



D4

Documento di sviluppo dell’applicazione

Gruppo T31

Luca Dematté

Diego Oniarti

Matteo Frizzera

Ingegneria del software

Anno Accademico 2022/23

Versione del documento

|  |  |  |
| --- | --- | --- |
| Versione | Data | Modifiche |
| 0.1 | 5/12/2022 | Creazione documento. |
| 0.2 | 8/12/2022 | Aggiunta struttura del documento |
| 0.3 | 21/12/2022 | Aggiunti user flow con descrizione |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Sommario

[Scopo del documento 4](#_Toc122543812)

[User flows 4](#_Toc122543813)

[Documentazione e implementazione dell’applicazione 6](#_Toc122543814)

[Struttura del progetto 6](#_Toc122543815)

[Dipendenze del progetto 6](#_Toc122543816)

[Database del progetto 7](#_Toc122543817)

[API del progetto 7](#_Toc122543818)

[Risorse estratte dal diagramma delle classi 7](#_Toc122543819)

[Modelli delle risorse 7](#_Toc122543820)

[Sviluppo delle API 7](#_Toc122543821)

[<API 1> 7](#_Toc122543822)

[<API 2> 7](#_Toc122543823)

[Documentazione delle API 8](#_Toc122543824)

[Implementazione del FrontEnd 8](#_Toc122543825)

[Repository GitHub e informazioni sul deployment 8](#_Toc122543826)

[Testing 8](#_Toc122543827)

# Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte del sistema. In particolare, presenta tutti gli artefatti necessari per realizzare i servizi di gestione degli utenti e prenotazioni del sistema.

Partendo dalla descrizione degli user flow legati al ruolo dell’utente del sistema, il documento prosegue con la presentazione delle API necessarie (tramite l’API Model e il Modello delle risorse) per poter visualizzare, inserire e modificare sia gli utenti sia le prenotazioni del sistema.

Per ogni API realizzata, oltre ad una descrizione delle funzionalità fornite, il documento presenta la sua documentazione e i test effettuati. Infine, una sezione è dedicata alle informazioni del Git Repository e il deployment dell’applicazione stessa.

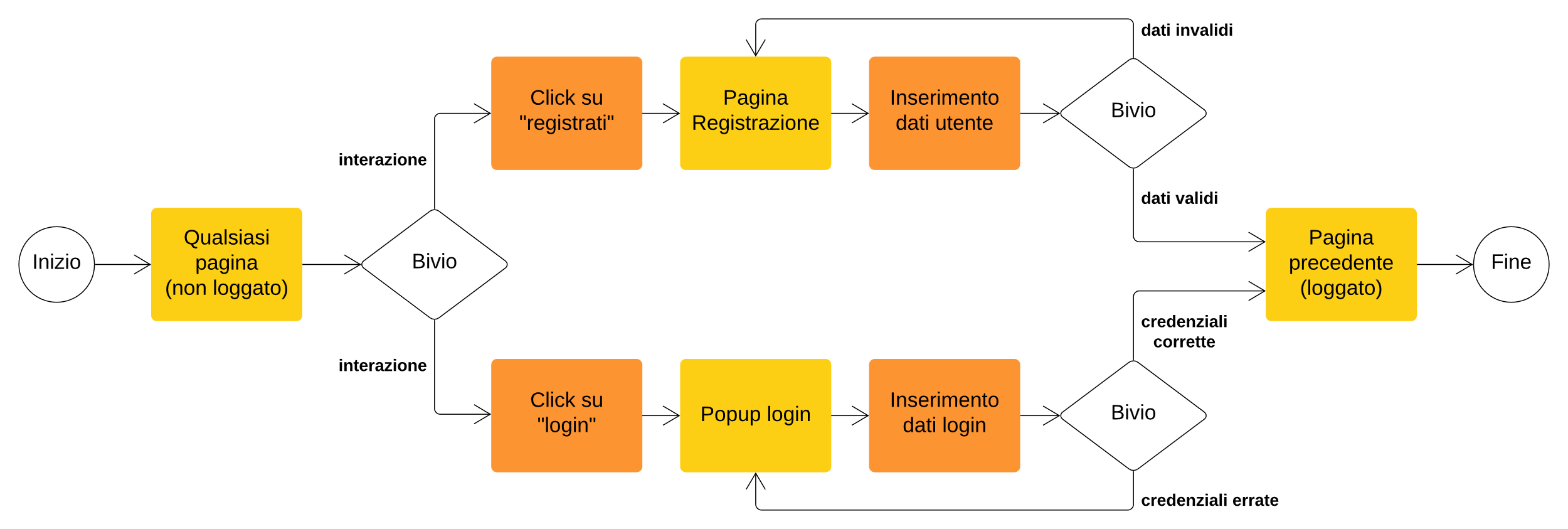
# User flows

In questa sezione del documento di sviluppo riportiamo gli “user flows” per il ruolo dell’utente della nostra applicazione.

