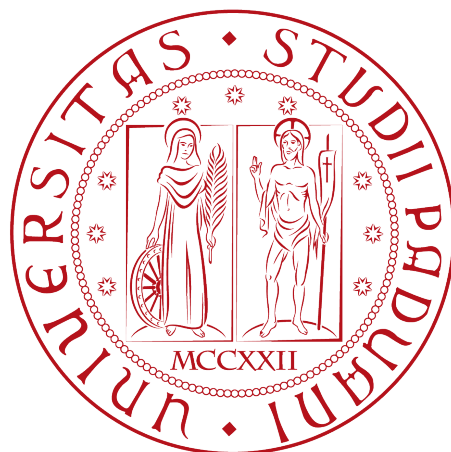


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Progettazione e realizzazione di UI per
WebApp tramite Angular e analisi degli
utenti con Hotjar

Tesi di laurea triennale

Relatore

Prof. Paolo Baldan

Laureando

Giovanni Cocco

Giovanni Cocco: *Progettazione e realizzazione di UI per WebApp tramite Angular e analisi degli utenti con Hotjar*, Tesi di laurea triennale, © Luglio 2022.

For now, what is important is not finding the answer, but looking for it.

— Douglas Richard Hofstadter

A Tommaso, per i preziosi consigli che sempre custodisco cari.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Giovanni Cocco presso l'azienda Sync Lab S.r.l. nel periodo che va dal 26/04/2022 al 24/06/2022.

Lo scopo dello stage era la realizzazione delle interfacce di una WebApp per permettere la gestione degli ordini in un ristorante di sushi tenendo conto degli aspetti di [UX](#)^[g].

In questo documento viene descritta l'organizzazione del suddetto stage, la fase di analisi del problema e la conseguente progettazione.

In particolare viene discussa la progettazione delle interfacce tramite Figma, la realizzazione di esse tramite il framework Angular e l'integrazione di quest'ultimo con HotJar per il tracciamento degli utenti.

“Non soffocare la tua ispirazione e la tua immaginazione, non diventare lo schiavo del tuo modello.”

— Vincent van Gogh

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Paolo Baldan, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Ho desiderio di ringraziare poi l'Ingegnere Pallaro Fabio, tutor aziendale, per la sua grande disponibilità dimostrata durante il periodo di stage.

Desidero ringraziare con affetto la mia famiglia per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Padova, Luglio 2022

Giovanni Cocco

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	Introduzione al progetto	1
1.3	Soluzione scelta	2
1.4	Descrizione del prodotto ottenuto	2
1.5	Principali problematiche	3
1.6	Strumenti utilizzati	3
1.7	Organizzazione del testo	5
2	Descrizione dello stage	7
2.1	Requisiti e obiettivi	7
2.2	Analisi preventiva dei rischi	8
2.3	Pianificazione	8
2.4	Organizzazione dello stage	9
3	Analisi dei requisiti	11
3.1	Confronto con gli stakeholders	11
3.2	Personas	11
3.3	Entità	12
3.4	Casi d'uso - Cliente	13
3.4.1	Attori	13
3.4.2	Gestione sessione tavolo	14
3.4.3	Gestione ordini	16
3.4.4	Area personale	22
3.5	Casi d'uso - Gestore	24
3.5.1	Attori	24
3.5.2	Area personale	25
3.5.3	Gestione menù	27
3.5.4	Gestione piatti	30
3.5.5	Guida d'aiuto	33
3.6	Tracciamento dei requisiti	34
3.6.1	Requisiti funzionali	34
3.6.2	Requisiti qualitativi	37
3.6.3	Requisiti di vincolo	37
4	Progettazione e codifica	39
4.1	Progettazione interfacce	39
4.1.1	Cliente	39

4.1.2	Gestore	43
4.1.3	Aggiornamento dopo confronto	45
4.1.4	Design della <i>API REST</i>	46
4.2	Formazione sulle tecnologie	46
4.3	Struttura software	47
4.3.1	Moduli	47
4.3.2	Servizi	48
4.3.3	Interceptor	48
4.3.4	Guardie	48
4.3.5	Componenti	49
4.4	Design Pattern utilizzati	49
4.5	Codifica	51
4.5.1	Componente di navigazione	51
4.5.2	Maschera di login	52
4.5.3	Maschera di registrazione	53
4.5.4	Maschera di inserimento ordini	54
4.5.5	Gestione dei temi	58
4.5.6	Blacklist ingredienti	59
4.5.7	Gestione piatti	60
4.5.8	Gestione menù	61
5	HotJar	63
5.1	Integrazione con Angular	63
5.2	Limiti tecnici di HotJar	63
5.3	Dashboard	64
5.4	Heatmap	65
5.5	Recording	68
6	Verifica e validazione	69
6.1	Verifica accessibilità	69
6.2	Validazione dei requisiti	73
7	Conclusioni	75
7.1	Raggiungimento degli obiettivi	75
7.2	Conoscenze acquisite	76
7.3	Valutazione personale	76
	Acronimi	77
	Glossario	79
	Bibliografia	81

Elenco delle figure

3.1	Figura casi d'uso 1	14
3.2	Figura casi d'uso 2	17
3.3	Figura casi d'uso 3	18
3.4	Figura casi d'uso 4	19
3.5	Figura casi d'uso 5	20
3.6	Figura casi d'uso 6	22
3.7	Figura casi d'uso 7	25
3.8	Figura casi d'uso 8	27
3.9	Figura casi d'uso 9	28
3.10	Figura casi d'uso 10	29
3.11	Figura casi d'uso 11	30
3.12	Figura casi d'uso 12	32
3.13	Figura casi d'uso 13	33
4.1	Mock up della parte di gestione della sessione	40
4.2	Mock up della parte di menù del ristorante	41
4.3	Mock up della parte di gestione ordini	42
4.4	Mock up della parte di autenticazione	43
4.5	Mock up della parte di gestione menù	44
4.6	Mock up della parte di gestione piatti	45
4.7	Mock up modificato dopo il feedback	46
4.8	Menù di navigazione	51
4.9	Maschera di login	52
4.10	Maschera di registrazione	53
4.11	Maschera di inserimento ordini	54
4.12	Interfaccia che dichiara le informazioni di cui si compone un piatto	55
4.13	Menù espandibile del piatto	56
4.14	Visualizzazione desktop	57
4.15	Menù con tema alternativo	58
4.16	Maschera di configurazione blacklist	59
4.17	Maschera di modifica piatto	60
4.18	Maschera di modifica menù	61
5.1	Dashboard del sito	64
5.2	Click map della pagina del menù	65
5.3	Move map della pagina del menù	66
5.4	Scroll map della pagina del menù	67
5.5	Recording di un utente desktop	68

5.6	Recording di un utente mobile	68
6.1	Ordine di tabulazione della maschera piatto	70
6.2	Simulazione dei vari tipi di daltonismo	70
6.3	Risultati dei test di Lighthouse	71
6.4	Widget di valutazione selezionato tramite tab	72

Elenco delle tabelle

3.1	Gestione sessione tavolo	34
3.2	Gestione ordini	35
3.3	Login e Registrazione - Cliente	35
3.4	Login e Registrazione - Gestore	36
3.5	Gestione menù	36
3.6	Gestione piatti	36
3.7	Guida d'aiuto	36
3.8	Requisiti qualitativi	37
3.9	Requisiti di vincolo	37
6.1	Copertura dei requisiti	73
7.1	Completamento degli obiettivi di stage	75

Capitolo 1

Introduzione

In questo capitolo viene fornita una breve introduzione al progetto e alla soluzione realizzata.

1.1 L'azienda

Sync Lab è una azienda che fornisce da oltre vent'anni soluzioni [IT](#)^[g] per diversi mercati quali: sanità, industria, energia, telecomunicazioni, finanza, trasporti e logistica¹.

Nasce a Napoli nel 2002 ed è continuamente cresciuta da allora: attualmente ha sei sedi in tutta Italia e oltre trecento dipendenti.

Sync Lab ha una lunga storia di collaborazione con diverse università a partire già dal 2003.

1.2 Introduzione al progetto

L'idea dello stage è quella di far fronte alla necessità di tracciare gli ordini in un ristorante di sushi nella formula *all you can eat*^[g].

In questa situazione i clienti ricevono un vasto menù con oltre cento piatti, ognuno identificato da un numero, e ordinano comunicando al cameriere l'elenco dei propri numeri.

Data la difficoltà di ricordarsi tutti i numeri da ordinare, di solito almeno una dozzina, si è costretti a prendere appunti. L'obiettivo è dunque agevolare il processo sia per i clienti che per il cameriere fornendo un'interfaccia più accattivante al processo di ordinazione.

Il progetto di stage proposto dall'azienda prevede l'analisi del problema sopracitato, la conseguente progettazione e realizzazione del *frontend*^[g].

Al fine di permettere un facile accesso da parte dei clienti al software la miglior soluzione è una Progressive WebApp, ovvero una pagina Web che una volta caricata si comporti allo stesso modo di un applicazione nativa.

Questo permette sia di evitare ai clienti la necessità di installare un applicazione sul

¹ *Sito di Sync Lab.* URL: <https://www.synclab.it/about.php>.

proprio dispositivo che di fornire un'unica applicazione in grado di essere eseguita su qualsiasi tipo di dispositivo e sistema operativo.

La soluzione software pensata da Sync Lab, oltre al *frontend*, prevede la realizzazione da parte di altri stagisti del *backend*^[8] in Java, di una parte di Machine Learning per identificare automaticamente i piatti tramite la fotocamera del proprio cellulare e infine la trasformazione della Progressive WebApp in applicativo nativo per Android e iOS.

1.3 Soluzione scelta

L'architettura individuata per il progetto è basata sui micro servizi il che permette una scalabilità orizzontale. Una parte, il *frontend*, si occupa di gestire l'interazione con l'utente fornendo un'interfaccia grafica per, ad esempio, visualizzare i menù ed effettuare gli ordini. La comunicazione con il *backend* avviene tramite chiamate *REST*^[9] e questo permette di ridurre la dipendenza tra *frontend* e *backend* a una mera interfaccia.

Si è scelto di realizzare il *frontend* tramite una Single Page WebApplication, ovvero di fornire tutte le pagine del sito tramite una singola pagina web che, tramite l'uso di Javascript, modifichi dinamicamente il suo contenuto. Questa soluzione permette di aggiornare solo alcune parti della pagina, risultando più efficiente e adatta in presenza di contenuti fortemente dinamici.

Per l'implementazione della WebApp è stato utilizzato Angular, un framework di Google realizzato al fine di facilitare lo sviluppo di Single Page WebApplication e di Progressive WebApp. La scelta di questo framework rispetto ad altri quali React o Vue è stata dettata principalmente dal maggiore utilizzo da parte dell'azienda di Angular.

Al fine di poter valutare l'effettiva usabilità del sito da parte degli utenti, senza incorrere in lunghe e costose analisi di user experience, è stato deciso di integrare un sistema di tracciamento e analisi del comportamento di quest'ultimi. Lo strumento di tracciamento usato è HotJar, con il quale l'azienda aveva già avuto qualche esperienza maturata a fronte di passati progetti di stage.

1.4 Descrizione del prodotto ottenuto

Maschere realizzate Al termine dello stage la WebApp implementa le maschere necessarie per:

- * visualizzare il menù;
- * ordinare;
- * gestire le sessioni di tavolo;
- * registrarsi;
- * autenticarsi;
- * recuperare la password;
- * inserire una blacklist di ingredienti da filtrare;

- * gestire in autonomia da parte del ristoratore i menù del proprio ristorante;
- * generare i codici *QR*^[g] da posizionare sui tavoli.

Tracciamento degli utenti L'integrazione tra Angular e HotJar è stata realizzata ed è quindi possibile raccogliere automaticamente statistiche e informazioni riguardo a come gli utenti interagiscono con la WebApp.

Questo permetterà, una volta in produzione, di monitorare quali migliorie possano essere realizzate per facilitare l'utilizzo per i clienti.

Inoltre sarà possibile fornire ai ristoratori utili informazioni riguardo alle preferenze dei clienti.

1.5 Principali problematiche

Durante lo svolgimento dell'attività di stage si sono presentate alcune criticità, ovvero:

- * l'assenza del backend che fornisca i dati ha richiesto l'utilizzo di un *mock*^[g] che lo simuli;
- * la vastità del menù, oltre cento piatti, ha richiesto una particolare attenzione alla realizzazione delle interfacce;
- * la necessità da parte dei gestori di personalizzare la grafica del proprio menù aumenta notevolmente la complessità del *frontend*;
- * la libreria HotJar per il tracciamento degli utenti richiede un server esposto pubblicamente, il che non è ottimale con un applicativo ancora in fase di sviluppo.

1.6 Strumenti utilizzati

Per la realizzazione del progetto sono stati utilizzati diversi strumenti e tecnologie:

Git

Git è un sistema di versionamento che permette la collaborazione di diversi sviluppatori allo stesso progetto.

L^AT_EX

L^AT_EX è un linguaggio dichiarativo per la stesura di documenti, è stato usato per la realizzazione dei documenti per il progetto di stage in cui sono state documentate le scelte effettuate.

Visual Studio Code

Editor di codice realizzato da Microsoft, fornisce molte feature atte a migliorare la qualità della vita degli sviluppatori, quali highlight della sintassi, integrazione con Git e segnalazione di molti errori già in fase di scrittura del codice senza necessità di compilare.

Figma

Figma è uno strumento che permette la facile realizzazione e condivisione di *mock up*^[g] di interfacce grafiche, permettendo lo studio sia dell'aspetto grafico che di user flow.

StopLight

StopLight permette la progettazione di *API*^[g] *REST*, generando automaticamente sia la documentazione appropriata che dei *mock* per il testing.

Typescript

Typescript è un linguaggio derivato da Javascript con lo scopo di implementare un sistema di tipi per segnalare possibili errori in fase di compilazione. Il codice Typescript viene compilato in codice Javascript il che permette di utilizzare questo linguaggio ovunque sia presente una macchina virtuale Javascript.

Angular

Angular è un framework di Google per la realizzazione di Single Page WebApplication. Basato sul linguaggio Typescript permette la realizzazione di *frontend* in maniera molto strutturata il che si presta bene a progetti di grandi dimensioni.

Angular Material

Angular Material è una libreria per Angular che fornisce molti componenti material già pronti velocizzando lo sviluppo e fornendo all'utente finale un'interfaccia Material familiare.

HotJar

HotJar è una libreria che fornisce funzionalità di tracciamento degli utenti sui siti web. Le informazioni raccolte, una volta analizzate, permettono di meglio comprendere come migliorare l'esperienza di questi ultimi nell'utilizzo del sito.

1.7 Organizzazione del testo

Il Capitolo 2 descrive gli obiettivi, il metodo di lavoro e la struttura dello stage.

Il Capitolo 3 approfondisce la fase di raccolta ed analisi dei requisiti.

Il Capitolo 4 sviluppa la fase di progettazione e codifica.

Il Capitolo 5 si occupa della fase di integrazione con la libreria HotJar.

Il Capitolo 6 descrive la fase di verifica e validazione del prodotto ottenuto.

Il Capitolo 7 trae le conclusioni finali sull'esperienza di stage.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- * gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- * per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*^[g].

Capitolo 2

Descrizione dello stage

In questo capitolo viene descritta la pianificazione e l'organizzazione della stage

2.1 Requisiti e obiettivi

Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- * *O* per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- * *D* per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- * *F* per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Obiettivi fissati

Gli obiettivi fissati erano i seguenti:

- * Obbligatori
 - *O-01*: acquisire competenze sul framework Angular;
 - *O-02*: acquisire competenze sulle tematiche di user experience, customer experience e user interface;
 - *O-03*: realizzare l'analisi funzionale del problema;
 - *O-04*: progettare le interfacce tramite Figma;
 - *O-05*: realizzare le maschere di login, registrazione ed inserimento ordini;
 - *O-06*: raggiungimento degli obiettivi richiesti in autonomia seguendo il cronoprogramma;
 - *O-07*: portare a termine le implementazioni previste con una percentuale di superamento pari o superiore al 80%.

* Desiderabili

- *D-01*: portare a termine le implementazioni previste con una percentuale di superamento pari al 100%;
- *D-02*: apportare un valore aggiunto al team soprattutto nelle fasi di analisi requisiti e disegno delle interfacce.

* Facoltativi

- *F-01*: descrivere mediante il tool StopLight tutti i servizi *REST* necessari alla piattaforma.

2.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Scarsa conoscenza delle tecnologie da utilizzare

Descrizione: Le tecnologie che si andranno a utilizzare non sono adeguatamente conosciute.

Soluzione: la prima parte del tirocinio consisterà nello studio e formazione riguardo alle suddette tecnologie.

2. Scarsa conoscenza del dominio di riferimento

Descrizione: La conoscenza della realtà di un ristorante di sushi è quasi del tutto assente.

Soluzione: si effettueranno le dovute interviste agli stakeholders e si collaborerà con l'altro stagista, partecipante al progetto, che ha esperienza lavorativa come cameriere nell'ambito.

3. Assenza di backend

Descrizione: Il *backend* verrà realizzato solo in seguito da altri tirocinanti.

Soluzione: si useranno degli strumenti quali StopLight per effettuare il *mock* del *backend* e degli *endpoint*^[5] *REST*.

2.3 Pianificazione

Lo stage è stato strutturato in due parti della durata di un mese ciascuna:

- * nella prima parte viene effettuata l'analisi del problema, la progettazione e lo studio delle tecnologie;
- * nella seconda parte viene realizzata parte della soluzione individuata.

Cronoprogramma settimanale

* **Prima Settimana - Formazione UX (40 ore)**

- presentazione strumenti di lavoro per la condivisione del materiale di studio e per la gestione dell'avanzamento del percorso;

- condivisione scaletta degli argomenti;
 - veloce panoramica sulle tecnologie Agile/Scrum;
 - studio sul significato e sulla distinzione tra user experience, customer experience e user interface;
 - studio sui principi di psicologia cognitiva, percezione visiva e gerarchia visuale;
 - studio sul copy writing e story telling;
 - studio su user centered design, user journey, empathy map e user flow;
 - analisi dei competitor e delle personas;
 - studio sul mouse tracking, click map e heat map;
 - studio sul responsive web design e sull’adaptive web design;
 - incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare.
- * **Seconda Settimana - Formazione Figma e StopLight (40 ore)**
- studio tool Figma e StopLight;
 - analisi funzionale e disegno delle interfacce grafiche mediante Figma.
- * **Terza Settimana - Formazione Javascript e Typescript (40 ore)**
- ripasso linguaggio Javascript;
 - studio del linguaggio TypeScript.
- * **Quarta Settimana - Formazione Node, Angular e HotJar (40 ore)**
- studio piattaforma NodeJS e AngularCLI;
 - studio framework Angular;
 - studio della libreria grafica HotJar e possibile integrazione con Angular.
- * **Quinta Settimana - Implementazione login e registrazione (40 ore)**
- implementazione maschere di login e registrazione utente.
- * **Sesta Settimana - Implementazione ordini utente (40 ore)**
- implementazione maschera inserimento ordine dell’utente.
- * **Settima Settimana - Integrazione HotJar (40 ore)**
- integrazione libreria HotJar;
- * **Ottava Settimana - Conclusione (40 ore)**
- termine integrazioni e collaudo finale.

