Università degli Studi di Padova
Laurea in Informatica
Corso di Ingegneria del Software
Anno Accademico 2023/2024

Gruppo: SWEet16
Email: `sweet16.unipd@gmail.com`

Specifica Tecnica

Redattori:	Alberto C., Alberto M., Alex S.	
Verificatori:	Alberto M., Alex S., Iulius S.	
Amministratore:	Alberto M.	
Destinatari:	T. Vardanega	R. Cardin
Versione:	1.0.0	

Registro delle modifiche

Versione	Data	Autore	Verificatore	Descrizione
1.0.0	2024/06/16	Alex S.		Approvazione per il rilascio
0.6.0	2024/06/15	Alberto M.	Alex S.	Aggiunta sezioni codice
0.5.0	2024/06/12	Alberto C.	Alberto M.	Stesura sezione pattern backend
0.4.0	2024/06/03	Alberto M.	Alex S.	Stesura sezione pattern frontend
0.3.0	2024/05/08	Alberto M.	Alex S.	Stesura sezione tecnologie
0.2.0	2024/05/07	Alberto M.	Alex S.	Stesura sezione introduzione
0.1.0	2024/05/05	Alberto M.	Iulius S.	Stesura scheletro

Indice

1 Introduzione e scopo del documento

1.1 Scopo del documento

Il presente documento ha lo scopo di descrivere e motivare tutte le scelte architetturali che sono state fatte nella fase di progettazione e codifica del prodotto.
Vengono quindi descritte l'architettura logica e di deployment, i design pattern adottati e le tecnologie impiegate nella realizzazione del prodotto per il progetto *Easy Meal*.

1.2 Scopo del prodotto

Lo scopo dell'applicazione è quello di creare una piattaforma che permetta di gestire e semplificare il processo di *prenotazione*^{G} di tavoli all'interno dei ristoranti.
Sarà inoltre possibile anticipare l'esperienza culinaria visionando prima il menù ed andando ad effettuare la propria *ordinazione*^{G} prima di arrivare al ristorante.
Il prodotto offre inoltre un'esperienza di ordinazione delle *pietanze*^{G} collaborativa e coinvolgente, permettendo di condividerla con amici.

L'idea è una piattaforma *SaaS (Software as a Service)*^{G}, in cui i saranno presenti due tipi di utenti:

- *Cliente*^{G}: Utente registrato all'interno dell'applicazione, può cercare ristoranti, effettuare prenotazioni, ordinazioni e inserire feedback e recensioni;
- *Ristoratore*^{G}: Utente registrato all'interno dell'applicazione, può gestire uno o più ristoranti, controllando le prenotazioni e le ordinazioni dei clienti ed i menù del/i ristorante/i.

La piattaforma dovrà essere disponibile attraverso una *Webapp*^{G} accessibile da qualsiasi dispositivo, esso sia *Desktop*^{G} o *Mobile*^{G}.

1.3 Glossario

Al fine di evitare possibili ambiguità o incomprensioni riguardanti la terminologia usata nel documento, è stato deciso di adottare un glossario in cui vengono riportate le varie definizioni.
In questa maniera in esso verranno posti tutti i termini specifici del dominio d'uso con relativi significati.
La presenza di un termine all'interno del glossario viene indicata applicando una " ^{G} " ad apice della parola.

1.4 Maturità del documento

Il presente documento è redatto con un approccio incrementale al fine di poter trattare nuove o ricorrenti questioni in modo rapido ed efficiente, sulla base di decisioni concordate tra tutti i membri del gruppo.
Non può pertanto essere considerato definitivo nella sua attuale versione.

1.5 Riferimenti

1.5.1 Riferimenti normativi

- Regolamento del progetto didattico:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf`;
- *Capitolato d'appalto*^{G} C3 - Easy Meal:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C3.pdf`.

1.5.2 Riferimenti informativi

- I processi di ciclo di vita del software:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf`;
- Glossario:
  `https://github.com/SWEet16-SWE-Group/docs/blob/main/RTB/Documentazione%20Esterna/Glossario.pdf`;
- ISO/IEC 12207:
  `http://www.colonese.it/SviluppoSw_Standard_ISO12207.html`.

1.5.3 Riferimenti tecnici

Tutti i riferimenti (normativi e informativi) a risorse web soggette a variazione sono stati consultati il 2024/06/16.

