Scopo del documento:

In questo documento andremo a descrivere tutti i componenti necessari per lo sviluppo dell'applicazione Work4Me, basandoci sui punti descritti e studiati nel corso dei vari deliverable precedenti.
Andremo ad identificare le user stories che descrivono le necessità degli utenti, le features che ne derivano e un diagramma User Flow che descrive il flusso di attività che un utente può compiere all'interno dell'applicazione.
Successivamente ci concentreremo sull'implementazione e sullo sviluppo dell'applicazione stessa, descrivendo le API realizzate, il database ad esse connesso e l'interfaccia grafica che si presenterà durante l'uso del software.

1. User Stories:

Con il termine "User story" intendiamo una "storia" scritta dalla prospettiva di un utente finale (che nel nostro caso potrà essere un comune utilizzatore o un amministratore) e che descrive in modo informale una sua necessità all'interno dell'applicazione.

Ogni user stories. identificata da un ID, descriverà con un breve testo il perché si sta sviluppando una particolare funzione del software. Saranno inoltre presenti due ulteriori colonne che riportano i requisiti funzionali e non relativi ad ognuna delle stories.
Avranno tutte una struttura simile del tipo: "Come [tipo di utente], voglio [un azione] in modo da [un beneficio]".

Di seguito le user stories identificate

ID	Descrizione	Req. Funzionale	Req. non Funzionale
US1	Come utente, voglio poter visualizzare gli annunci pubblicati da altri utenti, in modo da vederne i dettagli	RF 6	RNF 6
US2	Come utente, voglio poter filtrare gli annunci visualizzati, in modo da poter ricercare quelli più adatti alle mie esigenze	RF 6	
US3	Come utente, voglio poter valutare un annuncio, in modo da esprimere il mio giudizio sul servizio offerto	RF 8	
US4	Come utente, voglio poter segnalare un annuncio, così da evidenziarne l'inappropriatezza	RF 9	
US5	Come utente, voglio poter aggiungere un annuncio ai miei preferiti, in modo	RF 7	

	da poter visualizzarlo con più velocità in un secondo momento		
US6	Come utente, voglio poter ingaggiare un altro utente per un certo annuncio, in modo da ottenere i servizi offerti	RF 13	
US7	Come utente, voglio poter creare dei nuovi annunci inserendo un titolo, delle foto, una descrizione, una categoria, un costo orario e una località, in modo da poter essere selezionato ed ingaggiato da altri utenti	RF 4	
US8	Come utente, voglio poter visualizzare i soli annunci da me pubblicati, in modo da visualizzarne le informazioni	RF 11	
US9	Come utente, voglio poter modificare i miei annunci, in modo da cambiarne le caratteristiche ad esso relative	RF 5	
US10	Come utente, voglio poter rimuovere miei annunci, così da eliminare quelli non più disponibili	RF 12	
US11	Come utente, voglio poter modificare il mio profilo, in modo da aggiornarne le informazioni e contenuti	RF 18	
US12	Come utente, voglio poter sponsorizzare i miei annunci, in modo da renderli più visibili all'interno dell'applicazione	RF 10	
US13	Come utente, voglio poter visualizzare i profili di altri utenti, in modo da ricavare informazioni sulla loro persona e sugli altri annunci da loro pubblicati	RF 16	
US14	Come utente, voglio poter contattare altri utenti, in modo da poter comunicare con loro riguardo i loro annuncio o riguardo il mio ingaggio	RF 15	
US15	Come utente, voglio poter cercare i profili di altri utilizzatori	RF 16	

	dell'applicazione, in modo da visualizzare i loro annunci e mettermi in contatto con loro		
US16	Come utente, voglio poter interagire con un chatbot, in modo da chiarire i miei dubbi relativi al servizio	RF 14	
US17	Come utente, voglio poter contattare un supporto umano, ottenendo così delle informazioni varie a cui il chatbot non ha saputo rispondere	RF 14	
US18	Come amministratore, voglio poter limitare la visione di un annuncio, in modo da poter verificare che questo sia appropriato o meno	RF 10	RNF 5
US19	Come amministratore, voglio poter riabilitare o eliminare un annuncio, gestendo così le segnalazioni effettuate dagli utenti	RF 10	RNF 5