2.4 Organizzazione dello stage

Smart working

L’azienda permette ai propri dipendenti di lavorare in smart working. In particolare il tutor aziendale era presente in azienda due giorni a settimana. Data l’importanza del lavoro in sede per fare esperienza, scopo ultimo dello stage, è stato scelto di essere presente tutte le volte fosse presente anche il tutor aziendale.

Lavoro in sede

In ufficio i colleghi si sono sempre rivelati disponibili a condividere la loro esperienza lavorativa in un ambiente cordiale e socievole.

Meeting settimanali

Almeno una volta a settimana si sono svolti incontri in sede per aggiornare gli stakeholders sullo stato di avanzamento del progetto e meglio coordinare le attività.

Discord

L'azienda ha messo a disposizione il server Discord aziendale, nota applicazione di messaggistica, il che ha permesso di rimanere sempre aggiornati sul lavoro svolto. Inoltre in diverse occasioni è stato possibile ricevere feedback da persone presenti sul server che appartengono a varie sedi di Sync Lab sparse per l'Italia.

Questo strumento di collaborazione si è rivelato molto utile e versatile per lo svolgimento del tirocinio.

Interazione con Luciano Wu

Luciano Wu è il collega con cui ho sviluppato la WebApp. Entrambi siamo iscritti all'Università degli Studi di Padova.

Per coordinare le operazioni si è utilizzato sia Discord che Telegram e la condivisione del codice è avvenuta tramite un repository privato su GitHub al quale ha avuto accesso, oltre a noi due, il tutor aziendale. Ogni modifica al codice è stata realizzata da uno e verificata dall'altro prima di essere accettata nella branch principale del repository.

Capitolo 3

Analisi dei requisiti

In questo capitolo viene descritta la fase di analisi dei requisiti che è stata effettuata per la realizzazione del progetto.

3.1 Confronto con gli stakeholders

Per prima cosa, al fine di aver chiari i requisiti, è stata effettuata una discussione con il proponente, ovvero l'azienda Sync Lab.

Chiariti gli obiettivi di massima mi sono confrontato con l'altro stagista assegnato al progetto, Luciano Wu, per definire meglio gli obiettivi e i bisogni che la soluzione avrebbe dovuto coprire.

Abbiamo dunque presentato la nostra proposta al committente che si è detto soddisfatto e ci ha autorizzato a procedere.

In seguito siamo stati informati che un'altra azienda di Padova aveva sviluppato un prototipo di un'applicativo con lo stesso scopo. Il tutor aziendale ha organizzato un colloquio coi responsabili di quest'altra soluzione che ci hanno informati delle criticità che hanno riscontrato aiutandoci a meglio progettare la nostra soluzione.

3.2 Personas

Per meglio immedesimarsi nei possibili utenti e tener traccia dei loro bisogni sono state creati alcuni stereotipi di cliente tipo usando la metodologia delle "personas", come ad esempio:

- * un cliente che non era mai stato al sushi, in compagnia di amici;
- * un cliente saltuario, in un uscita di coppia;
- * un cliente habitué;
- * un cliente che si reca al sushi durante la pausa pranzo e ha intenzione di ordinare dall'ufficio prima di arrivare.

3.3 Entità

Definite le personas si è reso necessario documentare le entità di dominio per meglio tener traccia dei dati necessari alla soluzione. Il risultato di questa operazione è stato il seguente:

Menù

Un menù è una lista di piatti e di fasce orarie di validità.

Piatto

Ogni piatto è caratterizzato da:

1. numero: numero di piatto (non necessariamente sequenziale);
2. variante: stringa della variante del piatto;
3. nome: nome del piatto;
4. prezzo: prezzo del piatto per ordini alla carta;
5. immagine: singola immagine che mostra il piatto;
6. testo alternativo per l'immagine: per gli utenti con disabilità;
7. allergeni: lista dei possibili allergeni;
8. sezione: categoria di piatto;
9. limite per persona: limite ordinabile per persona nella sessione *all you can eat* (NB: lo stesso piatto può avere limiti diversi a seconda del menù);
10. valutazione: media delle valutazioni dei clienti;
11. ingredienti: elenco degli ingredienti;
12. popolare: se è tra i più ordinati;
13. consigliato: se è consigliato dal ristorante.

Cliente registrato

Un cliente del ristorante che si è registrato sulla nostra app; si deve tener traccia di:

1. email;
2. hash della password;
3. piatti preferiti;
4. cronologia piatti;
5. valutazione data ai singoli piatti;
6. punti fedeltà accumulati per uno specifico ristorante.

Gestore

Un gestore di un ristorante; si deve tener traccia di:

1. email;
2. hash della password;
3. i suoi menù (un ristorante di sushi, solitamente, ha almeno un menù pranzo e uno cena separati);
4. i suoi patti.

3.4 Casi d'uso - Cliente

Individuate le entità si è proceduto alla stesura dei casi d'uso lato cliente, divisi per classi di funzionalità per facilitarne la numerazione e l'estensione.

Segue l'elenco dell'ultima versione individuata dei casi d'uso divisa per macro categorie di funzionalità della parte cliente.

3.4.1 Attori

- * **Cliente:** cliente riconosciuto o meno;
- * **cliente non riconosciuto:** cliente del ristorante che non ha effettuato il login;
- * **cliente riconosciuto:** cliente del ristorante che ha effettuato il login;
- * **cameriere:** cameriere del ristorante che prende le ordinazioni.

3.4.2 Gestione sessione tavolo

Descrizione informale

Per aggregare gli ordini, tutti i clienti di un tavolo devono partecipare alla stessa sessione. Questo richiede la possibilità per un cliente di creare una sessione alla quale gli altri clienti si uniranno.

Per unirsi alla sessione sono previste due modalità: una stringa, che identifica la sessione, da inserire nell'applicativo e un più agevole codice *QR* per dispositivi dotati di fotocamera.

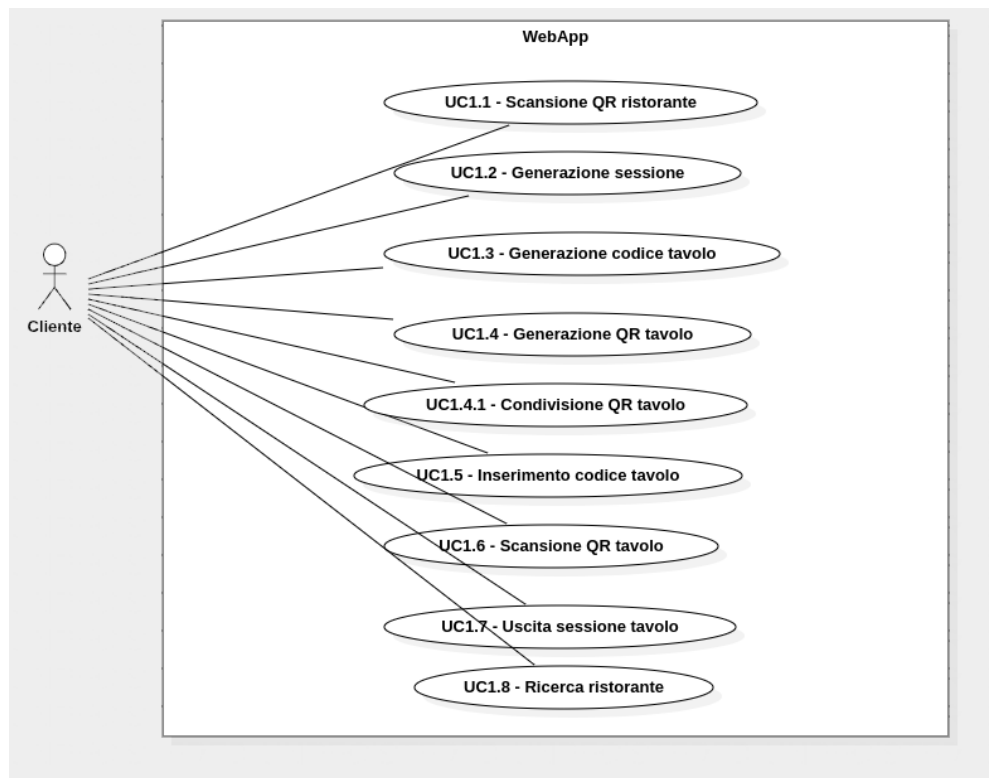


Figura 3.1: Figura casi d'uso 1

UC1.1 - Scansione *QR* ristorante

Attori primari: cliente.

Precondizioni: il cliente è seduto al tavolo e vuole creare la sessione per il tavolo.

Post-condizioni: il cliente è nella sezione specifica di un ristorante.

Scenario principale:

1. il cliente scansiona il *QR* che lo indirizza alla pagina relativa al ristorante.

Scenario secondario:

1. il cliente scansiona il *QR* di un menù con un orario sbagliato;
2. il cliente viene informato che il menù non è valido in questa fascia oraria.

UC1.2 - Generazione sessione

Attori primari: cliente.

Precondizioni: il cliente è nella sezione specifica di un ristorante.

Post-condizioni: il cliente è inserito in una nuova sessione di tavolo.

Scenario principale:

1. il cliente preme genera nuovo tavolo;
2. il cliente è inserito in una nuova sessione.

UC1.3 - Generazione codice tavolo

Attori primari: cliente.

Precondizioni: il cliente è in una sessione per un tavolo.

Post-condizioni: il cliente genera un codice del tavolo che individua la sessione.

Scenario principale:

1. il cliente vede un codice che permetterà ad altri clienti di partecipare alla stessa sessione.

UC1.4 - Generazione *QR* tavolo

Attori primari: cliente.

Precondizioni: il cliente è in una sessione per un tavolo.

Post-condizioni: il cliente genera un codice *QR* del tavolo che individua la sessione.

Scenario principale:

1. il cliente vede un codice *QR* che permetterà ad altri clienti di partecipare alla stessa sessione.

UC1.4.1 - Condivisione *QR* tavolo

Attori primari: cliente.

Precondizioni: il cliente ha generato un *QR* per la sessione del tavolo.

Post-condizioni: il cliente condivide il *QR*.

Scenario principale:

1. il cliente genera il *QR*;
2. il cliente lo condivide tramite la funzione "share" nativa della piattaforma.

UC1.5 - Inserimento codice tavolo

Attori primari: cliente.

Precondizioni: il cliente è nella sezione specifica di un ristorante.

Post-condizioni: il cliente è inserito nella sessione del tavolo.

Scenario principale:

1. il cliente inserisce il codice fornito da un suo commensale;
2. il cliente si unisce alla stessa sessione.

UC1.6 - Scansione *QR* tavolo

Attori primari: cliente.

Precondizioni: il cliente è nella sezione specifica di un ristorante.

Post-condizioni: il cliente è inserito nella sessione del tavolo.

Scenario principale:

1. il cliente scansiona il codice *QR* fornito da un suo commensale;
2. il cliente si unisce alla stessa sessione.

UC1.7 - Uscita sessione tavolo

Attori primari: cliente.

Precondizioni: il cliente è nella sezione specifica di un tavolo.

Post-condizioni: il cliente esce dalla sessione del tavolo.

Scenario principale:

1. il cliente ha terminato e vuole uscire dalla sessione;
2. il cliente esce dalla sessione.

UC1.8 - Ricerca ristorante

Attori primari: cliente.

Precondizioni: il cliente è nella WebApp.

Post-condizioni: il cliente trova il ristorante.

Scenario principale:

1. il cliente è alla ricerca di un ristorante;
2. la WebApp gli mostra i ristoranti nella zona.

3.4.3 Gestione ordini**Descrizione informale**

Per la gestione degli ordini è indispensabile innanzitutto fornire il menù con tutte le informazioni di ogni piatto.

Deve essere poi possibile ordinare i vari piatti che andranno in una specifica lista. Queste liste, personali per ogni cliente al tavolo, devono poter essere aggregate e visualizzabili come un'unica lista complessiva da fornire al cameriere.

Una volta ordinati al cameriere, i piatti andranno spostati su una terza lista che permetterà di tener traccia dei piatti in consegna e aiuterà i commensali a ricordare, tramite le foto dei piatti, quali hanno ordinato.

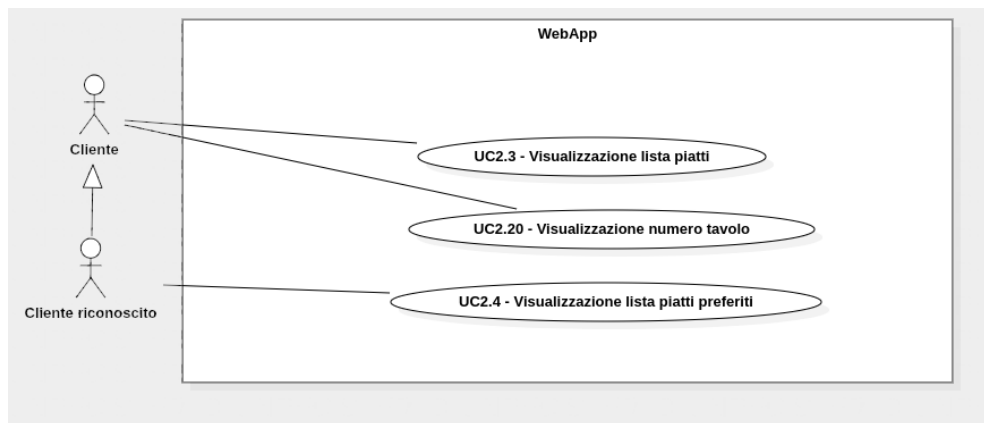


Figura 3.2: Figura casi d'uso 2

UC2.3 - Visualizzazione lista piatti

Attori primari: cliente.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente ha visto la lista completa dei piatti.

Scenario principale:

1. il cliente visualizza la lista completa dei piatti ordinata per numero e divisa per sezioni.

UC2.4 - Visualizzazione lista piatti preferiti

Attori primari: cliente riconosciuto.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente ha visto dei piatti nella sua lista preferiti.

Scenario principale:

1. il cliente visualizza la lista dei suoi piatti preferiti.

UC2.20 - Visualizzazione numero tavolo

Attori primari: cliente.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente ha visto il numero del tavolo.

Scenario principale:

1. il cliente ha finito di mangiare;
2. il cliente va alla cassa e gli viene chiesto il numero del tavolo;
3. il cliente tramite l'app si ricorda del numero.

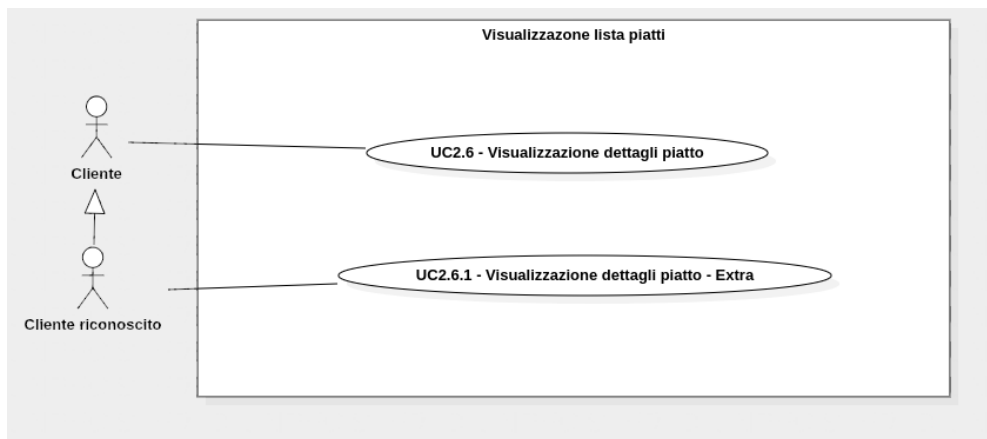


Figura 3.3: Figura casi d'uso 3

UC2.6 - Visualizzazione dettagli piatto

Attori primari: cliente.

Precondizioni: il cliente è in una lista piatti sopra citata.

Post-condizioni: il cliente vede i dettagli di un piatto.

Scenario principale:

1. il cliente visualizza i dettagli di un piatto:
 - * numero;
 - * nome;
 - * variante;
 - * immagine;
 - * testo alternativo per l'immagine;
 - * allergeni;
 - * limite ordinabile per persona;
 - * ingredienti;
 - * attualmente ordinati da lui in questa sessione;
 - * valutazione media dei clienti (0-5 stelle);
 - * molteplicità ordinata nell'ordine corrente;
 - * se è consigliato;
 - * se è popolare.

UC2.6.1 - Visualizzazione dettagli piatto - Extra

Attori primari: cliente **ricosciuto**.

Precondizioni: il cliente è in una lista piatti sopra citata.

Post-condizioni: il cliente vede i dettagli di un piatto.

Scenario principale:

1. il cliente visualizza i dettagli di un piatto:

- * la sua valutazione per il piatto, se presente;
- * se è presente nella sua lista preferiti;
- * ultima volta che è stato ordinato dal cliente.

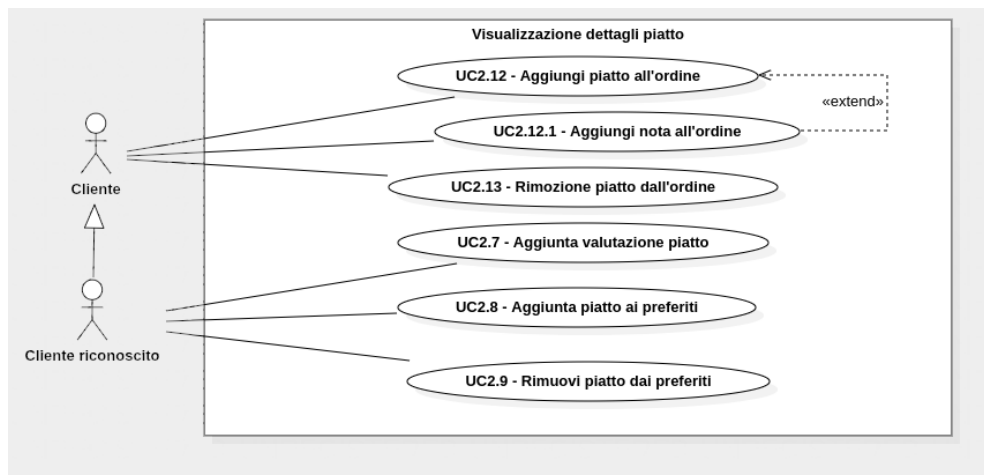


Figura 3.4: Figura casi d'uso 4

UC2.7 - Aggiunta valutazione piatto

Attori primari: cliente riconosciuto.

