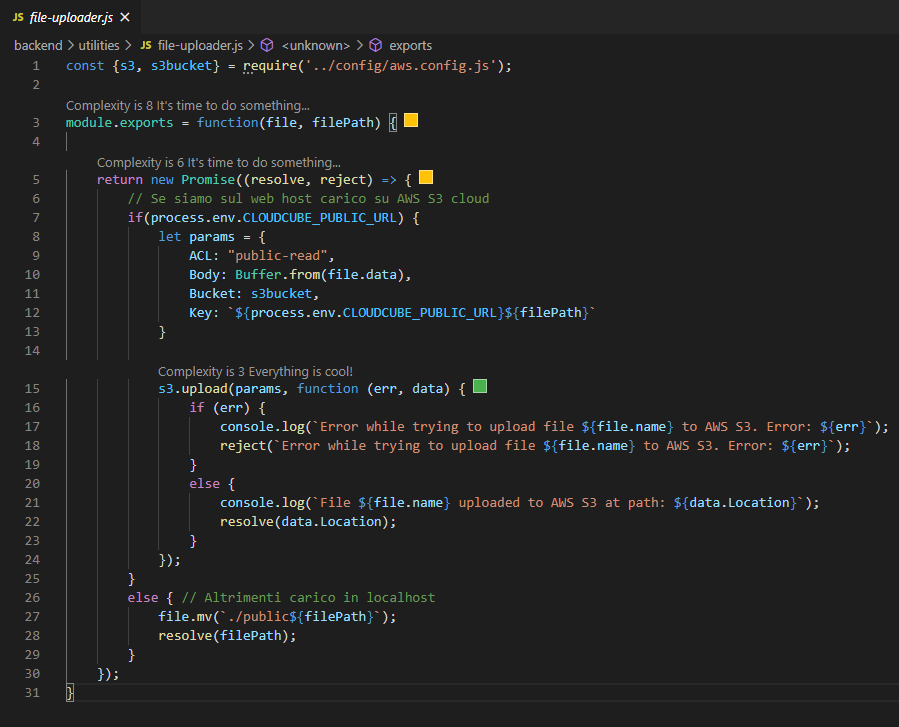


Reservation



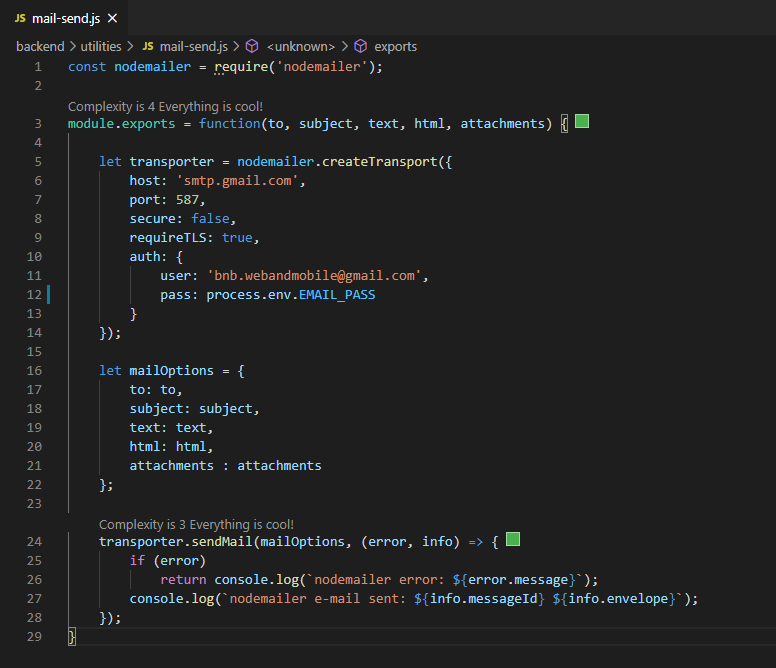
Utilities e scripts di supporto

file-uploader.js



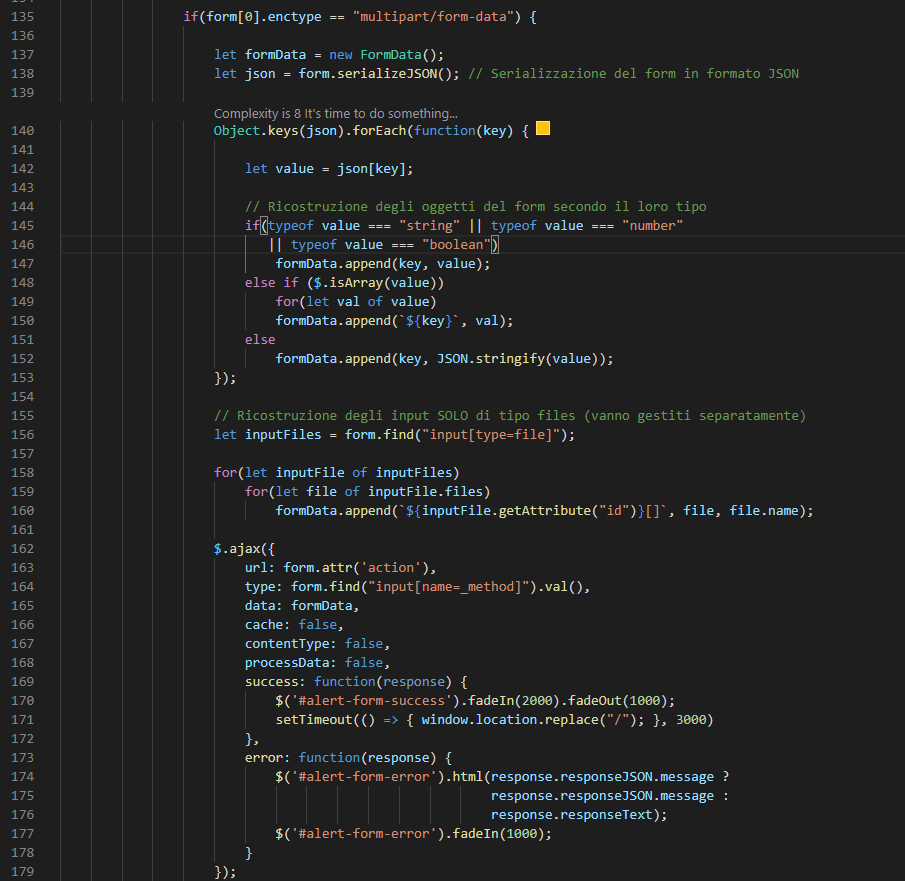
Questa utility ci permette l’uploading dei files su cloud quando la web app viene eseguita sul server web esterno di Heroku, utilizzando la libreria di Amazon aws-sdk.

file-uploader.js



Questa utility utilizza la libreria nodemailer per Node.js per poter inviare le e-mail richieste dalle specifiche dell’applicazione.

BNB.js



Nello script principale del frontend (bnb.js) gestiamo l’invio dei form, intercettando tutti i submit sui form e utilizzando AJAX per inviare le richieste in modo asincrono al server, gestendo anche le varie tipologie di from (application/x-www-form-urlencoded e multipart/form-data).

Note finali

Rilascio

La web app è stata distribuita su un server di hosting (Heroku) all’indirizzo  
<https://bnb-webmobile.herokuapp.com> e come file apk (con alcune funzionalità non completamente implementate).

Note di pre-rilascio

Le seguenti funzionalità non sono state implementate per il rilascio finale dell’app:

* Pagamento
* Recensioni