2. Features:

Di seguito vengono riportate le varie funzionalità che dovranno essere presenti nella versione finale del software realizzato. Ognuna delle seguenti fà riferimento ad una delle storie descritte nel punto precedente, come indicato nella seconda colonna

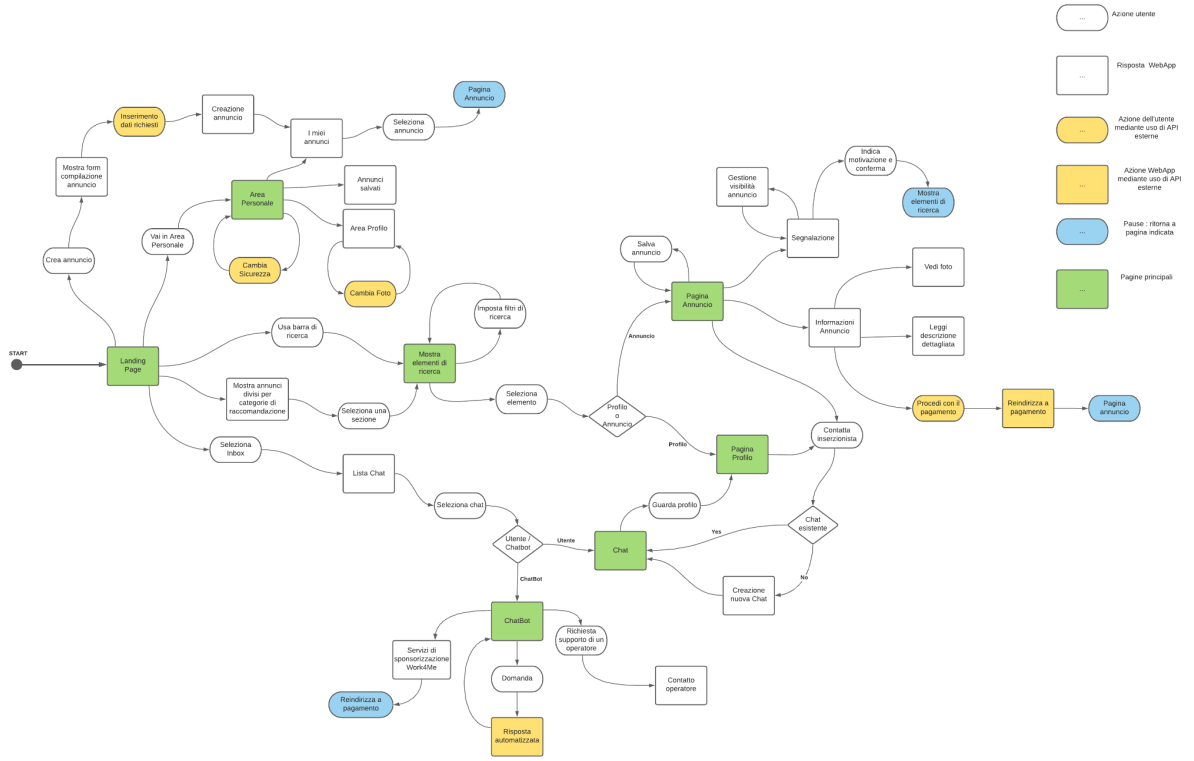
ID	**US ID**	**Descrizione**
F1	US1	Estrarre tutti gli annunci memorizzati tramite API
F2	US1	Lista degli annunci, divisi in tre categorie: "Consigliati per te", "I più votati" e "Sponsorizzati"
F3	US1	Pagina informazioni di un annuncio
F4	US2	Filtri per gli annunci per posizione, costo, valutazione media, categoria selezionabili attraverso un apposito form
F5	US2	Form per la ricerca di un annuncio
F6	US3	Form per la valutazione di un annuncio con un punteggio da 1 a 5
F7	US4	Tasto per la segnalazione di un annuncio