Precondizioni: il cliente vede i dettagli di un piatto.

Post-condizioni: il cliente ha valutato il piatto.

Scenario principale:

1. il cliente aggiunge una valutazione la piatto in un range da 0 a 5 stelle.

UC2.8 - Aggiunta piatto ai preferiti

Attori primari: cliente riconosciuto.

Precondizioni: il cliente vede i dettagli di un piatto che non è nella sua lista preferiti.

Post-condizioni: il cliente ha aggiunto il piatto alla lista dei preferiti.

Scenario principale:

1. il cliente aggiunge un piatto ai preferiti.

UC2.9 - Rimuovi piatto dai preferiti

Attori primari: cliente riconosciuto.

Precondizioni: il cliente vede i dettagli di un piatto che è nella sua lista preferiti.

Post-condizioni: il cliente ha rimosso il piatto alla lista dei preferiti.

Scenario principale:

1. il cliente rimuove un piatto ai preferiti.

UC2.12 - Aggiungi piatto all'ordine**Attori primari:** cliente.**Precondizioni:** il cliente vede i dettagli di un piatto.**Post-condizioni:** il cliente ha aggiunto il piatto all'ordine corrente.**Scenario principale:**

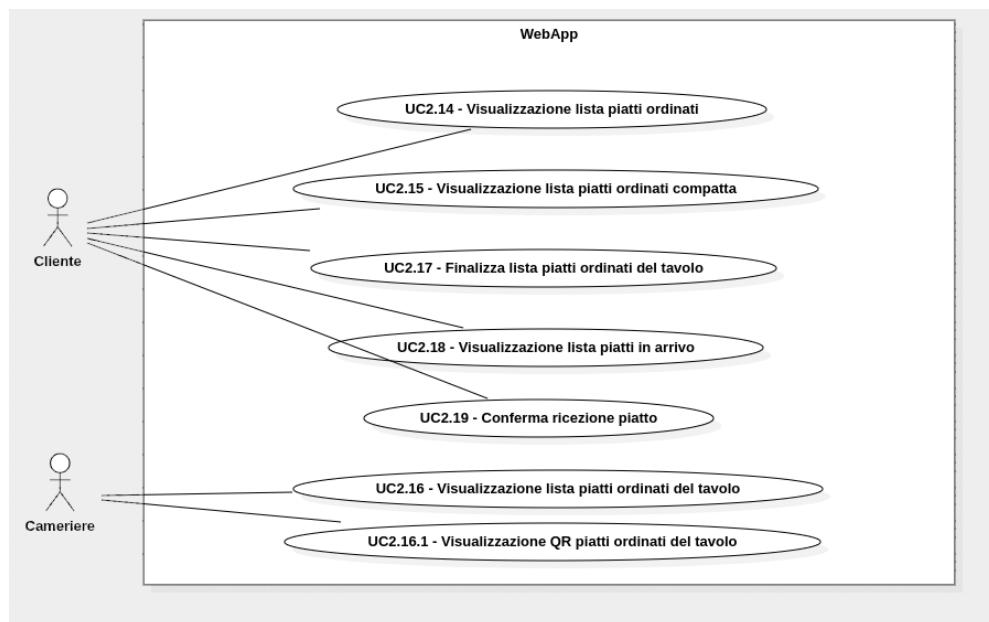
1. il cliente aggiunge un piatto all'ordine corrente (se fatto più volte aumenta la molteplicità dell'ordine).

UC2.12.1 - Aggiungi nota all'ordine**Attori primari:** cliente.**Precondizioni:** il cliente vede i dettagli di un piatto.**Post-condizioni:** il cliente ha aggiunto il piatto all'ordine corrente.**Scenario principale:**

1. il cliente specifica una nota per il ristorante;
2. il cliente aggiunge un piatto all'ordine corrente (se fatto più volte aumenta la molteplicità dell'ordine).

UC2.13 - Rimozione piatto dall'ordine**Attori primari:** cliente.**Precondizioni:** il cliente vede i dettagli di un piatto.**Post-condizioni:** il cliente ha rimosso il piatto dall'ordine corrente.**Scenario principale:**

1. il cliente rimuove un piatto all'ordine corrente.

**Figura 3.5:** Figura casi d'uso 5

UC2.14 - Visualizzazione lista piatti ordinati

Attori primari: cliente.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente vede la lista dei piatti che ha ordinato.

Scenario principale:

1. il cliente vede la lista dei piatti che ha ordinato.

UC2.15 - Visualizzazione lista piatti ordinati compatta

Attori primari: cliente.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente vede la lista dei numeri dei piatti che ha ordinato.

Scenario principale:

1. il cliente vede la lista dei numeri dei piatti che ha ordinato.

UC2.16 - Visualizzazione lista piatti ordinati del tavolo

Attori primari: cameriere.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cameriere vede la lista dei numeri dei piatti che l'intero tavolo ha ordinato.

Scenario principale:

1. il cliente vede la lista dei numeri dei piatti che l'intero tavolo ha ordinato;
2. il cliente mostra l'elenco al cameriere;
3. il cameriere inserisce l'elenco nel sistema di comande già in possesso del ristorante.

UC2.16.1 - Visualizzazione *QR* piatti ordinati del tavolo

Attori primari: cameriere.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cameriere vede il *QR* che rappresenta l'ordine.

Scenario principale:

1. il cliente vede il *QR* che rappresenta l'ordine;
2. il cliente mostra il *QR* al cameriere;
3. il cameriere scansiona il *QR* che inserirà l'ordine nel sistema di comande del ristorante.

UC2.17 - Finalizza lista piatti ordinati del tavolo

Attori primari: cliente che ha creato il tavolo.

Precondizioni: il cliente vede la lista dei numeri dei piatti che l'intero tavolo ha ordinato.

Post-condizioni: il cliente finalizza la lista dei numeri dei piatti che l'intero tavolo ha ordinato.

Scenario principale:

1. il cliente conferma la lista che passerà dallo stato "ordinato" allo stato "in arrivo".

UC2.18 - Visualizzazione lista piatti in arrivo

Attori primari: cliente.

Precondizioni: il cliente è nella sessione specifica di un tavolo.

Post-condizioni: il cliente vede la lista dei piatti tavolo ha ordinato e sono in attesa di essere consegnati.

Scenario principale:

1. il cliente vede la lista dei piatti che ha ordinato e sono in attesa di essere recapitati dal cameriere.

UC2.19 - Conferma ricezione piatto

Attori primari: cliente.

Precondizioni: il cliente è nella sessione lista piatti in arrivo.

Post-condizioni: il cliente ha rimosso un piatto dalla lista.

Scenario principale:

1. il cliente ha ricevuto un piatto dal cameriere;
2. lo rimuove dalla sua lista per tenere traccia di quali piatti devono essere ancora effettivamente consegnati.

3.4.4 Area personale**Descrizione informale**

L'area personale permette di gestire le informazioni personali di ogni utente, quali punti fedeltà e ingredienti indesiderati. Per fare ciò sono necessarie le classiche feature di login e registrazione.

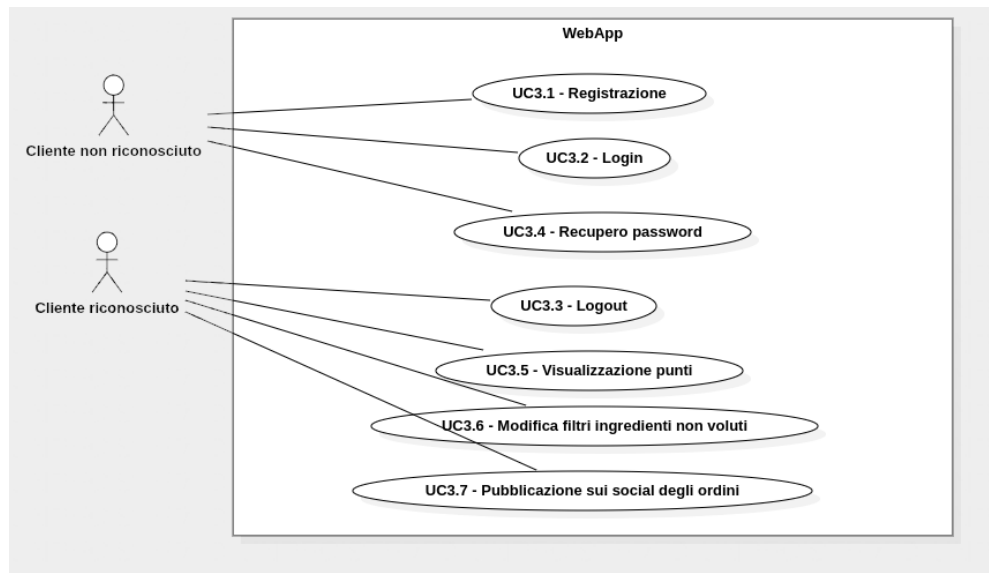


Figura 3.6: Figura casi d'uso 6

UC3.1 - Registrazione

Attori primari: cliente non riconosciuto.

Precondizioni: Il cliente è nella WebApp.

Post-condizioni: Il cliente ha un account registrato.

Scenario principale:

1. il cliente vuole farsi un account;
2. il cliente inserisce un username;
3. il cliente inserisce la password;
4. il cliente reinserte la password;
5. il cliente si registra con successo.

UC3.2 - Login

Attori primari: cliente non riconosciuto.

Precondizioni: Il cliente si è registrato precedentemente nella WebApp.

Post-condizioni: Il cliente viene riconosciuto dalla WebApp.

Scenario principale:

1. il cliente inserisce un username;
2. il cliente inserisce la password;
3. il cliente si logga con successo.

UC3.3 - Logout

Attori primari: cliente riconosciuto.

Precondizioni: Il cliente si è loggato precedentemente nella WebApp.

Post-condizioni: Il cliente viene sloggato dalla WebApp.

Scenario principale:

1. Il cliente si slogga con successo.

UC3.4 - Recupero password

Attori primari: cliente non riconosciuto.

Precondizioni: il cliente si è registrato precedentemente nella WebApp.

Post-condizioni: il cliente riceve un email di recupero password.

Scenario principale:

1. il cliente si è dimenticato la password;
2. il cliente riceve un'email di recupero;
3. il cliente imposta una nuova password.

UC3.5 - Visualizzazione punti

Attori primari: cliente riconosciuto.

Precondizioni: il cliente si è loggato.

Post-condizioni: il cliente vede i suoi punti accumulati nei vari ristoranti.

Scenario principale:

1. il cliente si è loggato;
2. il cliente vede i suoi punti nei vari ristoranti.

UC3.6 - Modifica filtri ingredienti non voluti

Attori primari: cliente riconosciuto.

Precondizioni: il cliente si è loggato.

Post-condizioni: il cliente ha aggiornato i filtri dei piatti.

Scenario principale:

1. il cliente si è loggato;
2. il cliente aggiorna i filtri ingredienti non voluti;
3. l'app non mostrerà i piatti contenenti quegli ingredienti.

UC3.7 - Pubblicazione sui social degli ordini

Attori primari: cliente riconosciuto.

Precondizioni: il cameriere ha confermato l'ordine.

Post-condizioni: il cliente ha pubblicato sui propri social i piatti ordinati.

Scenario principale:

1. il cliente ordina;
2. il cameriere approva gli ordini;
3. il cliente si vanta sui social dell'abbuffata.

3.5 Casi d'uso - Gestore

Si è poi proceduto a redigere i casi d'uso lato gestore.

Segue l'elenco dell'ultima versione individuata dei casi d'uso diviso per macro categorie di funzionalità della parte gestore ristorante.

3.5.1 Attori

- * **Gestore:** gestore del ristorante che non ha effettuato il login;
- * **gestore riconosciuto:** gestore del ristorante che ha effettuato il login;
- * **cameriere:** cameriere del ristorante che ha effettuato il login con l'account del gestore.

3.5.2 Area personale

Descrizione informale

L'area personale del gestore del ristorante permette la gestione dei punti fedeltà lato ristoratore e la conferma degli ordini da parte del cameriere. Anche qui sono necessarie le feature di login e registrazione.

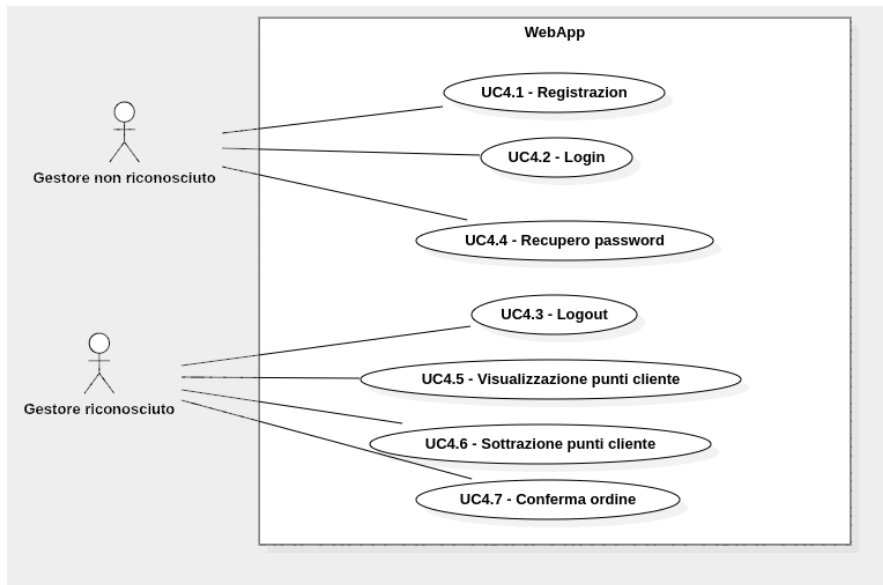


Figura 3.7: Figura casi d'uso 7

UC4.1 - Registrazione

Attori primari: gestore non riconosciuto.

Precondizioni: il gestore è nella WebApp.

Post-condizioni: il gestore ha un account registrato.

Scenario principale:

1. il gestore vuole farsi un account;
2. il gestore inserisce un username;
3. il gestore inserisce la password;
4. il gestore reinserisce la password;
5. il gestore si registra con successo.

UC4.2 - Login

Attori primari: gestore non riconosciuto.

Precondizioni: il gestore si è registrato precedentemente nella WebApp.

Post-condizioni: il gestore viene riconosciuto dalla WebApp.

Scenario principale:

1. il gestore inserisce un username;

2. il gestore inserisce la password;
3. il gestore si logga con successo.

UC4.3 - Logout

Attori primari: gestore riconosciuto.

Precondizioni: il gestore si è loggato precedentemente nella WebApp.

Post-condizioni: il gestore viene sloggato dalla WebApp.

Scenario principale:

1. il gestore si slogga con successo.

UC4.4 - Recupero password

Attori primari: gestore non riconosciuto.

Precondizioni: il gestore si è registrato precedentemente nella WebApp.

Post-condizioni: il gestore riceve un email di recupero password.

Scenario principale:

1. il gestore si è dimenticato la password;
2. il gestore riceve un'email di recupero;
3. il gestore imposta una nuova password.

UC4.5 - Visualizzazione punti cliente

Attori primari: gestore riconosciuto.

Precondizioni: il gestore si è loggato.

Post-condizioni: il gestore vede i suoi punti accumulati dal cliente.

Scenario principale:

1. il cliente si è loggato;
2. il cliente mostra un codice al ristoratore;
3. il gestore loggato inserisce il codice;
4. il gestore vede i punti accumulati dal cliente.

UC4.6 - Sottrazione punti cliente

Attori primari: gestore riconosciuto.

Precondizioni: il gestore si è loggato.

Post-condizioni: il gestore vede i suoi punti accumulati dal cliente.

Scenario principale:

1. il cliente si è loggato;
2. il cliente mostra un codice al ristoratore;
3. il gestore loggato inserisce il codice;
4. il gestore sottrae dei punti al cliente;
5. il gestore fornisce un bonus fedeltà al cliente.

UC4.7 - Conferma ordine

Attori primari: cameriere.

Precondizioni: il cliente mostra il *QR* riassuntivo al cameriere.

Post-condizioni: l'ordine è considerato valido e inserito nel sistema informativo del ristorante.

Scenario principale:

1. il mostra il *QR* riassuntivo dell'ordine al cameriere;
2. il cameriere lo scansione;
3. il cameriere approva l'ordine.

3.5.3 Gestione menù**Descrizione informale**

Il ristoratore ha necessità di gestire i propri menù e generare i codici *QR* da posizionare sui tavoli.

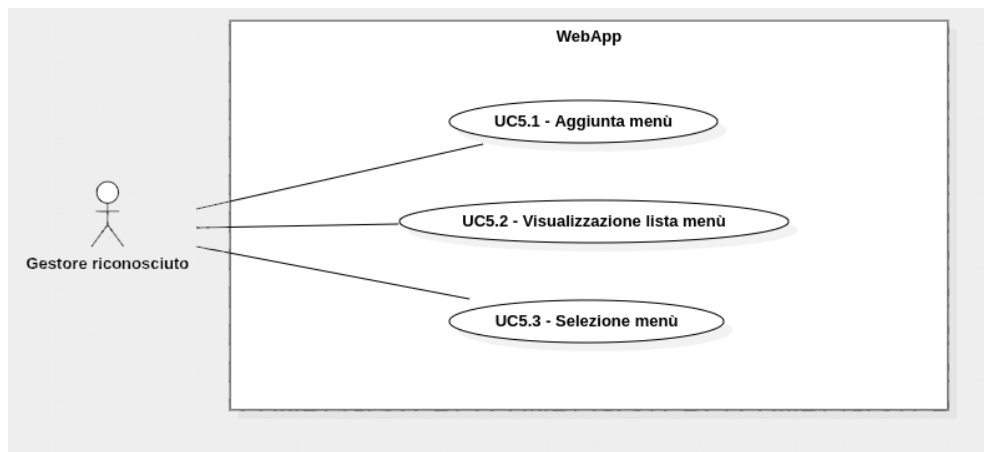


Figura 3.8: Figura casi d'uso 8

UC5.1 - Aggiunta menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore è nella WebApp.

Post-condizioni: il gestore ha aggiunto un nuovo menù.

Scenario principale:

1. il gestore specifica l'orario del menù;
2. il gestore inserisce un nuovo menù.

UC5.2 - Visualizzazione lista menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore è nella WebApp.

Post-condizioni: il gestore vede la lista dei suoi menù.

Scenario principale:

1. il gestore vede la lista dei suoi menù.

UC5.3 - Selezione menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore vede la lista dei suoi menù.

Post-condizioni: il gestore ha selezionato un menù.

Scenario principale:

1. il gestore vede la lista dei suoi menù;
2. il gestore seleziona un menù.