La Figura 1 descrive il procedimento di login o di registrazione dell’utente.

In ogni momento e da ogni pagina, un utente non autenticato può premere il relativo pulsante per effettuare il login o per registrarsi, raggiungendo la pagina per effettuare l’operazione selezionata. Dopo aver inserito i dati richiesti, viene effettuato un controllo sulla loro validità:

* Nel caso in cui i dati inseriti siano errati, l’utente viene riportato alla pagina per inserire nuovamente i dati,
* Se invece i dati inseriti sono corretti, l’operazione è conclusa e l’utente viene riportato alla pagina che stava visualizzando prima di iniziare il procedimento, ma questa volta è autenticato.

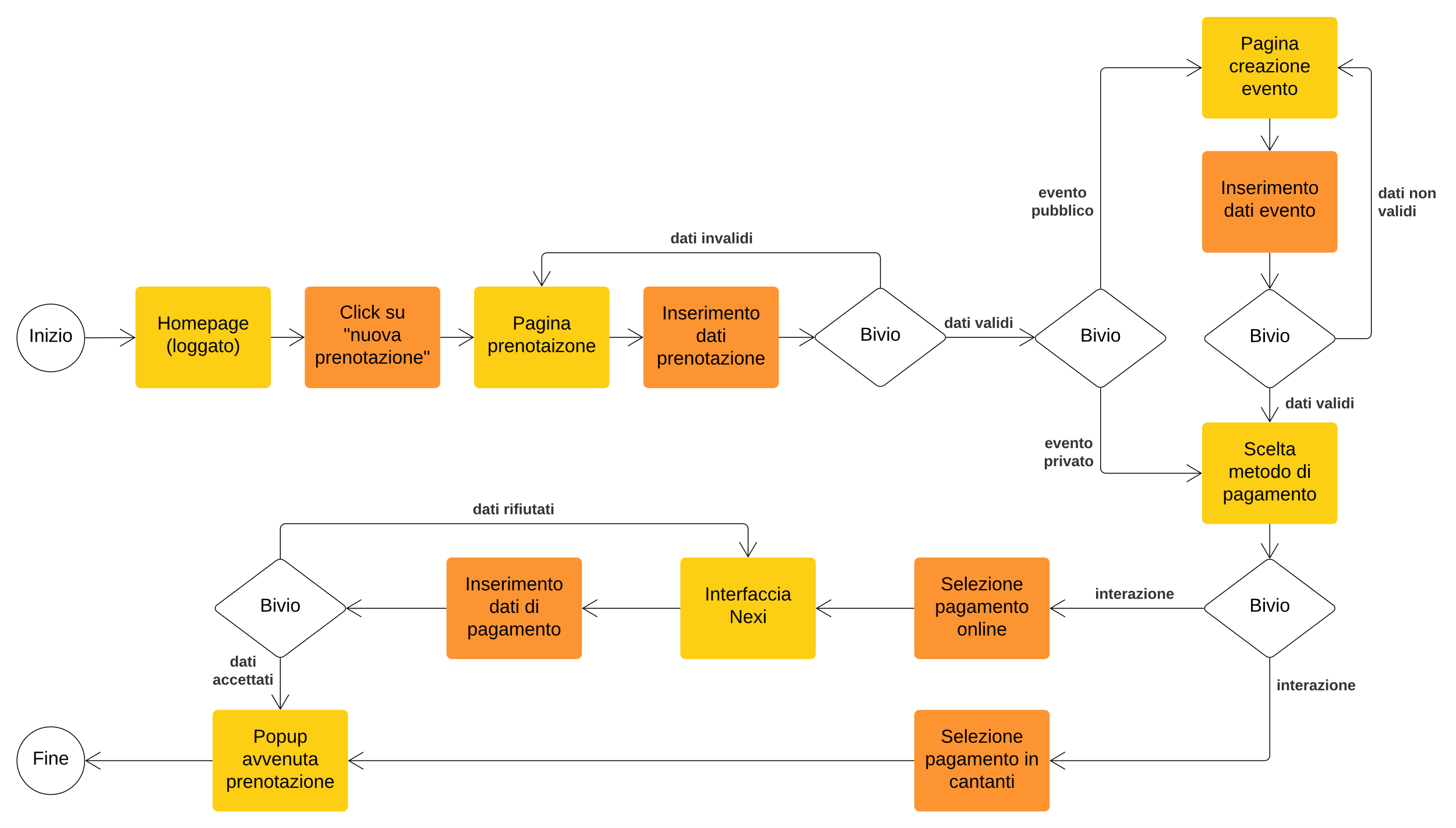


La Figura 2 mostra un esempio di inserimento di prenotazione da parte di un utente autenticato.