F8	US5	Tasto per aggiungere un annuncio ai preferiti
F9	US5	Lista degli annunci preferiti da un utente
F10	US6	Tasto per l'ingaggio di un utente per un certo annuncio
F11	US6	Schermata di pagamento
F12	US7	Form per la creazione di un annuncio, richiede un titolo, almeno una foto, una descrizione, una categoria, un costo orario e una località
F13	US7	Memorizzazione di un nuovo annuncio tramite API
F14	US8	Lista annunci pubblicati da un utente
F15	US9	Form per la modifica di un annuncio già esistente
F16	US9	Aggiornamento dei dati di un annuncio in seguito alla sua modifica, tramite API
F17	US10	Tasto per la rimozione di un annuncio esistente
F18	US10	Rimozione di un annuncio, tramite API
F19	US11	Form per la modifica di un profilo utente
F20	US11	Aggiornamento dei dati di un profilo in seguito alla sua modifica, tramite API
F21	US12	Form per la sponsorizzazione di un annuncio. Richiede la selezione di uno dei tier disponibili (1-4)
F22	US13	Profili utente con dati anagrafici dell'utente e riferimenti ai vari annunci da lui pubblicati
F23	US13	Tasto per il reindirizzamento alla pagina utente
F24	US14	Lista delle chat attive con altri utenti
F25	US14	Tasto per la creazione di una nuova chat
F26	US15	Form per la ricerca di un utente
F27	US16	Chatbot integrato con la chat dotato di risposte prestabilite da inviare in base alla domanda posta
F28	US17	Tasto per il reindirizzamento alla creazione di una mail verso il supporto utenti, attraverso l'applicazione di posta elettronica di default del dispositivo
F29	US18	Limitazione della visibilità degli annunci con più di 10 segnalazioni
F30	US18	Lista degli annunci momentaneamente nascosti ed in attesa di revisione
F31	US19	Visualizzazione degli annunci segnalati con tasti per rimozione o riabilitazione degli stessi

F32	US19	Aggiornamento dello stato degli annunci nascosti in seguito alla revisione

3. User Flow:

In questa sezione andremo a mostrare, tramite un apposito grafico, lo user flow di un utilizzatore dell'applicazione. Questo tipo di grafici descrivono attraverso degli appositi simboli (nel nostro caso indicati con un'apposita legenda esplicativa in alto a destra) il flusso di attività ed interazioni che un certo utente finale può effettuare durante l'uso del software.



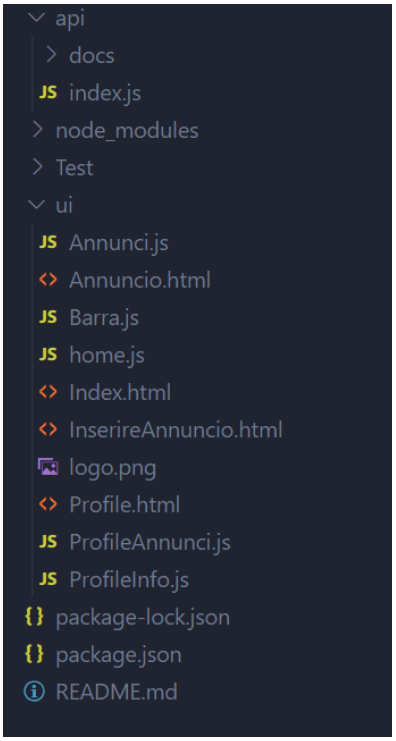
4. Application Implementation and Documentation:

In questa sezione andremo a descrivere come l'applicazione è stata sviluppata seguendo le linee guida ricavate dalla sezione precedente relativa alle features. Per lo sviluppo sono stati usati NodeJS per la gestione del lato server dell'applicativo e MongoDB per ciò che riguarda il database contenente i dati di utenti e annunci.