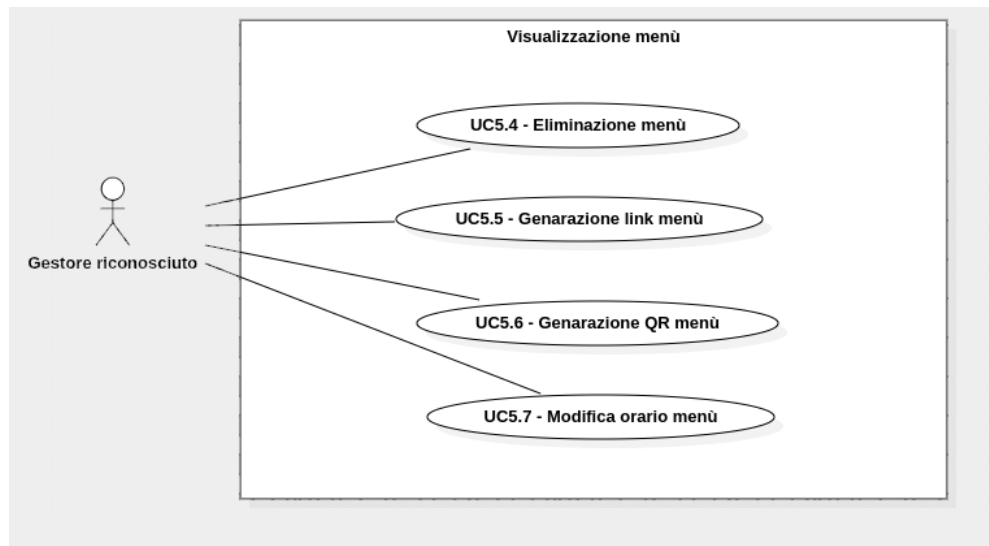


Figura 3.9: Figura casi d'uso 9

UC5.4 - Eliminazione menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha eliminato il menù.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore clicca elimina;
3. il gestore conferma la sua intenzione;
4. Il menù viene eliminato.

UC5.5 - Generazione link menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha generato il link al menù.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore genera il link per i clienti.

UC5.6 - Generazione *QR* menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha generato il *QR* al menù.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore genera il *QR* per i clienti.

UC5.7 - Modifica orario menù

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha modificato l'orario del menù.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore modifica l'orario del menù.

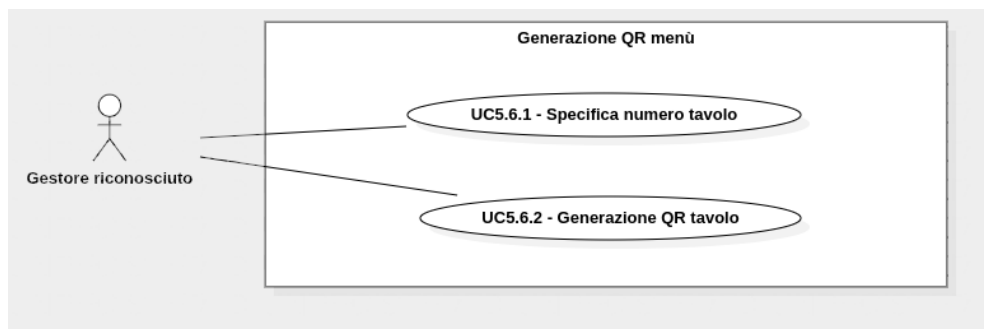


Figura 3.10: Figura casi d'uso 10

UC5.6.1 - Specifica numero tavolo

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha specificato il numero tavolo con cui verrà generato il *QR*.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore specifica con che numero tavolo verrà generato il *QR*.

UC5.6.2 - Generazione *QR* tavolo

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù e specificato il numero del tavolo.

Post-condizioni: il gestore ha generato il *QR* al menù che specifica anche il tavolo.

Scenario principale:

1. il gestore ha selezionato un menù;
2. il gestore ha specificato il numero del tavolo;
3. il gestore genera il *QR*;
4. il gestore stampa il *QR* (e lo plastifica);
5. il gestore applica il *QR* sul tavolo.

3.5.4 Gestione piatti

Descrizione informale

Deve essere possibile per un ristoratore inserire i propri piatti nei suoi menù, con tutte le informazioni necessarie.

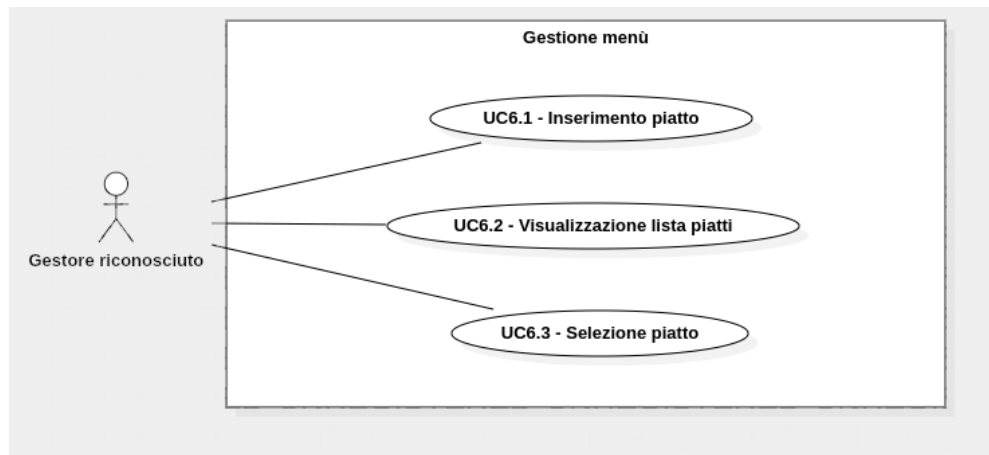


Figura 3.11: Figura casi d'uso 11

UC6.1 - Inserimento piatto

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore ha aggiunto un piatto al menù.

Scenario principale:

1. il gestore inserisce un piatto specificando:

- * numero;
- * nome;
- * variante;
- * immagine;
- * testo alternativo per l'immagine;
- * allergeni;
- * ingredienti;
- * sezione;
- * limite per persona.

UC6.2 - Visualizzazione lista piatti

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un menù.

Post-condizioni: il gestore vede la lista dei piatti relativi al menù.

Scenario principale:

1. il gestore vede la lista dei piatti relativi al menù.

UC6.3 - Selezione piatto

Attori primari: gestore riconosciuto.

Precondizioni: il gestore vede la lista dei piatti.

Post-condizioni: il gestore ha selezionato un piatto.

Scenario principale:

1. il gestore vede la lista dei piatti;

2. il gestore seleziona un piatto.

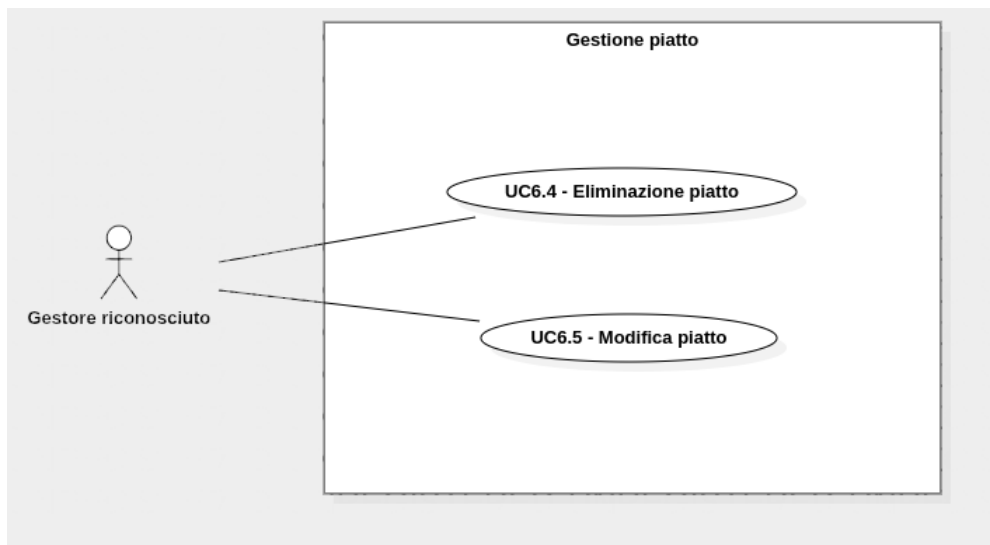


Figura 3.12: Figura casi d'uso 12

UC6.4 - Eliminazione piatto

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un piatto.

Post-condizioni: il gestore ha eliminato il piatto.

Scenario principale:

1. il gestore ha selezionato un piatto;
2. il gestore clicca elimina;
3. il gestore conferma la sua intenzione;
4. il piatto viene eliminato.

UC6.5 - Modifica piatto

Attori primari: gestore riconosciuto.

Precondizioni: il gestore ha selezionato un piatto.

Post-condizioni: il gestore ha modificato il piatto.

Scenario principale:

1. il gestore ha selezionato un piatto;
2. il gestore modifica le informazioni del piatto;
3. Il gestore conferma la sua intenzione;
4. Il piatto viene modificato.

3.5.5 Guida d'aiuto

Descrizione informale

La WebApp deve fornire una guida in linea, sia per il cliente che per il gestore, che ne illustri il funzionamento.

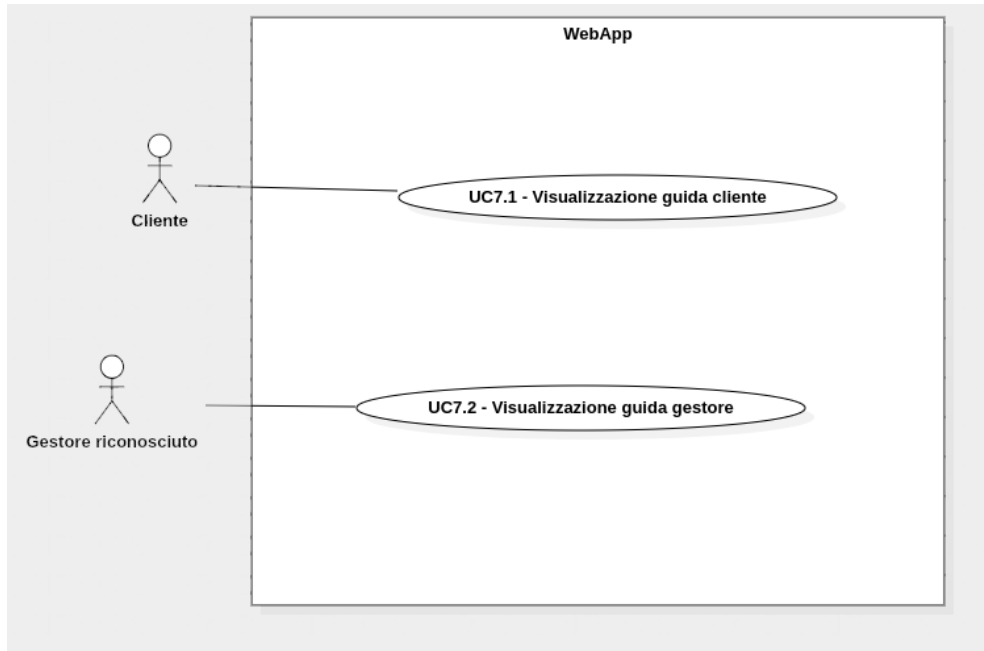


Figura 3.13: Figura casi d'uso 13

UC7.1 - Visualizzazione guida cliente

Attori primari: cliente.

Precondizioni: il cliente è nella WebApp.

Post-condizioni: il cliente ha visto la guida d'aiuto.

Scenario principale:

1. Il cliente visualizza la guida in linea d'aiuto il cliente.

UC7.2 - Visualizzazione guida gestore

Attori primari: gestore riconosciuto.

Precondizioni: il gestore è nella WebApp.

Post-condizioni: il gestore ha visto la guida d'aiuto.

Scenario principale:

1. il gestore visualizza la guida in linea d'aiuto per il gestore.

3.6 Tracciamento dei requisiti

Completata la redazione dei casi d'uso si è proceduto alla fase di tracciamento dei requisiti. A ogni requisito individuato è stato assegnato un codice univoco in cui:

- * la prima lettera è **R** per requisito;
- * la seconda lettera indica il tipo di requisito:
 - **F** per i requisiti funzionali;
 - **Q** per i requisiti qualitativi;
 - **V** per i requisiti di vincolo;
- * un numero individua a quale classe dei casi d'uso fa riferimento;
- * un numero progressivo individua il requisito all'interno della classe.

3.6.1 Requisiti funzionali

Codice	Descrizione	Rilevanza	Fonti
RF1.1	Si deve poter accedere alla sezione di un ristorante tramite URL	Obbligatorio	UC1.1
RF1.2	Il sistema deve poter tenere traccia di diverse sessioni di tavolo	Obbligatorio	UC1.2
RF1.3	Il sistema deve poter identificare la sessione di tavolo con un codice	Obbligatorio	UC1.3, UC1.4
RF1.4	Il codice deve essere randomico per evitare possibili attacchi (e.g. entrare in sessioni altrui)	Desiderabile	UC1.3, UC1.4
RF1.5	Si deve poter generare un codice <i>QR</i> con un URL	Obbligatorio	UC1.4
RF1.6	Si deve poter inserire il codice nella WebApp per unirsi alla sessione di tavolo	Obbligatorio	UC1.5
RF1.7	Si deve poter unirsi alla sessione di tavolo tramite URL	Obbligatorio	UC1.4, UC1.6
RF1.8	Si deve poter lasciare la sessione di tavolo	Obbligatorio	UC1.7
RF1.9	Il sistema deve liberare le risorse allocate per gestire le sessioni concluse	Obbligatorio	UC1.7
RF1.10	Il sistema deve concludere le sessioni forzatamente dopo un lungo periodo di inattività (i.e. 12 ore)	Obbligatorio	UC1.7
RF1.11	Il sistema deve poter associare il numero di tavolo del ristorante alla sessione	Obbligatorio	UC2.20
RF1.12	Il sistema deve accedere all'ora corrente per verificare se la fascia oraria è valida	Obbligatorio	UC1.1
RF1.13	Il sistema deve accedere alla posizione dell'utente	Obbligatorio	UC1.8
RF1.14	Il sistema deve sapere la posizione dei ristoranti	Obbligatorio	UC1.9

Tabella 3.1: Gestione sessione tavolo

Codice	Descrizione	Rilevanza	Fonti
RF2.1	Il sistema deve tener traccia dei vari piatti per ogni ristorante con annesse informazioni: §3.3	Obbligatorio	UC2.x
RF2.2	Il sistema deve tener traccia dei piatti suggeriti dal ristoratore	Obbligatorio	UC2.1
RF2.3	Il sistema deve tener traccia dei piatti popolari	Obbligatorio	UC2.2
RF2.4	Il sistema deve raccogliere gli ordini finalizzati al fine di fare analisi statistiche	Obbligatorio	UC2.2
RF2.5	Il sistema deve tener traccia dei piatti preferiti per coppia utente/ristorante	Obbligatorio	UC2.4
RF2.6	Il sistema deve tener traccia della cronologia degli ordini	Obbligatorio	UC2.6.1
RF2.7	Il sistema deve tener traccia della valutazione dei piatti per coppia utente/ristorante	Obbligatorio	UC2.6, UC2.7
RF2.8	Il sistema deve calcolare e fornire la valutazione media dei piatti	Obbligatorio	UC2.6
RF2.9	Il sistema deve tener traccia dei piatti aggiunti all'ordine da ogni utente nella sessione	Obbligatorio	UC2.1x
RF2.10	Il sistema deve fornire a ogni utente la lista aggregata di tutti gli utenti nella sessione	Obbligatorio	UC2.16
RF2.11	Il sistema deve tener traccia dei piatti già ordinati e in attesa di consegna in un apposita lista	Obbligatorio	UC2.18, UC2.19
RF2.12	Il sistema deve fornire all'utente il numero del tavolo agli utenti	Obbligatorio	UC2.20
RF2.13	Il sistema deve tener traccia di chi ha creato il tavolo per permettere solo a quest'ultimo lo spostamento degli ordini alla lista "in arrivo"	Obbligatorio	UC2.17

Tabella 3.2: Gestione ordini

Codice	Descrizione	Rilevanza	Fonti
RF3.1	Il sistema deve permettere ai clienti di registrarsi	Obbligatorio	UC3.x
RF3.2	Il sistema deve tener traccia delle informazioni relative a ogni cliente: §3.3	Obbligatorio	UCx.x
RF3.3	Il sistema deve permettere di reimpostare la password tramite un token inviato via email al cliente	Obbligatorio	UC3.4
RF3.4	Il sistema deve tener traccia dei punti fedeltà	Obbligatorio	UC3.5
RF3.5	Il sistema deve poter inviare email automatiche ai clienti invitandoli a recensire i piatti presi	Obbligatorio	Committente
RF3.6	Il sistema deve poter creare post sui social degli utenti	Obbligatorio	UC3.7

Tabella 3.3: Login e Registrazione - Cliente

Codice	Descrizione	Rilevanza	Fonti
RF4.1	Il sistema deve permettere ai gestori di registrarsi	Obbligatorio	UC4.x
RF4.2	Il sistema deve tener traccia delle informazioni relative a ogni gestore: §3.3	Obbligatorio	UCx.x
RF4.3	Il sistema deve permettere di reimpostare la password tramite un token inviato via email al gestore	Obbligatorio	UC4.4
RF4.4	Il sistema deve permettere a un gestore di diminuire i punti fedeltà di un suo cliente quando fornisce un bonus	Obbligatorio	UC4.6
RF4.5	Il sistema deve potersi integrare con il sistema informativo del ristorante	Obbligatorio	UC4.7

Tabella 3.4: Login e Registrazione - Gestore

Codice	Descrizione	Rilevanza	Fonti
RF5.1	Il sistema deve tener traccia dei dati relativi a ogni menù: §3.3	Obbligatorio	UC5.x
RF5.2	Il sistema deve permettere la modifica dei dati relativi a ogni menù da parte del gestore: §3.3	Obbligatorio	UC5.x
RF5.3	Il sistema deve permettere la generazione di un URL che punta al menù	Obbligatorio	UC5.5
RF5.4	Il sistema deve permettere la generazione di un URL che punta al menù e tavolo specifico	Obbligatorio	UC5.6.2
RF5.5	Il sistema deve la generazione di <i>QR</i> a partire da un URL	Obbligatorio	UC5.6

Tabella 3.5: Gestione menù

Codice	Descrizione	Rilevanza	Fonti
RF6.1	Il sistema deve tener traccia dei dati relativi a ogni piatto: §3.3	Obbligatorio	UC6.x
RF6.2	Il sistema deve permettere la modifica dei dati relativi a ogni piatto da parte del gestore: §3.3	Obbligatorio	UC6.x

Tabella 3.6: Gestione piatti

Codice	Descrizione	Rilevanza	Fonti
RF7.1	Il sistema deve fornire una guida d'utilizzo che ne descriva il funzionamento per il cliente	Obbligatorio	UC7.1
RF7.2	Il sistema deve fornire una guida d'utilizzo che ne descriva il funzionamento per il gestore	Obbligatorio	UC7.1

Tabella 3.7: Guida d'aiuto

3.6.2 Requisiti qualitativi

Codice	Descrizione	Rilevanza	Fonti
RQ1.1	Deve essere documentata ogni scelta per quanto riguarda UI e UX	Obbligatorio	Committente
RQ1.2	Devono essere presenti test di unità per la parte di <i>backend</i>	Obbligatorio	Scelta interna
RQ1.3	Devono essere presenti test di unità per la parte di <i>frontend</i>	Desiderabile	Scelta interna
RQ1.4	La WebApp deve essere accessibile da tutte le categorie di utenti	Obbligatorio	Scelta interna
RQ1.5	La WebApp deve adattarsi a diversi dispositivi seguendo il paradigma responsive	Obbligatorio	Scelta interna
RQ1.6	La scelta della palette cromatica di default deve basarsi su consolidate pratiche di emotional design (nel settore del food è consigliato l' a-rancione)	Desiderabile	Scelta interna

Tabella 3.8: Requisiti qualitativi

3.6.3 Requisiti di vincolo

Codice	Descrizione	Rilevanza	Fonti
RV1.1	Non devono essere necessarie modifiche al sistema informativo già presente nel ristorante	Obbligatorio	Scelta interna
RV1.2	Utilizzo di Angular per il <i>frontend</i>	Obbligatorio	Committente
RV1.3	Utilizzo di Java Spring per il <i>backend</i>	Obbligatorio	Committente
RV1.4	<i>Backend</i> basato su architettura a microservizi e scalabile	Obbligatorio	Committente
RV1.5	I colori e i loghi devono essere personalizzabili per ogni ristorante	Obbligatorio	Committente
RV1.6	I dati degli ordini devono permanere nel local storage del browser	Obbligatorio	Committente
RV1.7	Deve essere possibile unirsi a un tavolo anche dopo aver già aggiunto alcuni ordini	Obbligatorio	Committente
RV1.8	Il sistema deve permettere la valutazione di un piatto solo a chi lo ha effettivamente ordinato	Obbligatorio	Committente

Tabella 3.9: Requisiti di vincolo

Capitolo 4

Progettazione e codifica

In questo capitolo viene trattata la progettazione e la successiva realizzazione del prodotto software

4.1 Progettazione interfacce

Per prima cosa è stato necessario realizzare dei *mock up* delle interfacce grafiche tramite Figma, studiando le conseguenze delle varie scelte sulla *UX*.