- React:
  `https://www.react.dev`;
- Axios:
  `https://www.axios-http.com/`;
- Bootstrap:
  `https://www.getbootstrap.com/`;
- Laravel:
  `https://www.laravel.com/`;
- MySQL:
  `https://www.mysql.com/`;
- NodeJS:
  `https://www.nodejs.org/`;
- Docker:
  `https://www.docker.com/`.

2 Tecnologie

In questa sezione viene fornita una panoramica generale delle tecnologie utilizzate per la realizzazione del prodotto in questione.
Vengono infatti descritte le procedure, gli strumenti e le librerie necessari per lo sviluppo, il test e la distribuzione del prodotto.
In particolare, verranno trattate le tecnologie impiegate per la realizzazione del front-end e del back-end, per la gestione del database e per l'integrazione con i servizi previsti.

2.1 Tecnologie per la codifica

2.1.1 Linguaggi

Tabella 1:

Tecnologia	Descrizione	Versione
HTML	Linguaggio di annotazione (markup) utilizzato per impostare la struttura delle singole pagine e definire gli elementi dell'interfaccia	5
CSS	Linguaggio utilizzato per la formattazione e la gestione dello stile degli elementi HTML	3
JavaScript	Linguaggio utilizzato per la gestione di eventi invocati dall'utente	ECMAScript 2023
PHP	Linguaggio per la codifica di applicazioni web lato server, utilizzato per la creazione di *API Rest*^{G}	8.x

2.1.2 Librerie e framework

Tabella 2:

Tecnologia	Descrizione	Versione
ReactJs	Libreria grafica per facilitare lo sviluppo front-end gestendo modularmente le componenti grafiche, permettendo performance buone grazie all'efficacia della sua renderizzazione	18.2.x
Laravel	Framework PHP utilizzato per facilitare la creazione di API Rest	11
Axios	Libreria JavaScript che viene utilizzata per effettuare richieste HTTP sia negli ambienti browser che Node.js	11.x

2.1.3 Strumenti e servizi

Tabella 3:

Tecnologia	Descrizione	Versione
Node.js	Runtime system per esecuzione di codice Javascript	20.11.0
NPM	Gestore di pacchetti per il linguaggio JavaScript e l'ambiente di esecuzione Node.js	9.6.x

Continued on next page

Tabella 3: (Continued)

Docker	Piattaforma di sviluppo e gestione di applicazioni che permette di creare, distribuire e eseguire in software in container virtualizzati	24.0.7
Git	Sistema di controllo di versione distribuito utilizzato per la gestione del codice sorgente dal parte del gruppo di progetto	/
Bootstrap	Libreria di strumenti liberi per la creazione di siti e applicazioni per il Web che contiene modelli di progettazione basati su HTML e CSS	5.3.3

2.2 Tecnologie per l'analisi del codice

2.2.1 Analisi statica

Tabella 4:

Tecnologia	**Descrizione**	**Versione**
Jest	Framework di test basato su JavaScript con funzionalità di creazione di mock e il testing del codice in modo asincrono.	5.17

2.2.2 Analisi dinamica

Tabella 5:

Tecnologia	**Descrizione**	**Versione**
React Testing Library	Libreria di test integrata nativamente che consente di testare il comportamento dei componenti React da una prospettiva degli utenti finali.	13.4.0
GitHub Actions	Servizio di CI/CD per automatizzare il processo di build, test e deploy del progetto software	/

3 Architettura

3.1 Architettura di Deployment

SWEet16 ha deciso di adottare un'architettura a tre livelli per implementare *Easy Meal*.

Questa architettura va a separare la logica di presentazione dalla logica di business e dal database. Questi tre livelli comunicano tra di loro attraverso interfacce ben definite.

Nel dettaglio sono:

- **Livello di presentazione (frontend)**: Viene utilizzato React, che riceve i dati dall'utente e mostra le informazioni restituite dal backend;
- **Livello di logica applicativa (backend)**: Viene utilizzato Laravel, che gestisce la logica di business elaborando le richieste del frontend interagendo con il database per recuperare o salvare i dati;
- **Livello di Dati (database)**: Viene utilizzato MySQL, che si occupa della gestione dei dati.

Questo approccio ha permesso al team di lavorare in modo parallelo sul frontend e sul backend, riducendo i tempi di sviluppo e rendendo più facile l'implementazione dei test.

3.2 Architettura Front-end

3.2.1 Introduzione

Per il frontend del progetto si è scelto di utilizzare una combinazione di design pattern specifici della libreria ReactJS, selezionati e adattati secondo le specifiche necessità del progetto. Questo approccio permette di garantire la separazione della logica di business tra le varie componenti, semplificando la gestione degli stati dell'applicazione.
Ogni design pattern adottato è descritto dettagliatamente nelle sezioni successive.