4.1. Project Structure:

Il progetto, come possiamo vedere dall'immagine sottostante è stato diviso in quattro sottocartelle. La cartella Api che contiene le api per la comunicazione con il database e la loro documentazione, Test che contiene le api per il testing dell'applicazione, node_modules che contiene le librerie necessarie per il funzionamento di Node e infine la cartella Ui che contiene tutti gli elementi dell'interfaccia grafica. Sono poi

presenti due file package.json e package-lock.json generati in automatico da Node e contenti informazioni sul progetto come ad esempio le dipendenze e i comandi start, test o simili. Infine è presente un file README.md che descriverà come utilizzare al meglio l'applicativo.



4.2. Project Dependencies:

Per la realizzazione del software abbiamo utilizzato i seguenti moduli Node:

- Body-Parser
- Cors
- Express
- MongoDB

Ognuno dei precedenti è stato aggiunto al file Package.json assieme alla relativa versione

4.3. Project Data or DB:

Per quanto riguarda il database, sono state definite due collezioni principali: "Annunci" e "Utenti", che conterranno rispettivamente i dati relativi agli annunci creati e agli account degli utenti registrati.
A queste due è stata poi aggiunta un'ulteriore collezione "AnnunciTest", che verrà utilizzata per il testing delle API, di cui discuteremo nelle sezioni successive. I dati al suo interno sono sostanzialmente identici a quelli della collection "Annunci".
Ognuno dei documenti contenuti all'interno delle due collezioni avrà una struttura con dei campi predefiniti. Un esempio di documenti presenti al loro interno sono i seguenti:

```
_id: ObjectId("61b1b99895c880c19dab8a33")
_idProprietario: ObjectId("313233343536373839303132")
titolo: "Dogsitter a Trento"
descrizione: "Offro servizi di dogsitting nei pressi di Trento e dintorni"
costo: "5"
luogo: "Trento"
categoria: "AnimaliDomestici"
foto: "foto"
voti: Array
  0: "4"
  1: "5"
segnalazioni: 0
sponsorizzato: true
nascosto: false

_id: ObjectId("61b88625667f42df46a3eeb2")
email: "nicola.mores@studenti.unitn.in"
passwordHash: "5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8"
refCode: "fiCYGGamvzCwF"
invitedBy: "5XNtT1urDBl0p"
preferiti: Array
  0: ObjectId("61b1c0baa1f86a1ede42740d")
recensioni: 2
badges: Array
  0: "3"
```


4.4. Project APIs:

4.4.1. Elenco Utenti:
Tramite questa API sarà possibile recuperare dal database l'elenco di tutti gli utenti registrati nel sistema. Per fare ciò viene utilizzato il metodo find chiamato senza alcun parametro

4.4.2. Elenco Annunci:
Similmente all'API per ottenere l'elenco degli utenti, è stata implementata un'ulteriore API di tipo GET per recuperare l'elenco di tutti gli annunci creati nell'applicativo

4.4.3. Numero Annunci presenti:
Attraverso questa API sarà possibile recuperare il numero di annunci presenti nel database. Può essere utilizzata per fini statistici e per varie operazioni gestite dai componenti esterni associati al nostro software, quali Amazon Lex e SageMaker

4.4.4. Dati di un Utente:
Chiamando questa API di tipo GET si potranno recuperare i dati relativi ad uno specifico utente. Per fare ciò verrà passato un parametro id, che rappresenta un codice identificativo di un determinato account, che verrà

usato come parametro nel metodo find

4.4.5. Dati di un Annuncio:
Corrispettivo dell'API precedente, ma per ricercare le informazioni di un annuncio.

4.4.6. Creazione Utente:
Questa API del tipo POST verrà chiamata ogni qual volta che un utente porterà a termine la registrazione all'applicazione. Sarà creato un nuovo document all'interno della collection Utenti che conterrà i dati, inseriti dall'utente, relativi a nome, cognome, data di nascita, email e l'eventuale referral code dell'affiliato che ha invitato il nuovo utente. Oltre a queste informazioni verrà poi memorizzata l'hash della password inserita durante la registrazione ed un referral code generato randomicamente tramite un'apposita funzione. Sono inoltre impostate di default il numero di recensioni, posto a 0, e gli array contenenti gli annunci preferiti e i badges posseduti, posti come vuoti.
Per compiere questa operazione verrà usato il metodo insertOne