I *mock up* realizzati si focalizzano sulla ricerca di un flow di navigazione ottimale tralasciando in parte la scelta dei colori che dovrà essere personalizzabile per ristorante.

Differenze rispetto ai casi d'uso In questa fase di progettazione era ancora presente una lista ulteriore simile ai preferiti chiamata "eat-later". Si è poi deciso di eliminare questa funzione in quanto poco utile e fonte di confusione.

4.1.1 Cliente

Flow generale

L'utilizzo della WebApp da parte del cliente segue i seguenti passi:

1. un commensale crea la sessione per il tavolo;
2. gli altri si uniscono;
3. tutti ordinano;
4. un commensale mostra la lista aggregata degli ordini al cameriere;
5. tutti segnano quali piatti gli sono arrivati (opzionale).

È possibile arrivare alla pagina del menù in 2 modi:

- * tramite link, se ad esempio si effettua un ordine dall'ufficio prima di recarsi al ristorante;
- * tramite scansione di un codice *QR*, ad esempio posto sul tavolo del ristorante.

A questo punto un commensale crea la sessione, mentre gli altri si uniscono in uno dei seguenti modi:

- * tramite codice *QR*;
- * tramite codice alfanumerico che identifica la sessione.

Una volta generata la sessione è possibile condividere sia un codice univoco della sessione che un codice *QR* per permettere agli altri di unirsi. Al fine di facilitare questa azione ogni persona all'interno della sessione potrà essa stessa condividere sia il *QR* che il codice.



Figura 4.1: Mock up della parte di gestione della sessione

Note Se viene scansionato un *QR* con fascia oraria del menù sbagliata viene visualizzato un avviso che chiede conferma al cliente prima di procedere.

La scelta di non mettere un bottone che porti direttamente al menù, ma "costringere" l'utente a passare per il menù ad hamburger serve a permettere a quest'ultimo di costruirsi la prima mappa mentale del sito.

Il numero di tavolo serve a ricordare all'utente in quale tavolo era per quando si recherà alla cassa, non ha altro significato all'interno del software.

È sempre possibile accedere al menù di aiuto per spiegare meglio la procedura all'utente.

Menù

Il menù di ogni ristorante è consultabile diviso in 5 sezioni tramite un layout a scheda:

- * tutti i piatti (separati per sezione);
- * consigliati dal ristorante;
- * più popolari tra i clienti (basati sull'analisi dei dati anonimi raccolti dalla WebApp);
- * preferiti;

- * eat-later, una lista che permette agli utenti registrati di segnare piatti che intendono mangiare un'altra volta.



Figura 4.2: Mock up della parte di menù del ristorante

Per ogni piatto si possono visualizzare:

- * numero (con annessa variante in caso);
- * nome;
- * immagine;
- * allergeni;
- * valutazione media degli utenti;
- * limite ordinabile a persona.

Sono presenti inoltre bottoni che permettono di:

- * aumentare la quantità ordinata;
- * ridurre la quantità ordinata;
- * aggiungere il piatto ai preferiti;
- * aggiungere il piatto ad eat-later;
- * valutare il piatto.

Note Le cinque sezioni sono indicate tramite icone per risparmiare spazio, una volta selezionata una sezione è presente il nome (o meglio ancora una breadcrumb) per chiarire eventuali dubbi sul significato di tali icone.

Le varie categorie nella lista "tutti i piatti" sono individuate solo da un header permettendo di visualizzare l'interno menù solo tramite scrolling ed evitando avanti e

indietro di schede per cambiare categoria.

I bottoni per l'aggiunta ai preferiti e alla lista eat-later se cliccati da un utente non loggato portano alla pagina di login.

Cliccando sulle stelle è possibile aggiungere la propria recensione al piatto.

I tasti di valutazione, aggiunta ai preferiti e eat-later sono raggruppati in quanto costituiscono un'unica classe logica.

I tasti di ordinazione e limite per persona sono anch'essi raggruppati per il medesimo motivo.

Ordini

Durante tutta la sessione è possibile accedere alla sezione ordini per visualizzare:

- * la lista degli ordini di tutto il tavolo;
- * la lista dei propri ordini;
- * la lista dei piatti già ordinati e in attesa di consegna.



Figura 4.3: Mock up della parte di gestione ordini

Note Un commensale, una volta mostrata la lista al cameriere, che l'avrà trascritta nel sistema di gestione delle comande del ristorante, clickerà il pulsante "Sposta la lista a in arrivo".

Quest'azione farà apparire un punto esclamativo sulla sezione dei piatti in arrivo per indirizzare l'utente verso la corretta sezione.

La lista degli ordini aggregati è composta solo da nome, numero e molteplicità per facilitare la lettura, è importante che la molteplicità sia vicina al numero per evitare di dover vagare eccessivamente con lo sguardo quando li si legge in ordine per trascriverli o dettarli al cameriere.

La lista dei propri ordini permette una visualizzazione più dettagliata per cambiare gli ordini stessi.

La lista dei piatti in arrivo permette di confermare la ricezione piatto che verrà rimosso dalla lista. Permette inoltre di valutare un piatto, aggiungerlo ai preferiti o alla lista eat-later.

Login e registrazione

È necessario un account registrato per le seguenti funzionalità:

- * valutare un piatto;
- * aggiungerlo ai preferiti;
- * aggiungerlo alla lista eat-later.

Tutte le altre funzionalità non richiedono autenticazione.

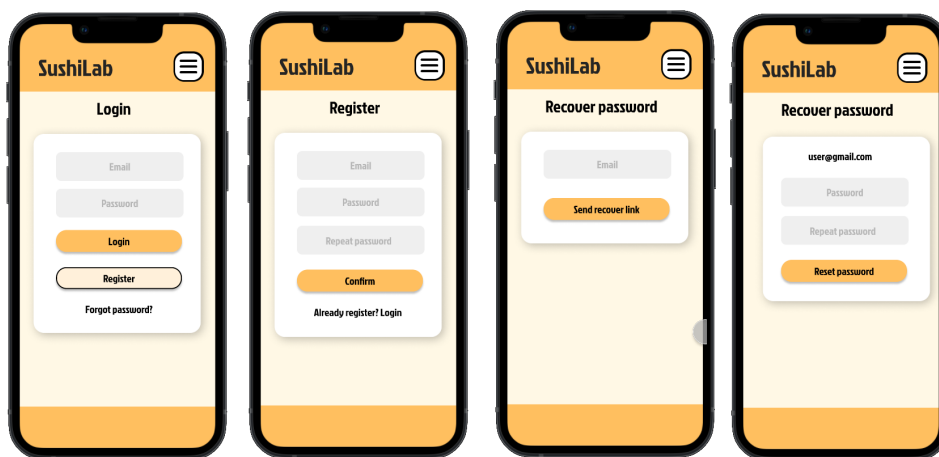


Figura 4.4: Mock up della parte di autenticazione

Note I bottoni sono differenziati con stili diversi avendo significato diverso.

È possibile muoversi agevolmente tra login e registrazione tramite appositi bottoni.

In caso di smarrimento password si può inserire l'indirizzo email al quale verrà inviato un link con un token per reimpostare la password.

4.1.2 Gestore

Flow generale

L'utilizzo della WebApp per il gestore permetterà di:

1. gestire i propri menù, specificando i piatti e le fasce di validità;
2. generare link e *QR* ai propri menù;
3. gestire i propri piatti.

Gestione menù

La gestione dei menù permette di aggiungere menù alla propria lista così come modificare quelli esistenti. Una volta selezionato un menù (anche inedito) è possibile:

- * modificare i piatti presenti;
- * modificare le fasce orarie di validità;
- * ottenere i link e i *QR* che puntano al menù;
- * eliminare il menù.



Figura 4.5: Mock up della parte di gestione menù

Note Il tasto di eliminazione è ben differenziato con stile diverso e chiederà conferma.

È possibile copiare agevolmente il link cliccando su di esso (o tramite pulsante posto al suo fianco).

È possibile scaricare il *QR* senza specificare il numero di tavolo (semplicemente non verrà ricordato all'utente il suddetto numero).

Le fasce orarie di validità sono gestite a lista per garantire la massima flessibilità. È possibile inserire più fasce per il medesimo giorno così come specificare come giorno alcuni preset quali:

- * tutti i giorni;
- * feriali (escluso il sabato);
- * week end.

Gestione piatti

I piatti sono sempre gestiti tramite una lista, una volta selezionati è possibile modificarne i parametri ed eliminarli.



Figura 4.6: Mock up della parte di gestione piatti

Note Il form è diviso in 4 parti per agevolare la compilazione senza sembrare troppo pesante.

Il campo variante permette di dichiarare piatti come ad esempio: "9a" e "9b".

Il testo alternativo favorisce l'accessibilità, tuttavia non è da ritenere obbligatorio in quanto meglio un alt vuoto che uno messo a caso per disinteresse di chi compila il form.

Gli allergeni si comportano come 14 check-box.

La sezione è un menù a tendina con possibilità di ricerca.

Anche qui il pulsante di eliminazione è ben differenziato e chiederà conferma.

4.1.3 Aggiornamento dopo confronto

A seguito con un colloquio in cui i realizzatori dell'altra soluzione ci hanno fornito un feedback basato sulla loro esperienza sono state effettuate alcune modifiche alla progettazione lato cliente.

- * sono state eliminate molte delle liste, tra cui eat-later, consigliati e popolari;
- * la dimensione delle immagini è stata ampiamente aumentata;

- * sono stati aggiunti gli ingredienti, il prezzo alla carta e l'ultima volta che il piatto è stato ordinato;
- * è stata aggiunta la possibilità di inserire note sugli ordini;
- * è adesso possibile generare un *QR* che il cameriere potrà scansionare per prendere automaticamente l'ordine;
- * è stata aggiunta la possibilità di filtrare i piatti in base ad ingredienti non di nostro gradimento.



Figura 4.7: Mock up modificato dopo il feedback

Note Il menù è stato ristrutturato di conseguenza.

Per rappresentare piatti consigliati verrà fornito uno stile *CSS*^[5] diverso a quei piatti.

Per visualizzare i piatti popolari si inserirà un "fuoco" vicino al nome.

4.1.4 Design della *API REST*

Sulla base dei *mock up* sono state individuate le informazioni necessarie da richiedere al *backend*.

Tramite StopLight si sono documentate le *API REST* che verranno usate per integrare il *backend* con il *frontend*, StopLight inoltre fornisce un *mock* con cui testare e sviluppare il *frontend* in attesa che il *backend* venga realizzato.

4.2 Formazione sulle tecnologie

A seguito della fine della progettazione di massima si è passato allo studio delle tecnologie che si sarebbero utilizzate per il progetto.

Javascript

Il linguaggio è relativamente semplicemente e di facile approccio e non ha presentato grosse difficoltà.

Typescript

Super set di Javascript, la documentazione è molto ben strutturata e fruibile anche con poche basi di Javascript.

Angular

Un framework molto complesso e, proprio a causa della sua vastità di concetti molto interconnessi, la documentazione appare come una lunga serie di precondizioni di conoscenza circolari. Tuttavia quasi la totalità di quello che potrebbe servire durante lo sviluppo viene coperto con esempi appropriati e ben documentati.

Lo studio di questo framework ha impiegato la maggior parte del tempo di formazione durante lo stage.

4.3 Struttura software

L'applicativo è stato realizzato in Angular seguendo le linee guida della documentazione ufficiale.

4.3.1 Moduli

Per prima cosa si sono individuati i vari moduli in cui suddividere l'applicativo, questo non solo permette una maggiore manutenibilità del prodotto, ma tramite la funzionalità di Lazy Loading dei moduli di Angular è possibile caricare i suddetti moduli solo quando richiesti diminuendo il tempo di prima risposta del sito.

I moduli individuati sono i seguenti:

Shared: fornisce componenti condivisi da più moduli, ad esempio la maschera di un piatto.

Nav: gestisce la barra di navigazione, l'header e il footer.

Menu: gestisce i menù.

Preferiti: gestisce la lista dei preferiti.

Ordini: gestisce gli ordini effettuati.

Tavolo: gestisce la sessione del tavolo.

Guida: gestisce la visualizzazione della guida d'aiuto.

Personale: gestisce l'area personale.

Gestore: gestisce l'area personale del gestore del ristorante.

REST: gestisce la comunicazione con il *backend*.

4.3.2 Servizi

I servizi sono classi iniettabili che forniscono funzionalità ai vari componenti. Questi sono inseriti nel costruttore dal Dependency Injector di Angular che si occupa di mantenere una singola istanza di questi ultimi senza incorrere nelle problematiche legate alla difficoltà di realizzare dei *mock* tipiche del pattern Singleton. La possibilità di iniettarli è dichiarata tramite decorator, curiosamente questa funzionalità di Typescript, per quanto molto usata dal framework Angular, viene indicata come sperimentale e soggetta a possibili cambiamenti futuri nella documentazione di Typescript.

I servizi realizzati sono i seguenti:

Auth: gestisce il token di autenticazione fornito dal server e la conseguente logica di login e logout.

Menu: gestisce la comunicazione con il *backend* per la gestione dei menù.

Ordini: gestisce la comunicazione con il *backend* per la gestione degli ordini.

Tavolo: gestisce la comunicazione con il *backend* per la gestione delle sessioni di tavolo.

Theme: gestisce la comunicazione con il *backend* per ottenere il tema del menù e fornirlo alla WebApp.

User: gestisce la comunicazione con il *backend* per la gestione degli utenti.

Gestore: gestisce la comunicazione con il *backend* per la gestione dei menù e dei piatti da parte del gestore di un ristorante.

4.3.3 Interceptor

Un interceptor *HTTP*^[g] permette di modificare tutte le richieste *HTTP* che la WebApp esegue, è stato dunque usato per inserire automaticamente il token di autenticazione (gestito dal servizio di autenticazione) nel header dei pacchetti *HTTP*.

4.3.4 Guardie

Per bloccare l'accesso a determinate path è possibile realizzare delle guardie che tramite un'appropriata logica interna decidano se permettere o meno l'accesso a quest'ultime.

Le guardie usate sono le seguenti:

LogIn: controlla che l'utente sia loggato.

LogOut: controlla che l'utente sia non loggato, ad esempio blocca l'accesso alla pagina di registrazione agli utenti già loggati.

Menu: controlla che l'utente abbia selezionato un menù.

Tavolo: controlla che l'utente sia all'interno di una sessione di tavolo.

Gestore: controlla che l'utente sia un gestore di un ristorante.

4.3.5 Componenti

Inoltre seguendo lo stile di sviluppo tipico di Angular i vari elementi della pagina sono stati individuati in componenti riutilizzabili e incapsulati in pieno stile di programmazione ad oggetti.

Ogni componente è composto da:

File Typescript: individua la logica intera e il comportamento.

Template *HTML*^[§]: descrive la struttura del componente, Angular fornisce costrutti propri più espressivi del *HTML* quali ad esempio cicli, istruzione condizionali e interpolazione di codice Javascript.

Foglio *SCSS*^[§]: fornisce lo stile al componente, il file *SCSS* viene compilato in un file *CSS* per l'utilizzo da parte del browser.

File di specifica: permette i test di unità in Typescript.

4.4 Design Pattern utilizzati

Per la realizzazione del progetto sono stati usati molti design pattern integrati con Angular.

Singleton

Questo pattern permette di avere un'unica istanza di un oggetto, utile ad esempio per simulare la connessione a un database che è unica e condivisa per l'intero programma.

Angular fornisce automaticamente un singleton di ogni oggetto iniettabile tramite il suo dependency injector.

Dependency Injector

Questo pattern permette a un oggetto di inserire autonomamente le dipendenze richieste per la costruzione delle varie classi.

I vantaggi sono sia la possibilità di fornire dei singleton condivisi alle varie classi, ma anche di permettere facilmente la sostituzione di questi oggetti con versioni più aggiornate oppure dei *mock* in fase di test.

Tutti gli oggetti da iniettare sono dichiarati nei costruttori delle varie classi il che permette un facile tracciamento delle dipendenze.

Decorator

Questo pattern permette di aggiungere dinamicamente funzionalità a un oggetto o una classe.

Angular ne fa uso principalmente per aggiungere meta-data alle classi per poi gestirle di conseguenza.

Service

Questo pattern modella la necessità di avere della logica condivisa dall'applicativo alla quale si può accedere da più viste.

I service vengono iniettati nelle classi dei componenti ove necessario.