L’app Android presenta qualche bug che non siamo riusciti a risolvere

Software utilizzati per lo sviluppo

I seguenti software sono stati fondamentali per la fase di progettazione, design e implementazione dell’applicazione, oltre che per la realizzazione dei diagrammi UML e per la coordinazione del team:

**IDE:**

* Visual Studio Code

**Linguaggi:**

* HTML5
* CSS3
* JavaScript

**Framework e altre librerie CSS/JS:**

* Node.js
* Express
* MongoDB
* Mongoose (ORM)
* Bootstrap
* jQuery
* MJML
* Librerie Node.js (server-side):
  + aws-sdk
  + bcryptjs
  + body-parser
  + connect-mongodb-session
  + consolidate
  + dotenv
  + ejs
  + express
  + express-fileupload
  + jquery
  + jquery-validation
  + moment.js
  + mongoose
  + morgan
  + nodemailer
  + path
  + popper.js
* Librerie CSS/JS (client-side)
  + ejs
  + jquery-deserialize
  + jquery-serializeJSON
  + moment.js
  + Google Maps APIs (Maps, Places)
  + FontAwesome

**Altri strumenti utilizzati per la realizzazione dei diagrammi e della relazione**

* GraphViz
* PlantUML
* Microsoft Word

**Strumenti adoperati per la coordinazione del team e per il controllo di versione**

* Git / GitHub
* Google Drive
* Microsoft Teams
* Telegram
* Trello