4.4.7. Creazione Annuncio:
Tramite questa API sarà possibile creare un nuovo annunci da memorizzare all'interno della collezione Annunci. Verrà creato un nuovo document contenente le informazioni dell'annuncio quali titolo, descrizione, costo orario, luogo, categoria. Verranno poi impostate anche in questo caso alcune informazioni di default: Un array di interi per contenere i voti ricevuti, un contatore delle segnalazioni posto a 0 e due valori booleani per indicare se l'annuncio in questione è sponsorizzato o nascosto, entrambi posti a false. Ultimo dato memorizzato riguarda il riferimento al proprietario dell'annuncio. Per fare questo verrà inserito un ObjectId contenente questa informazione

4.4.8. Aggiornamento Utente:
Attraverso questa API di tipo PUT sarà possibile modificare i dati di un utente memorizzato all'interno del database, selezionato tramite il suo id. In particolar modo, sarà concessa la modifica dell'email, della password e della foto profilo.
Per fare ciò verrà utilizzato il metodo updateOne

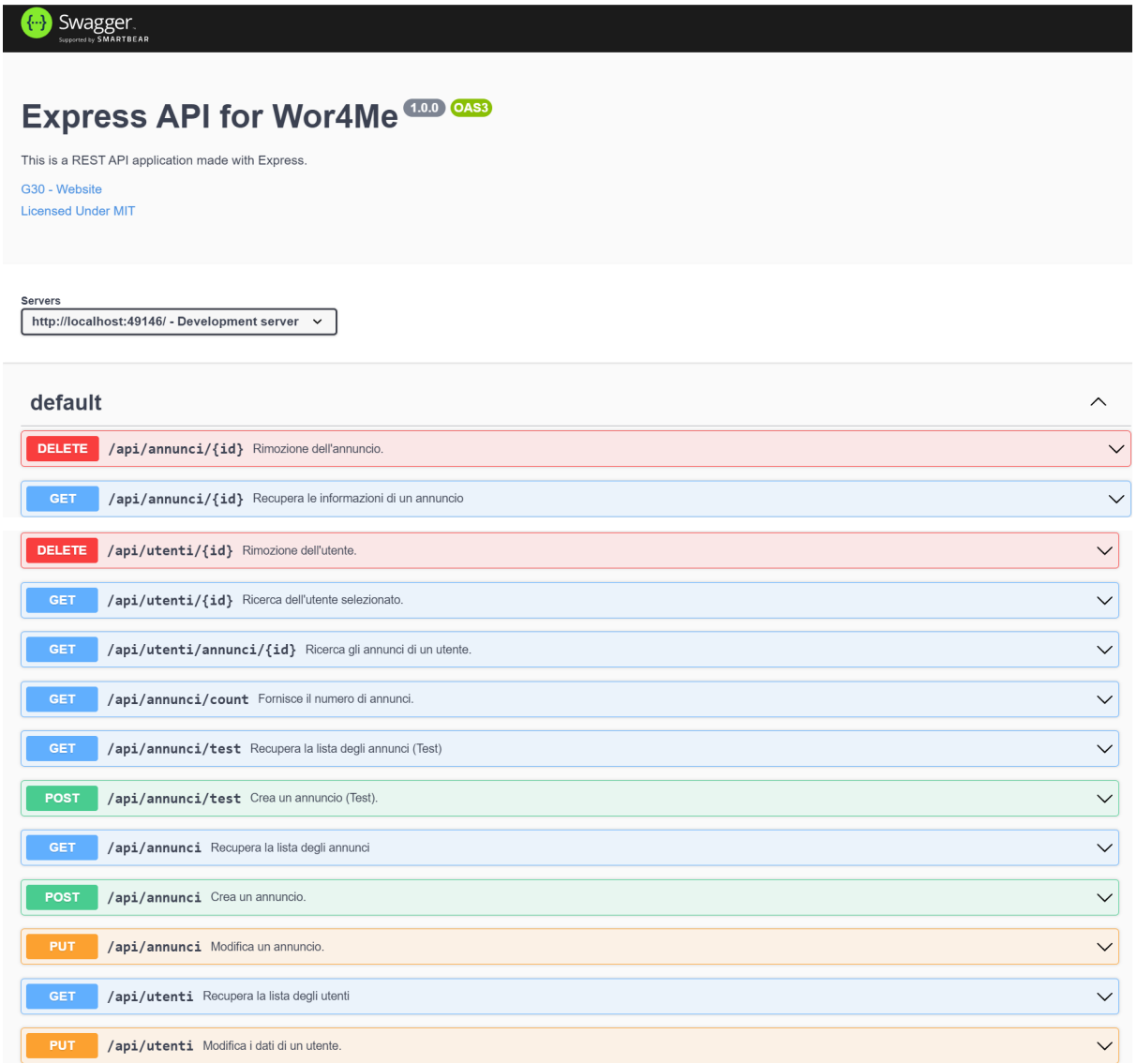
4.4.9. Aggiornamento Annuncio:
Similmente all'API precedente, utilizzando questa API si potranno modificare i dati relativi al titolo, descrizione, costo, luogo, categoria e foto di un annuncio selezionato sempre tramite l'id.