Lazy Loading

Questo pattern permette di ritardare l'invio di alcuni moduli al client fino a quando non sono richiesti fornendo tempi di caricamento iniziali minori.

Di default Angular carica tutti i moduli necessari all'avvio e li invia tutti al browser dell'utente.

Observer

Questo pattern permette di restare in ascolto di eventi generati da un'altra classe e si presta molto bene a una programmazione asincrona basata su messaggi.

In Angular le richieste *HTTP* sfruttano questo pattern permettendo di dichiarare che operazioni eseguire alla ricezione della risposta.

Builder

Questo pattern facilita la costruzione di oggetti complessi e articolati.

Viene usato in Angular per facilitare la generazione dinamica di form di input complesse.

4.5 Codifica

4.5.1 Componente di navigazione

Innanzitutto è stato realizzato il menù di navigazione che permette di navigare le varie parti del sito.

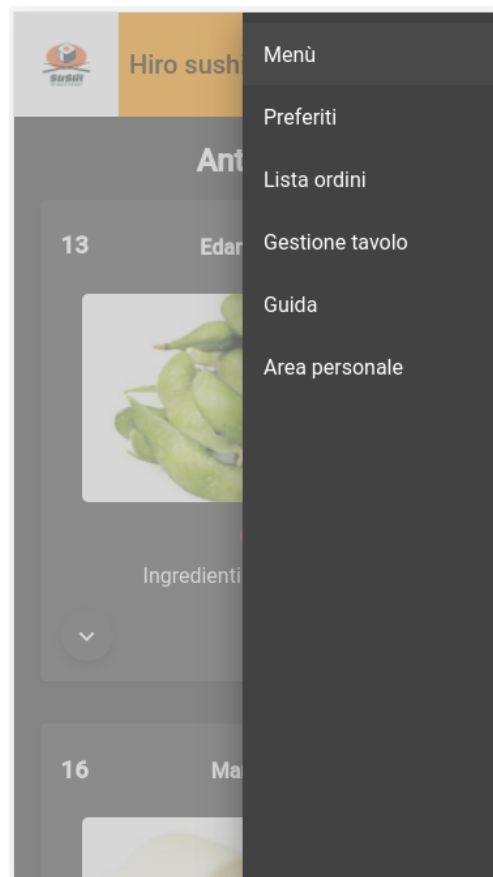


Figura 4.8: Menù di navigazione

Ogni parte viene fornita con un modulo a sè caricato solo all'occorrenza. La suddivisione in moduli ha permesso di meglio suddividere il lavoro con l'altro stagista.

4.5.2 Maschera di login

La prima sezione da me realizzata è stata la maschera di login. Questo ha permesso di prendere familiarità con i form di Angular material e la validazione dei loro campi.

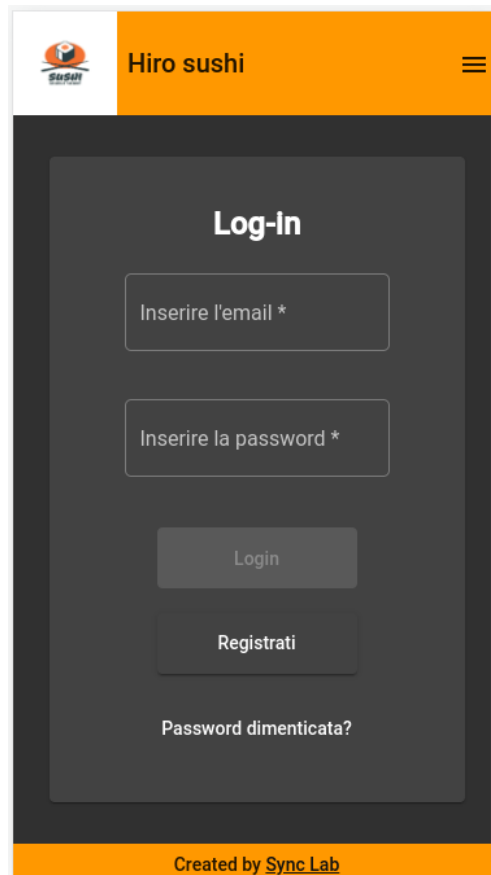


Figura 4.9: Maschera di login

La maschera di login si assicura che l'email sia in un formato valido e che la password sia presente. Non vengono fatti controlli sulla lunghezza della password nella fase di login in quanto è considerata buona norma non validare le password per non fornire informazioni a un possibile attaccante.

Se i dati sono validi il bottone di login viene abilitato e invierà al *backend* le credenziali di login, se il login andrà buon fine verrà fornito un token di autenticazione da utilizzare per le future comunicazioni.

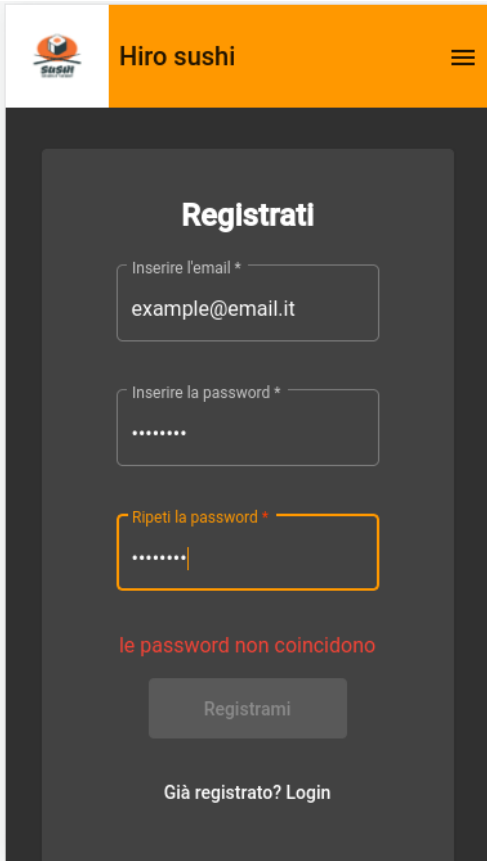
Il token ricevuto viene poi automaticamente iniettato nell'header di ogni pacchetto tramite un interceptor.

Servizi utilizzati

- * `AuthService`: tramite il quale viene gestito il token e la comunicazione con il *backend*.

4.5.3 Maschera di registrazione

Sono poi passato alla maschera di registrazione.



The screenshot shows a mobile application interface for 'Hiro sushi'. At the top, there is a logo on the left and the text 'Hiro sushi' in the center, with a hamburger menu icon on the right. The main content area is dark grey and titled 'Registrali'. It contains three input fields: 'Inserire l'email *' with the value 'example@email.it', 'Inserire la password *' with masked characters, and 'Ripeti la password *' with masked characters. Below the password fields, a red error message reads 'le password non coincidono'. At the bottom, there is a 'Registrami' button and a link 'Già registrato? Login'.

Figura 4.10: Maschera di registrazione

La maschera di registrazione non è molto diversa dalla maschera di login dal punto di vista tecnico.

L'unica differenza significativa è la necessità di realizzare una funzione di tipo *ValidatorFn* che fornisca la logica di cross-validation, ovvero controlli che la password inserita sia effettivamente uguale in entrambi i campi in cui viene richiesta.

Servizi utilizzati

- * `userService`: tramite il quale viene gestita la comunicazione per registrare un utente sul *backend*.

4.5.4 Maschera di inserimento ordini

La realizzazione della maschera di inserimento ordini, ovvero il menù è stata la più complessa ed articolata.

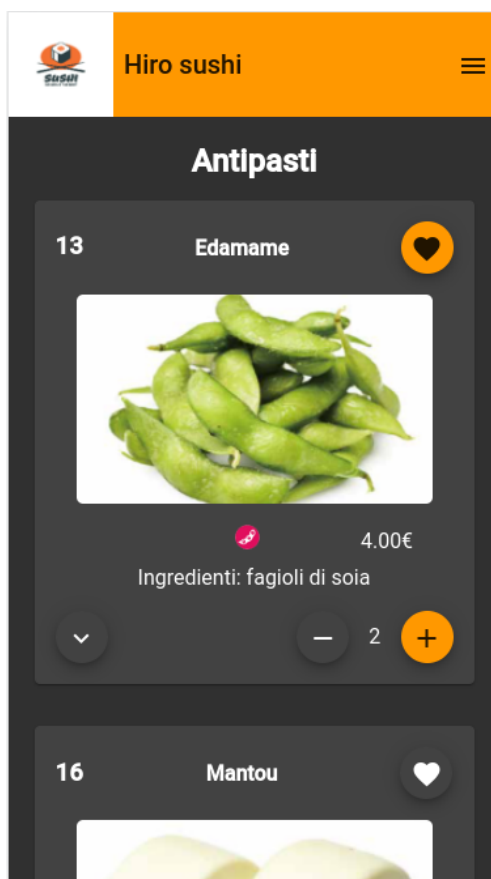


Figura 4.11: Maschera di inserimento ordini

Componente piatto

In primo luogo è stato necessario realizzare un component per il piatto. Questo component deve:

- * mostrare tutti i dati presenti in un piatto;
- * permettere di modificare la quantità ordinata;
- * permettere di modificare la valutazione;
- * permettere di aggiungere o rimuovere il piatto dai preferiti;
- * permettere l'aggiunta di note per il cameriere.

Tipizzazione Data la complessità dell'entità piatto è stato molto utile creare una classe che ne descrivesse i campi per permettere alla gestione dei tipi di Typescript di controllare eventuali errori o sviste.

```
export interface Piatto {
  id: number,
  numero: number,
  variante: string,
  nome: string
  prezzo: number,
  allergeni: string[],
  ingredienti: string[],
  limite: number,
  immagine: string,
  alt: string,
  valutazioneMedia: number,
  valutazioneUtente: number | null,
  preferito: boolean | null,
  ultimoOrdine: string | null,
  popolare: boolean,
  consigliato: boolean
}
```

Figura 4.12: Interfaccia che dichiara le informazioni di cui si compone un piatto

La scelta di separare gli allergeni dagli ingredienti può sembrare a prima vista strana, tuttavia c'è da considerare che la presenza di allergeni può essere dovuta a una contaminazione incrociata durante la preparazione e non va confusa con la lista ingredienti che invece ha lo scopo di descrivere il sapore del piatto.

Menù espandibile Per semplificare la visualizzazione si è optato per un menù espandibile per alcune informazioni. Inoltre la gestione della valutazione è stata delegata ad un altro componente seguendo il principio di *divide et impera*.

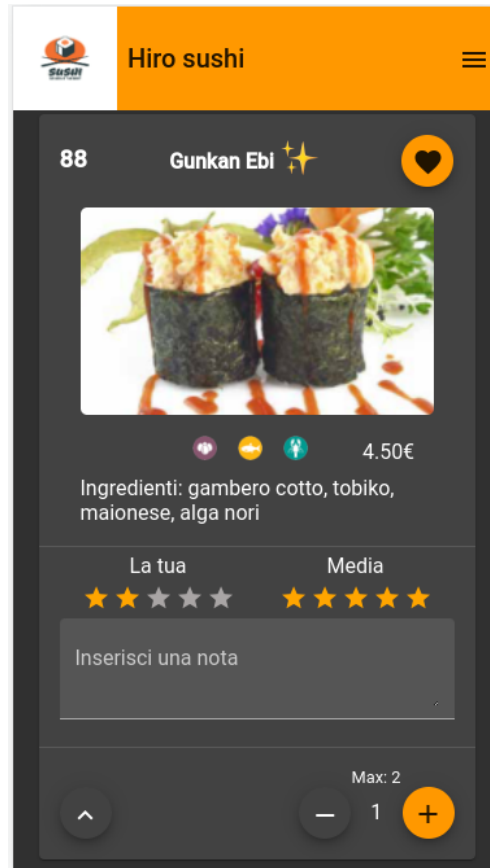


Figura 4.13: Menù espandibile del piatto

Gli ordini inseriti vengono inviati ogni 5 secondi al *backend* se sono stati modificati rispetto all'ultimo invio.

Componente menù

Una volta creato il componente del piatto è bastato inserire le varie sezioni del menù una dopo l'altra disponendo i piatti come card tramite il layout *CSS* di tipo flex.

Questo permette di visualizzare il menù su ogni tipo di dispositivo, per quanto il focus del progetto sia sulla versione mobile non è stata trascurata la componente desktop.

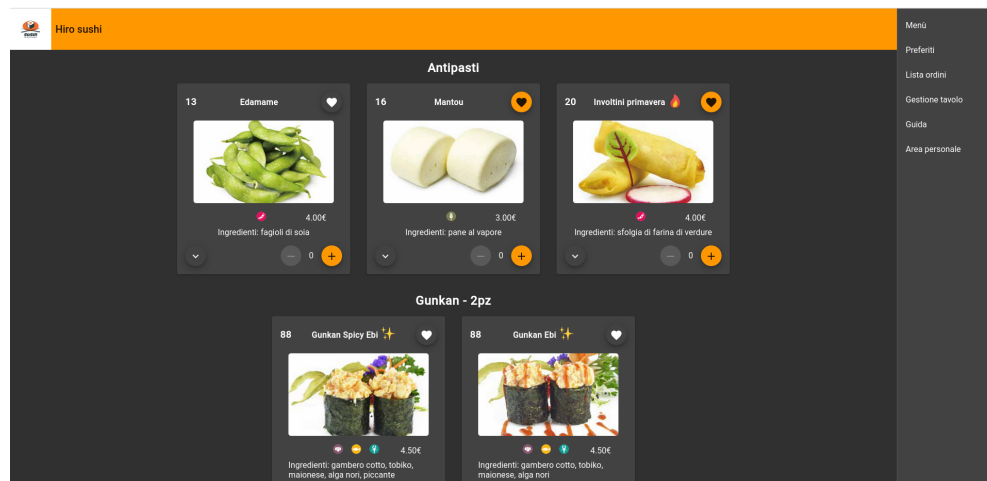


Figura 4.14: Visualizzazione desktop

Servizi utilizzati

- * menuService: tramite il quale viene ottenuto il menù dal *backend*;
- * ordiniService: tramite il quale vengono prima salvati gli ordini nel local storage o poi inviati al *backend*;
- * authService: tramite il quale viene verificato lo stato di login per le funzionalità di valutazione e gestione della lista preferiti.

4.5.5 Gestione dei temi

Data la richiesta di poter personalizzare lo stile grafico per ogni ristorante si è scelto di legare ad ogni menù un tema.

Questa personalizzazione permette non solo a un ristorante di usare i propri colori di bandiera, ma anche, ad esempio, fornire un tema chiaro per il menù pranzo e uno scuro per il menù cena.



Figura 4.15: Menù con tema alternativo

L'impostazione del tema viene applicata a tutte le pagine della WebApp usando il tema dell'ultimo menù visualizzato. Questo permette di mantenere una maggiore coerenza grafica all'interno dell'interfaccia utente.

Servizi utilizzati

- * menuService: tramite il quale vengono ottenute le informazioni relative al tema;
- * themeService: tramite il quale viene impostato il tema all'intera WebApp.

4.5.6 Blacklist ingredienti

Considerata la vastità del menù si è deciso di fornire la possibilità di impostare una blacklist degli ingredienti e allergeni da non visualizzare nel menù stesso.

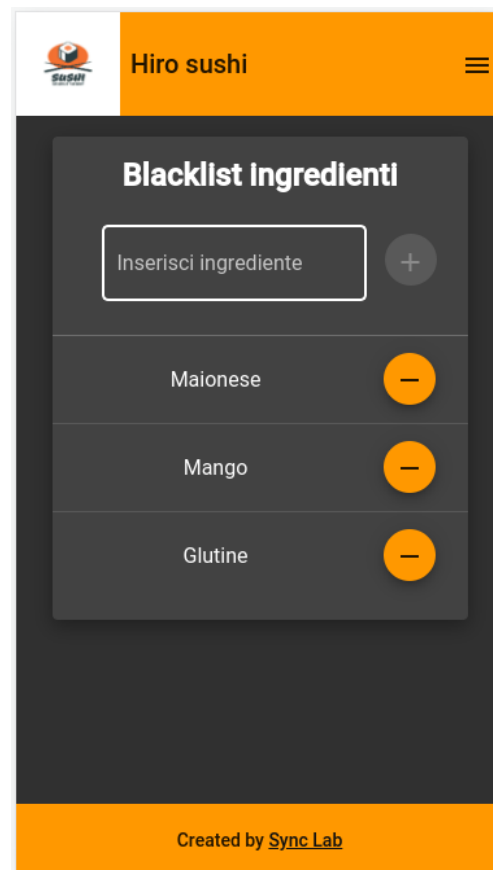


Figura 4.16: Maschera di configurazione blacklist

Il filtro è case in-sensitive e l'interfaccia previene l'inserimento di elementi duplicati nella blacklist.

Servizi utilizzati

- * userService: tramite il quale viene ottenuta e aggiornata la preferenza utente anche lato *backend*.

4.5.7 Gestione piatti

Lo sviluppo delle maschere sopracitate non ha comportato grosse criticità e per questo è stato possibile realizzare ulteriori maschere non previste inizialmente nel programma di stage.

La maschera di gestione piatti permette a un ristoratore di aggiungere e modificare in autonomia i propri piatti. Il design di questa maschera, essendo pensata per l'utilizzo da parte del gestore, usa un paradigma *desktop first* privilegiando l'utilizzo da computer rispetto a quello da cellulare.

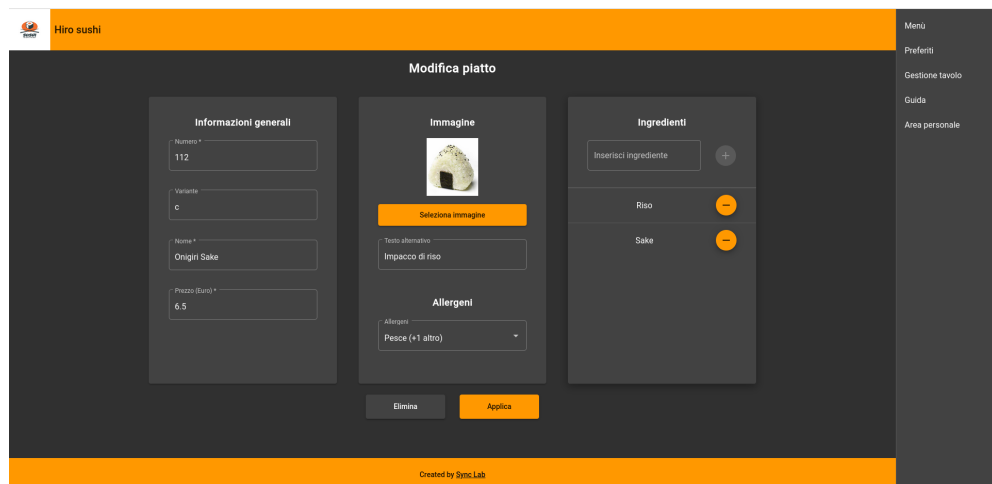


Figura 4.17: Maschera di modifica piatto

La selezione degli allergeni apre un menù a tendina con le varie checkbox che indicano i possibili allergeni, la lista ingredienti permette invece di inserire qualunque ingrediente senza limitarsi a un'elenco predeterminato.