Dopo aver selezionato l’opzione “nuova prenotazione”, viene richiesto di inserire i dati relativi alla prenotazione. Successivamente, l’utente deve indicare se la prenotazione è per un evento pubblico (che quindi diventerà visibile sul sito) oppure se si tratta di un evento privato. Nel caso in cui l’evento sia pubblico, vengono richiesti i dati relativi all’evento per la visualizzazione sul sito. In seguito, viene richiesto di indicare come si desidera effettuare il pagamento:

* Se l’utente seleziona il pagamento online, viene avviata la procedura di pagamento tramite il sistema esterno Nexi,
* Se invece l’utente decide di pagare in contanti (o non dispone di un metodo di pagamento online) viene informato su come ed entro quando recarsi in segreteria per pagare la prenotazione.

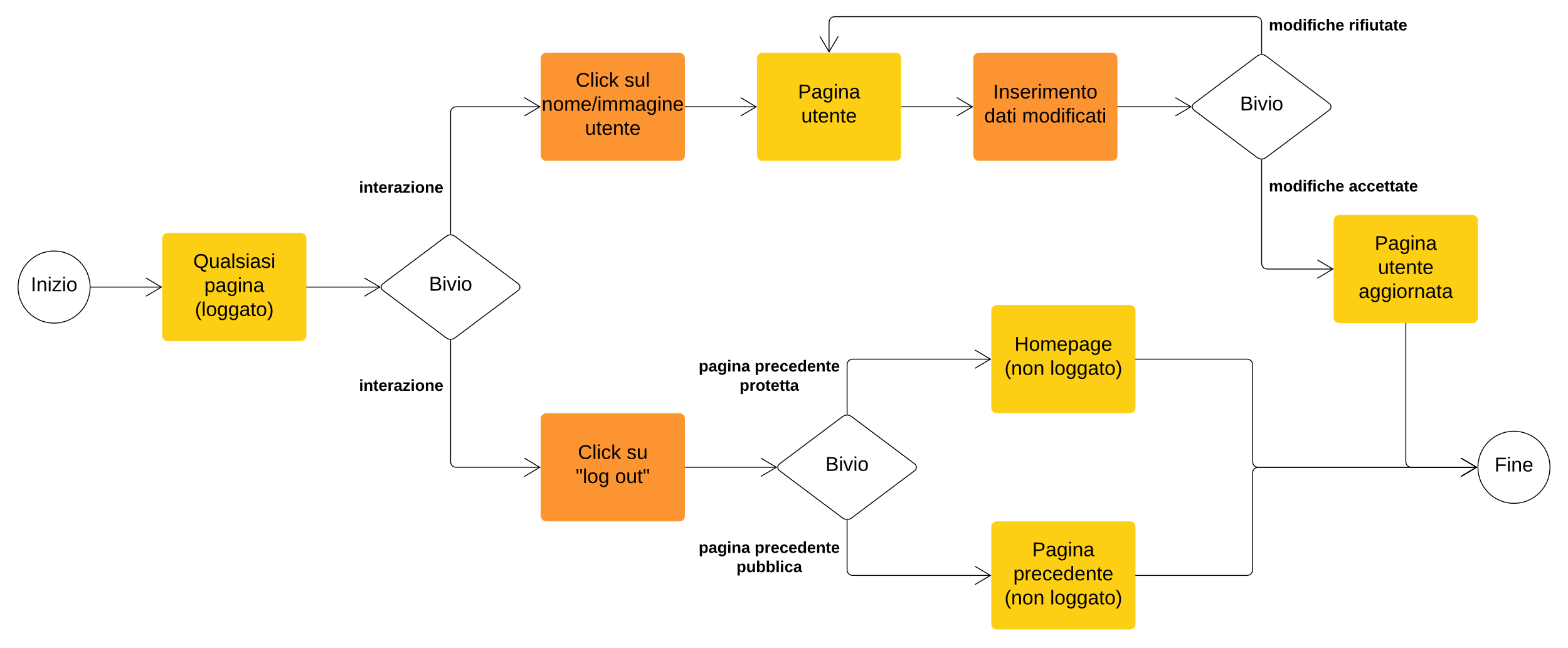
Al termine di entrambi i casi, la prenotazione viene registrata e viene mostrato un avviso di conferma.