4.4.10. Cancellazione Utente e Annuncio:
Queste due API dal comportamento pressoché identico utilizzano il metodo deleteOne per rimuovere rispettivamente un utente o un annuncio, in entrambi i casi l'elemento da eliminare è indicato con il suo id.

5. API Documentation:

Le API dell'applicazione, precedentemente descritte, sono state documentate mediante il modulo Swagger UI Express di NodeJS.
Ogni API implementata nel sistema è stata documentata tramite appositi documenti in formato .yaml accessibili nel folder docs all'interno della cartella api nella directory dell'applicazione.
Per poter presentare le API abbiamo utilizzato Swagger UI per ottenere una pagina web dalle definizioni delle specifiche OpenAPI;

Di seguito la pagina web relativa alla documentazione che presenta le API utilizzate nell'esempio :



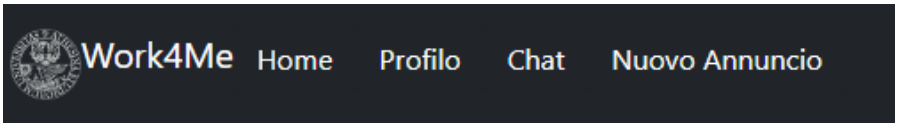
L'endpoint da invocare per raggiungere la seguente documentazione è :
http://localhost:49146/api-docs

6. FrontEnd Implementation:

Il FrontEnd fornisce all'utente la capacità di visualizzare, creare ed eliminare gli annunci oltre che visualizzare le informazioni personali e la lista dei propri annunci pubblicati. Le pagine messe a disposizione sono la home page, la pagina dedicata agli annunci singoli, quella dedicata all'inserimento dei nuovi annunci e per ultima la pagina dedicata al profilo personale. Lo stile grafico utilizzato è stato fornito dalla libreria open source Bootstrap.