Servizi utilizzati

- * `gestoreService`: tramite il quale avviene la trasmissione dei dati del piatto da e al *backend*.

4.5.8 Gestione menù

Una volta inserito un piatto è possibile aggiungerlo a un qualunque menù. Questo inserimento in due fasi, prima il piatto e poi l'associazione del piatto al menù facilita il riutilizzo dello stesso piatto per due menù, come ad esempio il menù pranzo e quello cena.

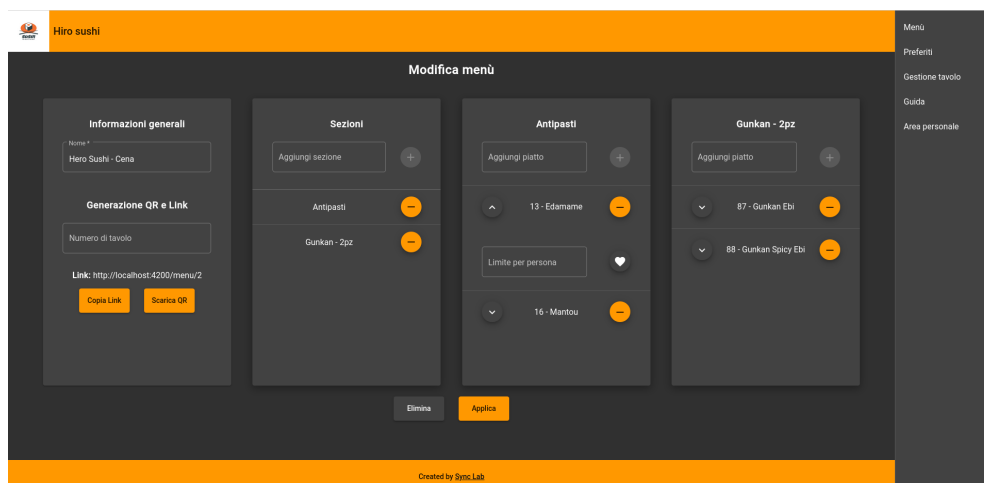


Figura 4.18: Maschera di modifica menù

Questa maschera permette di modificare, oltre al nome del menù, le sezioni di cui è composto e per ogni sezione quali piatti la compongono.

Per ogni sezione inserita è presente una *card* della suddetta sezione in cui inserire i piatti. Il campo di inserimento piatto fornisce un autocompletamento per facilitare l'inserimento avendo cura di non suggerire piatti già presenti nella sezione. Per ogni piatto è possibile specificare il limite per persona e se esso è consigliato dal ristorante.

Le funzioni di eliminazione sezione e dell'intero menù chiedono conferma tramite un pop up data la pericolosità della suddetta azione.

Tramite questa maschera il ristorante può sia generare i link sia scaricare i codici *QR* tramite i quali i clienti accederanno al menù.

Servizi utilizzati

- * *gestoreService*: tramite il quale avviene la trasmissione dei dati del menù da e al *backend*.

Capitolo 5

HotJar

In questo capitolo viene descritta la fase di integrazione con la libreria HotJar per il tracciamento degli utenti.

5.1 Integrazione con Angular

Per integrare la libreria HotJar in un progetto Angular è sufficiente inserire lo script di tracciamento fornito da HotJar all'interno dell'head di ogni pagina della WebApp in cui si vogliono raccogliere i dati di navigazione.

Nel caso del progetto di stage, volendo tracciare gli utenti in ogni pagina, è stato inserito all'interno del file *src/index.html*.

Nota Lo scopo era valutare la fattibilità di integrare Angular con HotJar e valutare le capacità di quest'ultimo. Una soluzione di questo genere non è ammissibile in produzione dove lo script andrebbe inserito all'interno dell'head solo dopo che l'utente ha accettato l'informativa sulla privacy.

Tuttavia essendo questa operazione tanto triviale quanto prematura allo stato attuale del progetto si è optato per una soluzione più semplice, ma da considerare assolutamente temporanea.

5.2 Limiti tecnici di HotJar

Server locale

Lo script in esecuzione sul browser dell'utente invia a HotJar le azioni effettuate dall'utente, quali ad esempio movimento del mouse, click e scroll.

Tuttavia per avere un riferimento visuale di dove questi eventi avvengano nella pagina è necessario uno *screenshot* del sito, quest'ultimo è effettuato direttamente da HotJar col fine di limitare i dati che il browser deve inviare. Da questo segue la necessità che HotJar sia in grado di accedere al sito che deve quindi essere esposto pubblicamente.

Questa necessità impedisce di usare HotJar in ambienti di testing, i quali di solito sono solamente accessibili dalla stessa macchina o al limite dalla stessa rete locale. Per ovviare al problema è stato installato il progetto su un server dell'azienda che è accessibile da tutta la rete internet.

HTTP

A partire dal 2 Maggio 2022 HotJar ha interrotto l'invio dati da pagine fornite tramite [HTTP](#) richiedendo l'utilizzo del più sicuro protocollo [HTTPS](#)^[8].

Questa scelta, sicuramente ragionevole per evitare che i dati possano essere facilmente intercettati da utenti malevoli, ha reso necessario fornire il sito tramite il protocollo [HTTPS](#).

Il tutor aziendale ha dunque provveduto a installare i certificati richiesti sul server.

Browser

Alcuni browser bloccano automaticamente il codice di tracciamento di HotJar per tutelare la privacy dei loro utenti.

Un esempio è Firefox nel quale l'impostazione anti tracciamento è abilitata di default. Chrome, sia desktop che mobile, invece permette il tracciamento degli utenti.

5.3 Dashboard

HotJar fornisce alcune statistiche sui visitatori del sito, quali il tipo di dispositivo, la nazionalità, il tempo di permanenza medio e il numero di pagine visitate.

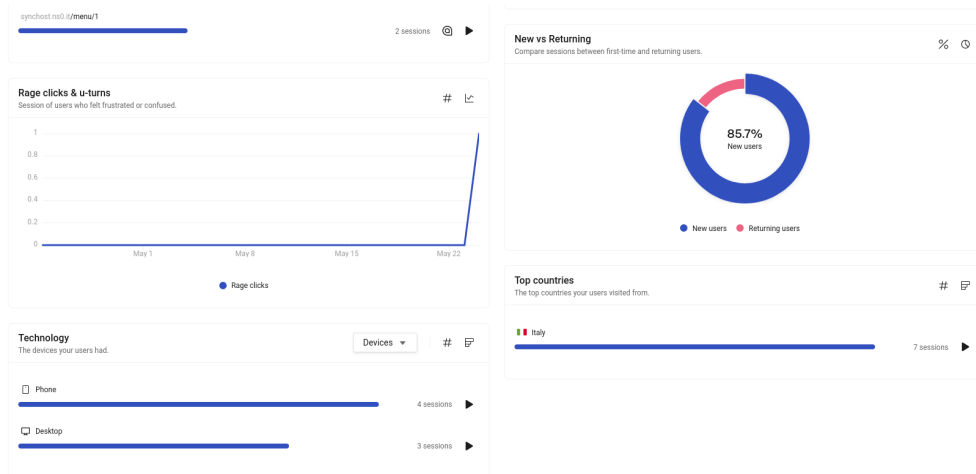


Figura 5.1: Dashboard del sito

Nella dashboard si nota il grafico che mostra i *rage clicks*, ovvero quando un utente preme molte volte in un piccolo intervallo di tempo, in questo caso si è potuto risalire alla recording della sessione dell'utente in questione e constatare come abbia, ai fini di test, inserito rapidamente oltre 70 ordini di un solo piatto.

5.4 Heatmap

Le Heatmap rappresentano tramite il colore i *punti caldi* del sito, permettendo di visualizzare vari tipi di informazioni sul comportamento degli utenti.

Essendo il progetto non ancora operativo e dunque non visualizzato da utenti reali, i dati che seguono rappresentano l'interazione degli sviluppatori con il sito.

Click map

Le click map rappresentano i punti i cui gli utenti sono più soliti cliccare, questo permette ad esempio di:

- * individuare pulsanti che non vengono premuti quanto vorremmo, e dunque renderli più visibili;
- * individuare click errati, ad esempio su elementi che vengono confusi per pulsanti, e dunque cambiare il loro stile di conseguenza.

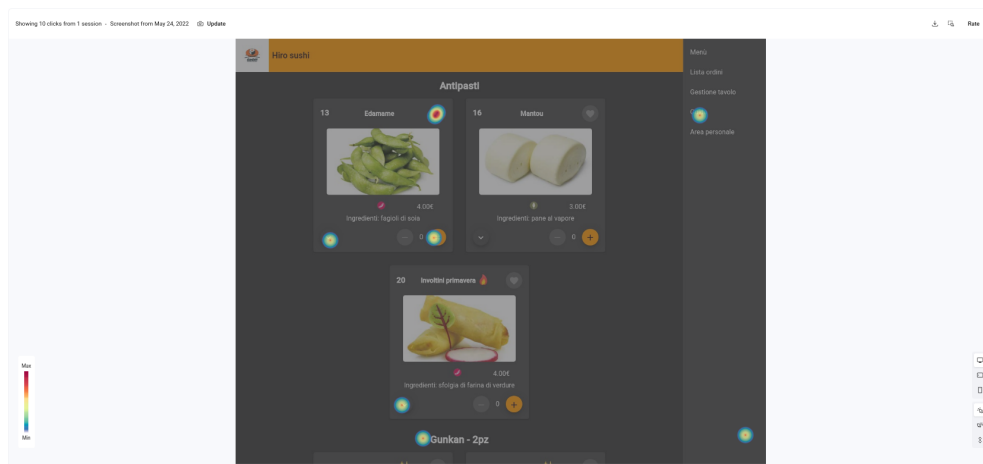


Figura 5.2: Click map della pagina del menù

In questa heatmap si può vedere come i punti evidenziati corrispondano ai bottoni come ci si aspetterebbe.

Move map

Le move map rappresentano la posizione del mouse e i suoi movimenti, dato che statisticamente gli utenti tendono a focalizzare l'attenzione dove è il mouse, è possibile stimare quali aree siano più soggette all'attenzione degli utenti. Non sono tuttavia utilizzabili per tracciare i dispositivi privi di mouse quali i cellulari.

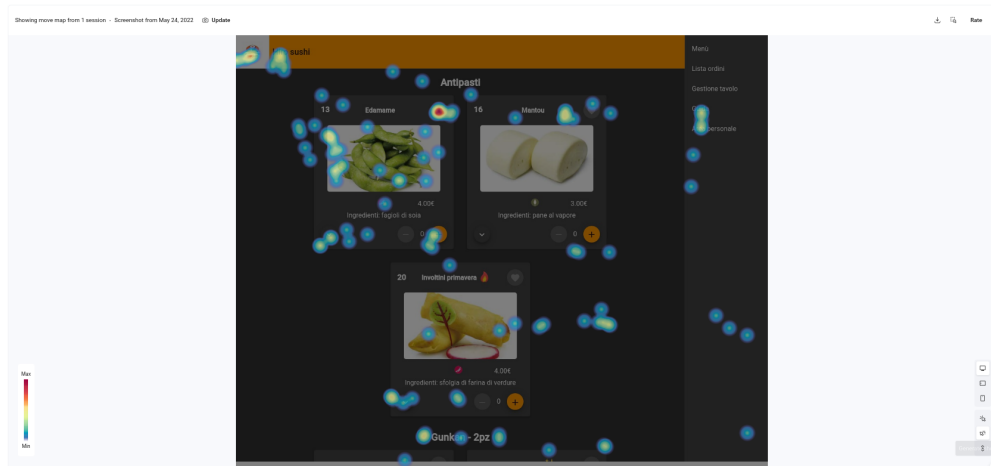


Figura 5.3: Move map della pagina del menù

Da questa heatmap si può vedere come l'attenzione dell'utente spazi su varie parti del sito, con maggior focus sui piatti e sulla barra di navigazione.

Scroll map

Le scroll map visualizzano quali parti del sito vengono viste dagli utenti tramite lo scroll, questo permette ad esempio di:

- * capire se gli utenti non effettuano lo scroll e quindi non sono interessati al contenuto della pagina.
- * individuare fino a che punto della pagina gli utenti guardano e mettere i contenuti più importanti sopra di esso.

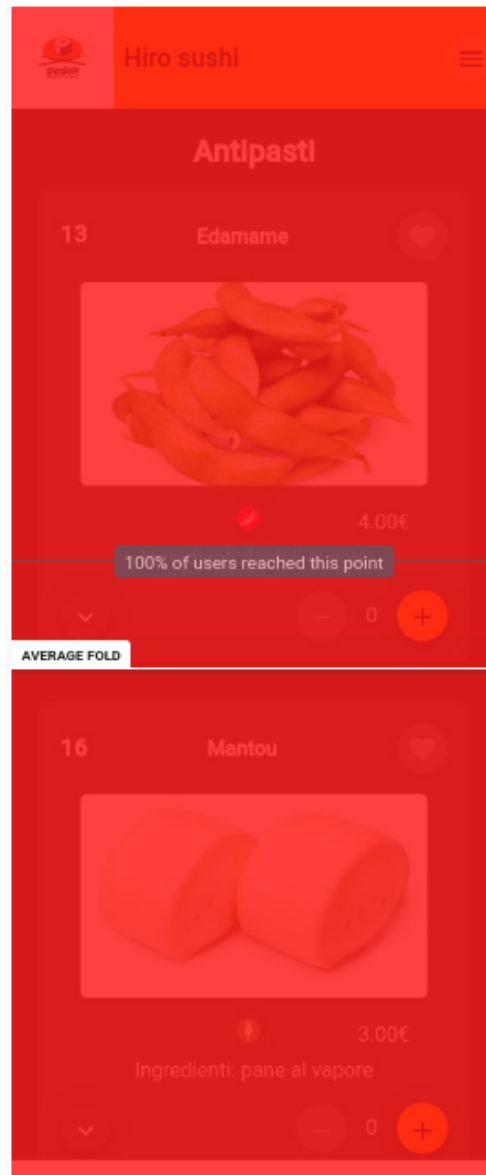


Figura 5.4: Scroll map della pagina del menù

In questa scroll map si può vedere come ogni utente abbia visualizzato l'intera pagina, cosa dovuta principalmente al fatto che gli utenti erano gli sviluppatori del sito.

5.5 Recording

Altra funzionalità offerta da HotJar è la registrazione del comportamento di un singolo utente che potrà poi essere vista successivamente.

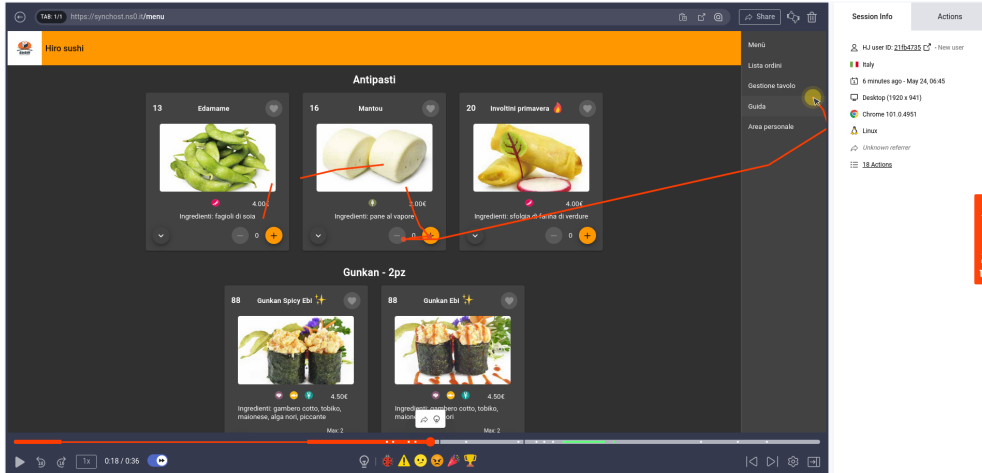


Figura 5.5: Recording di un utente desktop

Questo permette di visualizzare accuratamente il comportamento di un singolo utente e capire quali errori commette nella navigazione e dunque aggiornare il sito per rendere quelle parti più intuitive.

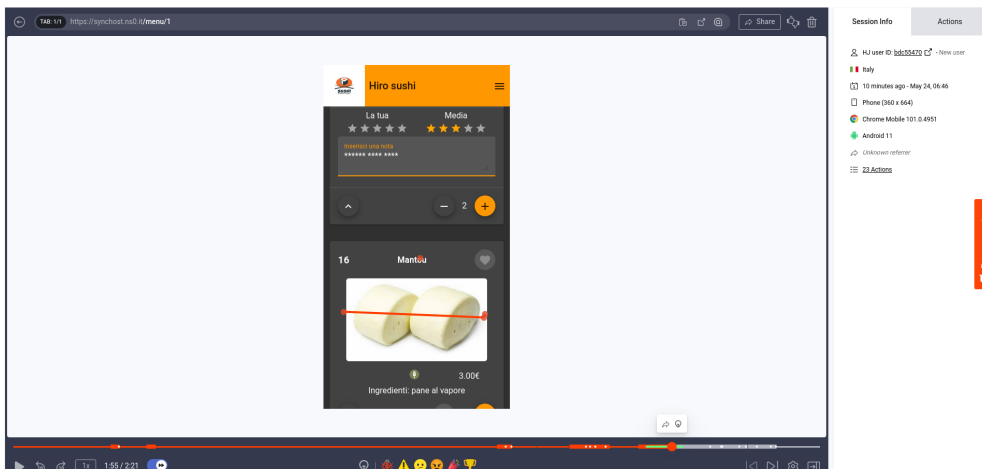


Figura 5.6: Recording di un utente mobile

In questo recording è possibile notare come HotJar sostituisca il contenuto delle note con degli asterischi per tutelare la privacy degli utenti.

Capitolo 6

Verifica e validazione

In questo capitolo viene descritta la fase di verifica e validazione del prodotto realizzato durante lo stage.

6.1 Verifica accessibilità

Per prima cosa è stato verificato il rispetto delle buone prassi in tema di accessibilità.

Si è controllato il corretto uso dei tag [ARIA](#)^[g], in particolare l'uso di *aria-label* per indicare le funzionalità dei bottoni. Inoltre si è stato controllato il rispetto dei attributi *lang* per indicare la lingua del testo.

Questi controlli sono stati effettuati dapprima tramite analisi statica del codice e in seguito ricontrollati usando la funzionalità di ispezione del browser.

Orca

Si è fatto un test di navigazione usando lo screen reader Orca, questo strumento fornisce il contenuto della pagina tramite un sintetizzatore vocale ed è usato da parte di utenti con disabilità visiva per accedere ai contenuti digitali.