La Figura 3 mostra i passaggi che l’utente deve svolgere per vedere e modificare il proprio profilo o per poter effettuare il logout.

Nel primo caso, dopo aver cliccato sulla propria immagine, viene portato alla pagina del profilo, dove può opzionalmente modificare le proprie informazioni personali. Dopo aver effettuato tutte le modifiche che desidera esse vengono verificate e, se sono valide, viene visualizzata nuovamente la pagina del profilo con i dati aggiornati.

Nel caso del logout, dopo l’operazione l’utente viene riportato alla pagina che stava visualizzando in precedenza oppure, se essa è disponibile solo per gli utenti autenticati, alla homepage.



# Documentazione e implementazione dell’applicazione

Nelle sezioni precedenti abbiamo identificato le varie features che devono essere implementate per la nostra applicazione con un’idea di come il nostro utente può utilizzarle nel suo flusso applicativo. L’applicazione è stata sviluppata utilizzando NodeJS e VueJS. Per la gestione dei dati abbiamo utilizzato MongoDB.

## Struttura del progetto

La struttura del progetto è presentata in figura ed è composta di una cartella api per la gestione delle api locali, di una cartella ui per la parte del front-end, e di una cartella Photos per la memorizzazione di immagini a supporto.

<TODO> descrivere e mettere foto dell’albero delle cartelle

## Dipendenze del progetto

I seguenti moduli Node sono stati utilizzati e aggiunti al file package.json

* Express
* Mongoose
* Dotenv
* <TODO> aggiungere le altre dipendenze

## Database del progetto

Per la gestione dei dati utili all’applicazione abbiamo definito due principali strutture dati come illustrato in figura. Una collezione di “dipartimenti” e una collezioni di “dipendenti”.

<TODO> mettere uno screenshot da mongoDB

Per rappresentare i dipartimenti e i dipendenti abbiamo definito i seguenti tipi di dati:

<TODO> inserire screenshot (o codice) della definizione delle collections

## API del progetto

### Risorse estratte dal diagramma delle classi

<TODO> testo Descrizione delle risorse estratte dal Class diagram

<TODO> Figura del diagramma delle risorse

### Modelli delle risorse

<TODO> testo Descrizione del resources model

<TODO> Figura del diagramma delle API (resources Model)

## Sviluppo delle API

<TODO> descrizione delle varie API sviluppate a partire dai modelli delle risorse

### <API 1>

<TODO> descrivere l’API 1

### <API 2>

<TODO> descrivere l’API 2

# Documentazione delle API

Le API Locali fornite dall’applicazione e descritte nella sezione precedente sono state documentate utilizzando il modulo NodeJS chiamato Swagger UI Express. In questo modo la documentazione relativa alle API è direttamente disponibile a chiunque veda il codice sorgente.

Per poter generare l’endpoint dedicato alla presentazione delle API abbiamo utilizzato Swagger UI in quanto crea una pagina web dalle definizioni delle specifiche OpenAPI.

In particolare, di seguito mostriamo la pagina web relativa alla documentazione che presenta le 3 API (GET, POST and DELETE) per la gestione dei dati della nostra applicazione.

La GET viene utilizzata per visualizzare i dati in una pagina HTML. La POST per inserire un nuovo dato nel nostro sistema. La DELETE per cancellare un dato dal nostro sistema.

L’endpoint da invocare per raggiungere la seguente documentazione e’:

http://localhost:8080/api-docs

<TODO> inserire screenshot di SwaggerUI

# Implementazione del FrontEnd

Il FrontEnd fornisce le funzionalità di visualizzazione, inserimento e cancellazione dei dati dell’applicazion. In particolare, l’applicazione è composta da una Home Page, da una pagina per la gestione …

<TODO> descrivere le varie pagine con testo e screenshot

# Repository GitHub e informazioni sul deployment

<TODO> Descrizione di come è strutturato il Repo Git

<TODO> Info sul deployment e sul link da utilizzare per eseguire l’applicazione su Heroku

# Testing

<TODO> Descrizione dei casi di test e del loro esito su alcune funzionalità dell’applicazione