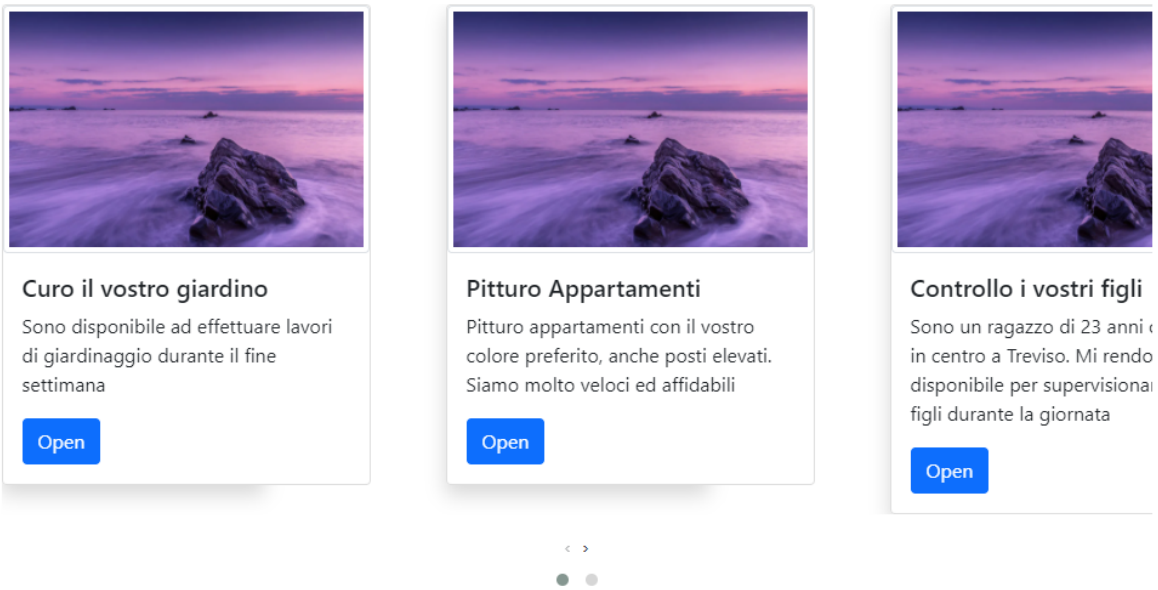
6.1. Navbar:

Ogni pagina implementa una navbar che contiene un logo, il nome dell'applicazione e i bottoni per la navigazione delle varie pagine.



6.2. Home:

La pagina principale dell'applicazione è la home page che permette la visualizzazione di tutti gli annunci presenti all'interno del database. La loro visualizzazione è stata effettuata tramite un carosello orizzontale che permette una facile visione di tutti gli annunci, tali annunci presentano un bottone che reindirizza alla pagina che mostra in modo dettagliato tutte le informazioni contenute nell'inserzione



6.3. Visualizzazione annuncio:

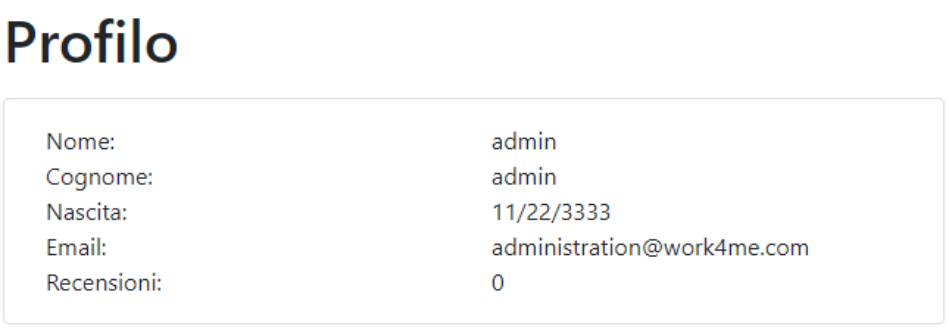
In questa pagina possiamo vedere con chiarezza tutte le informazioni contenute nelle inserzioni (titolo, descrizione, immagini, prezzo e voto). Anche qui è stato utilizzato un carosello per per la visualizzazione delle immagini

6.4. Inserimento annuncio:

Pagina per l'inserimento di un nuovo annuncio, tale pagina è composta da un classico form con svariati textField, una Select per le categorie e una TextArea per l'inserimento della descrizione. Per l'invio delle informazioni contenute nel form abbiamo utilizziamo un bottone