Orca è uno screen reader open source che attualmente non può rivaleggiare con altre soluzioni a pagamento, tuttavia permette di avere un'idea dell'iterazione col sito da parte di queste categorie di utenti.

Anche se la mia abilità con questo tipo di strumenti è molto limitata rispetto ad un utente che ne fa uso costantemente e la mia interazione con esso molto confusa e macchinosa risulta sempre buona pratica effettuare dei test anche con questi strumenti.

Firefox

Si sono poi effettuati alcuni test tramite gli strumenti forniti da Firefox.

Per prima cosa è stato controllato l'ordine di tabulazione, per assicurarsi che la navigazione tramite tastiera risulti agevole e intuitiva.

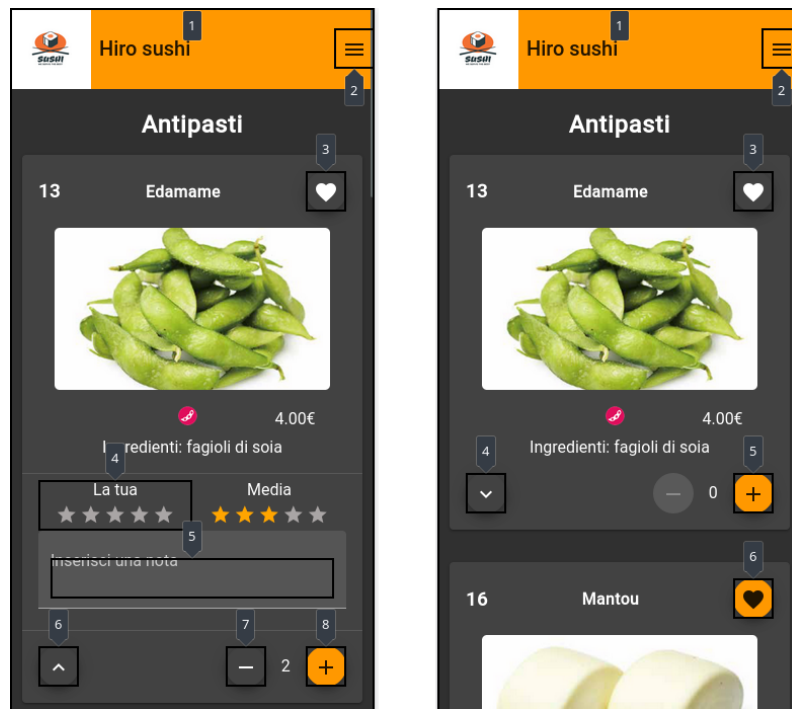


Figura 6.1: Ordine di tabulazione della maschera piatto

In secondo luogo si è verificato che nessuna informazione sia veicolata unicamente dal colore e che quindi risulti percepibile anche da utenti affetti da daltonismo.

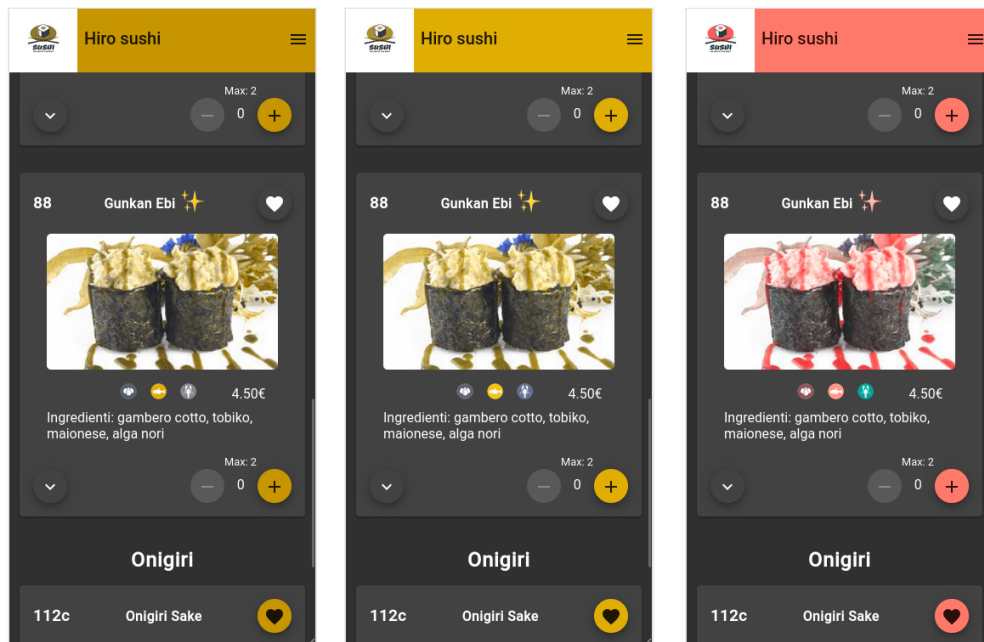


Figura 6.2: Simulazione dei vari tipi di daltonismo

Lighthouse

Lighthouse è uno strumento integrato in Google Chrome che effettua una serie di test automatici su pagine web in merito ad accessibilità, performance e buone pratiche per permettere una miglior indicizzazione da parte dei motori di ricerca.

I risultati di questi test sono stati assolutamente positivi per quanto riguarda l'accessibilità, tuttavia hanno evidenziato forti criticità per quanto riguarda le performance.

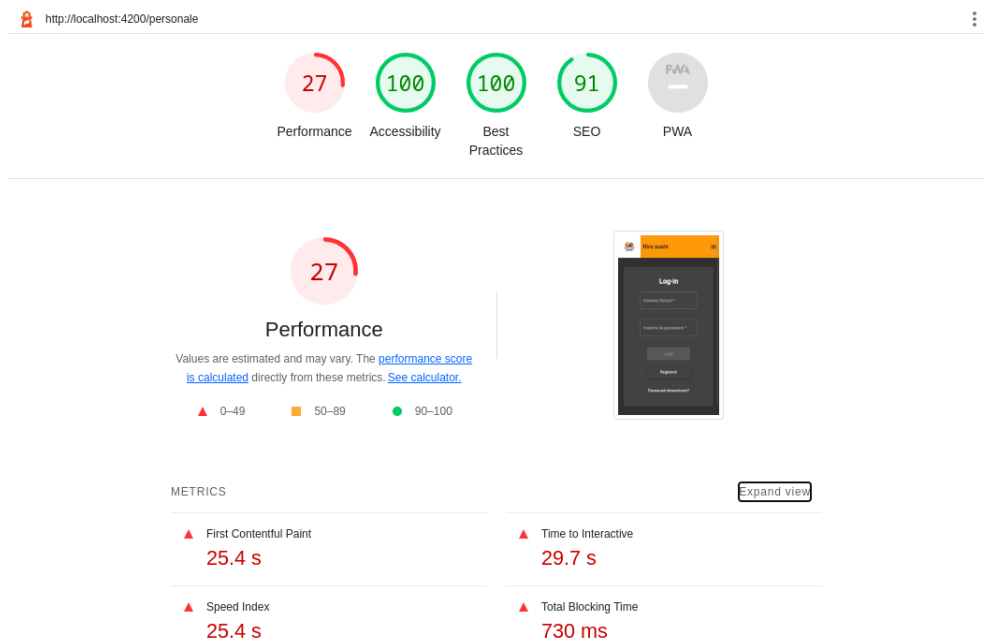


Figura 6.3: Risultati dei test di Lighthouse

Il motivo principale è da imputare al pesante utilizzo di Javascript da parte di Angular. La pagina viene generata dal browser dell'utente eseguendo un quantitativo non indifferente di codice Javascript, il che ne rallenta di molto il tempo di caricamento iniziale.

Angular fornisce delle misure di mitigazione, quali rendering delle pagine lato server per migliorare queste metriche, tuttavia allo stato attuale del progetto non sono ancora state implementate.

Criticità riscontrate

Durante la fase di verifica sono stati individuati alcuni problemi di accessibilità che sono stati prontamente risolti.

Tasto di caricamento immagine

La funzionalità di caricamento dell'immagine di un piatto, che era basata su codice altrui, forniva una visualizzazione più accattivante nascondendo parte del widget di input per mostrare solo un bottone.

Questa soluzione non permetteva l'interazione tramite tastiera, il che è stato risolto modificando l'implementazione senza però intaccare lo stile visuale della pagina.

Widget di valutazione

Il widget che permette di valutare un piatto, sempre basato su codice altrui, mancava sia di una descrizione testuale adeguata che del supporto per l'utilizzo da tastiera. La descrizione testuale è stata di facile aggiunta, per quanto riguarda il supporto da tastiera è sorto il problema di come realizzarlo.

Sarebbe stato possibile rendere ognuna delle cinque stelle selezionabili col focus e attivabili con il tasto spazio o invio, tuttavia dover navigare con il tab ogni stella in una pagina che potenzialmente supera i cento piatti avrebbe creato un certo disagio.

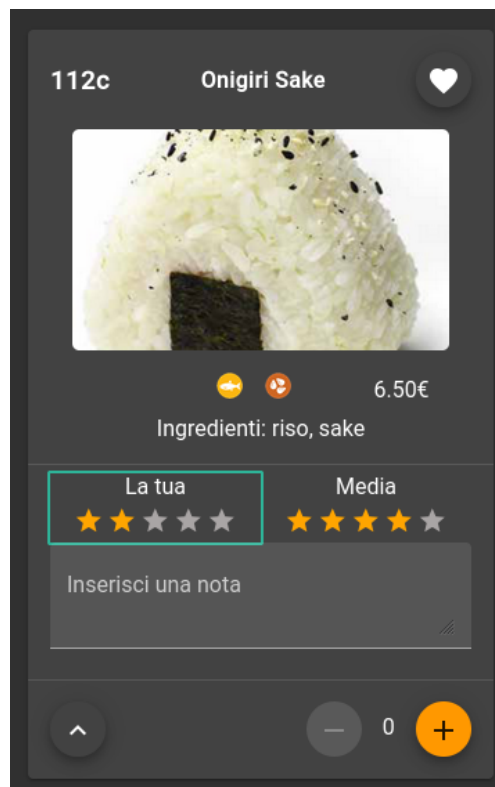


Figura 6.4: Widget di valutazione selezionato tramite tab

La soluzione adottata invece permette di selezionare con il tasto tab il widget nel suo insieme e poi fornire la valutazione tramite i tasti numerici, da 0 a 5, oppure di incrementare e diminuire la suddetta valutazione tramite i tasti più e meno oppure le frecce direzionali.

Se questa soluzione sia sufficientemente intuitiva tuttavia potrà essere accurato solo tramite lo studio dell'utilizzo da parte degli utenti.

6.2 Validazione dei requisiti

Al termine dello stage la copertura dei requisiti, considerando solo la parte *frontend* della soluzione software è quasi totale.

La copertura è stata calcolata comprendendo sia le maschere realizzata da me sia quelle realizzate da Luciano Wu.

Tipologia	Coperti	Totale	Percentuale
Funzionali	37	47	72%
Qualitativi	5	6	83%
Di vincolo	7	8	88%
Totale	49	61	80%

Tabella 6.1: Copertura dei requisiti

I requisiti mancanti riguardano funzionalità secondarie quali l'integrazione coi social e la ricerca dei ristoranti la cui realizzazione risulta ancora prematura.

Capitolo 7

Conclusioni

In questo capitolo vengono tratte le conclusioni sull'esperienza di stage.

7.1 Raggiungimento degli obiettivi

Il raggiungimento degli obiettivi, indicati dalla sezione 2.1, è da ritenere soddisfacente. Lo sviluppo di una WebApp in Angular mi ha permesso di ottenere competenze su quest'ultimo. L'analisi funzionale e l'analisi dei requisiti realizzate sono state ritenute più che soddisfacenti dall'azienda. La fase di progettazione mi ha permesso di cimentarmi con vari problemi legati alla user experience e di prendere familiarità con strumenti quali Figma. Le maschere richieste sono state realizzate in autonomia senza grosse criticità, il che ha permesso di realizzare anche qualche maschera non prevista dalla pianificazione iniziale. Le interfacce *REST* utilizzate sono state documentate con cura tramite StopLight per agevolare la futura realizzazione del *backend*.

Codice	Descrizione	Stato
O-01	Acquisire competenze sul framework Angular	Soddisfatto
O-02	Acquisire competenze sulle tematiche di user experience, customer experience e user interface	Soddisfatto
O-03	Realizzare l'analisi funzionale del problema	Soddisfatto
O-04	Progettare le interfacce tramite Figma	Soddisfatto
O-05	Realizzare le maschere di login, registrazione ed inserimento ordini	Soddisfatto
O-06	Raggiungimento degli obiettivi richiesti in autonomia seguendo il cronoprogramma	Soddisfatto
O-07	Portare a termine le implementazioni previste con una percentuale di superamento pari o superiore al 80%	Soddisfatto
D-01	Portare a termine le implementazioni previste con una percentuale di superamento pari al 100%	Soddisfatto
D-02	Apportare un valore aggiunto al team soprattutto nelle fasi di analisi requisiti e disegno delle interfacce	Soddisfatto
F-01	Descrivere mediante il tool StopLight tutti i servizi <i>REST</i> necessari alla piattaforma	Soddisfatto

Tabella 7.1: Completamento degli obiettivi di stage

7.2 Conoscenze acquisite

Durante l'esperienza di stage ho appreso diverse tecnologie e strumenti che ritengo mi risulteranno utili anche in futuro.

La progettazione di interfacce tramite tool quali Figma è stata sicuramente istruttiva e ha dimostrato le potenzialità dell'utilizzo di *mock up* di riferimento nel lavoro collaborativo.

Tramite StopLight ho appreso l'utilizzo di servizi *REST* e come realizzare dei *mock* di quest'ultimi in fase di sviluppo.

Ho poi maturato una discreta esperienza con il framework Angular che è ad oggi molto in voga in ambito lavorativo e infine ho potuto constatare le potenzialità e i limiti di strumenti di tracciamento utente quali HotJar.

7.3 Valutazione personale

L'esperienza di stage è stata particolarmente istruttiva e mi ha permesso di avere una prima visione del mondo del lavoro.

Rispetto all'ambiente accademico è risultata un'esperienza più pragmatica e meno legata al rispetto di regole e metodi formali. Ci tengo tuttavia a sottolineare che le conoscenze da me maturate durante il percorso di Laurea si sono rivelate indispensabili ad affrontare le sfide che mi sono state proposte durante lo sviluppo e la progettazione richiesta dall'azienda.

Purtroppo non mi è stato possibile vedere appieno il risultato finale in quanto l'assenza del *backend* mi ha costretto all'utilizzo di *mock* che non riescono a dare una visione complessiva al pari di un vero *backend*. La realizzazione di quest'ultimo, che dovrebbe iniziare nel mese di Luglio 2022, sarà il prossimo passo nello sviluppo del progetto, a quel punto sarà possibile concentrarsi su estensioni secondarie quali l'integrazione coi social e con i sistemi gestionali dei ristoranti.

Nel complesso mi ritengo soddisfatto del lavoro svolto e dell'opportunità che mi è stata concessa durante il percorso di Laurea di confrontarmi con mondo del lavoro.

Acronimi

API Application Program Interface. xii, 4, 46

ARIA Accessible Rich Internet Applications. 69

CSS Cascading Style Sheets. 46, 49, 57

HTML Hypertext Markup Language. 49

HTTP Hypertext Transfer Protocol. 48, 50, 64

HTTPS Hypertext Transfer Protocol over Secure Socket Layer. 64

IT Information Tecnology. 1

QR Quick Response. 3, 14–16, 21, 27, 29, 30, 34, 36, 39, 40, 43, 44, 46, 61

REST Representational State Transfer. xii, 2, 4, 8, 46, 75, 76

SCSS Syntactically Awesome Style Sheet. 49

UX User Experience. vii, 39

Glossario

accessible rich internet applications ARIA definisce uno standard per rendere più accessibile i contenuti web a tutte le categorie di utenti. [77](#)

all you can eat in un ristorante di sushi con il termine *all you can eat* si indica una modalità in cui ogni cliente è libero di ordinare a piacimento da un menù pagando una cifra fissa (che di solito non include bibite, dolci e coperto). Tuttavia al fine di evitare ordini eccessivi e sprechi ogni piatto avanzato deve essere pagato a parte. [1](#), [12](#)

application program interface fornisce un contratto tramite il quale è possibile, da parte di un programmatore, interagire con un programma altrui. [77](#)

backend in informatica col termine backend ci si riferisce alla parte non visibile dell'applicazione che gestisce la logica di dominio. [2](#), [8](#), [37](#), [46–48](#), [52](#), [53](#), [56](#), [57](#), [59–61](#), [75](#), [76](#)

cascading style sheets linguaggio che permette di definire lo stile delle pagine web. [77](#)

endpoint in informatica col termine endpoint ci si riferisce all'interfaccia esposta da un'altra macchina a cui si accede tramite un canale di comunicazione. [8](#)

frontend in informatica col termine frontend ci si riferisce alla parte visibile dell'applicazione esposta all'utente. [1–4](#), [37](#), [46](#), [73](#)

hypertext markup language linguaggio che descrive la struttura di un documento basata sul significato semantico dei suoi elementi collocati in una struttura ad albero. [77](#)

hypertext transfer protocol protocollo che permette l'invio e la ricezione di ipertesti, molto usato in ambito web. [77](#)

hypertext transfer protocol over secure socket layer protocollo che permette l'invio e la ricezione di ipertesti protetti da adeguata crittografia. [77](#)

information technology termine inglese per informatica. [77](#)

mock oggetto fittizio che permette di simulare il comportamento di un oggetto reale in modo controllato. Usato solitamente per test o in attesa del completamento dell'oggetto reale. [3](#), [4](#), [8](#), [46](#), [48](#), [49](#), [76](#)

mock up modello, una realizzazione a scopo illustrativo o meramente espositivo di un oggetto o un sistema, senza le complete funzioni dell'originale. [4](#), [39](#), [46](#), [76](#)

quick response con quick response *code* si intende un codice a barre bidimensionale scannerizzabile tramite smartphone. [77](#)

representational state transfer in informatica il REST è uno stile architetturale per sistemi distribuiti basato sul protocollo HTTP. [77](#)

syntactically awesome style sheet un superset di CSS, originalmente detto SASS con una sintassi stile Python, ha poi aggiunto il supporto per una sintassi più simile al CSS col nome di SCSS. [77](#)

user experience termine utilizzato per definire la relazione tra una persona e un prodotto, un servizio o un sistema. [77](#)

Bibliografia

Siti web consultati

Documentazione Angular. URL: <https://angular.io/docs>.

Documentazione Angular Material. URL: <https://material.angular.io/guides>.

Documentazione Typescript. URL: <https://www.typescriptlang.org/docs>.

Sito di Sync Lab. URL: <https://www.synclab.it/about.php> (cit. a p. 1).

Stack Overflow. URL: <https://stackoverflow.com/>.