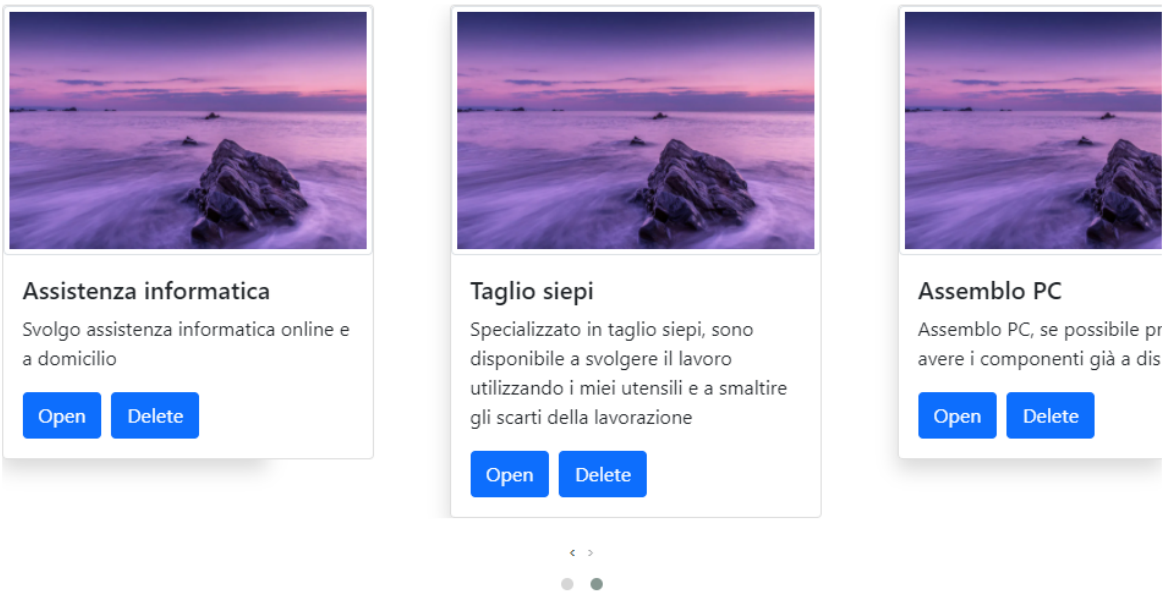
6.5. Profilo:

In questa sezione si possono visualizzare le informazioni del profilo e i propri annunci, tali annunci diventeranno visibili dopo la pressione del bottone apposito



6.6. Propri annunci e loro eliminazione:

Dopo la pressione del tasto “Mostra Annunci” verrà mostrato a schermo un carosello con tutti gli annunci dell’account corrispondente, tali annunci potranno poi essere visualizzati o eliminati tramite i tasti appositi



7. Github repository info:

Il progetto Work4Me è disponibile per il download nel seguente link: https://github.com/MarcoRcc/Work4Me. Le informazioni per il corretto funzionamento dell’applicazione sono presenti all’interno del file **README.**

8. API Testing:

Per eseguire il testing del nostro software è stata creata un'apposita cartella "Test" contenente il file "index.js". All'interno di questo file verrà chiamata un API di tipo GET creata appositamente per questo scopo chiamata /api/annunci/test. Questa si interfaccia ad una collections di prova "AnnunciTest" contente degli esempi di annunci. Una volta chiamata l'API, il programma controllerà che i valori ricevuti siano consoni a quelli che ci si aspetta e, in caso lo siano, darà un output come il seguente:

```
TAP version 13
# Test 1: Annunci ritornati correttamente
Mongo DB Connection Successfull
ok 1 Nessun errore
ok 2 Valori di ritorno conformi a quelli previsti
```

Assieme a questa prima API è stata aggiunta similmente un API di tipo POST la quale andrà ad inserire un nuovo annuncio, ancora una volta nella collezione "AnnunciTest". Come per il caso precedente, in caso non si presentino errori, otterremo un output del tipo:

```
# Test 2: Annuncio aggiunto correttamente
ok 3 Nessun errore
ok 4 Valori inseriti come previsto
```

Inoltre è stato modificato il file package.json, nella sezione scripts, in modo tale da poter richiamare queste due funzionalità di testing tramite il comando npm test.