Ogni pagina dell'applicazione utilizza chiamate alle API per recuperare i dati necessari alla sua visualizzazione. Inoltre, sfrutta gli hooks forniti da React, come useState e useEffect, per gestire lo stato dell'applicazione e aggiornare dinamicamente la UI. Grazie a questo approccio, l'applicazione è in grado di fornire un'esperienza utente fluida e reattiva, adattandosi in tempo reale alle esigenze dell'utente.

3.2.2 Design Pattern utilizzati

In questa sezione, vengono descritti i design pattern utilizzati nell'applicazione basata su React. Sono state adottate diverse soluzioni per modulare le singole componenti e adattarle alle specifiche esigenze di realizzazione ed integrazione previste.
In particolare, possiamo dettagliare l'utilizzo di:

React Hook

Utilizzati per gestire lo stato dell'applicazione in modo efficiente e visualizzare dinamicamente le informazioni. Vengono impiegati sia gli hook nativi di React (come useState, useEffect e useContext), sia hook personalizzati creati in base alle esigenze delle singole viste e componenti, come precedentemente descritto.

Conditional Rendering

Permette di mostrare contenuti diversi in base a determinate condizioni. Vengono sviluppati componenti in grado di verificare suddette condizioni e rendere visibili i dati pertinenti in base a queste.

L'esempio migliore di questo pattern si trova nel Layout, dove in base al ruolo dell'attore utilizzante il prodotto, l'header cambia al fine di mostrare le funzionalità disponibili quell'attore.

Compound Components

Consentono di modulare le singole componenti attraverso una gerarchia padre-figlio, dove un componente padre contiene uno o più componenti figlio. Questo approccio permette di specializzare la gestione dei dati e personalizzare l'interfaccia utente in modo centralizzato, seguendo una lista di opzioni unica.

L'esempio migliore di questo pattern si trova di nuovo nel Layout in sinergia con il router, dato un url il router compone la pagina aggregando assieme diversi componenti. Ad esempio la lista di ristoranti viene visualizzata sia da utenti anonimi sia da clienti che stanno effettuando una prenotazione, mentre non può essere visualizzata dal ristoratore.

```
1  // router.jsx
2
3  function Cliente ({Content}) {
4    const {role} = useStateContext()
5    if (role !== 'CLIENTE') {
6      return <Navigate to={"/Login"} />
7  }
8    return <Layout Content={Content} />
9  }
10
11 function Ristoratore ({Content}) {
12   const {role} = useStateContext()
13   if (role !== 'RISTORATORE') {
14     return <Navigate to={"/Login"} />
15 }
16   return <Layout Content={Content} />
17 }
```

3.3 Architettura Back-end

3.3.1 Introduzione

Per il backend del progetto, è stato scelto di adottare il framework Laravel. Laravel è un framework PHP noto per la sua semplicità e robustezza, che ci ha permesso di sviluppare RESTful API in modo rapido ed efficiente. Questa scelta consente di beneficiare delle potenti funzionalità di Laravel, come l'ORM Eloquent, la gestione delle migrazioni, e una struttura modulare che favorisce la manutenibilità e la scalabilità dell'applicazione.

Per il database è stato scelto MySQL, un sistema di gestione di database relazionali molto diffuso e affidabile. MySQL si integra perfettamente con Laravel, permettendo di sfruttare al meglio le funzionalità di entrambe le tecnologie.

Le singole componenti del sistema sono strutturate come segue:

- **Laravel**: Utilizzato per gestire la logica di business, l'autenticazione, la gestione delle rotte e la comunicazione con il database;
- **MySQL**: Utilizzato per la persistenza dei dati, MySQL offre una soluzione robusta e scalabile per gestire le informazioni necessarie all'applicazione. Grazie all'ORM Eloquent di Laravel, è possibile interagire con il database in modo intuitivo e efficiente;
- **API RESTful**: Sono state implementate delle API RESTful, che consentono una chiara separazione dei dati tra client e server. Le API RESTful permettono di esporre le risorse dell'applicazione tramite chiamate HTTP, rendendo possibile la comunicazione sincrona tra il frontend e il backend.

L'adozione di Laravel per il backend e MySQL per il database, quindi, consente di creare un'applicazione web potente, scalabile e sicura, con una chiara separazione della logica di business e una gestione efficiente dei dati.

3.3.2 Design Pattern utilizzati

Nella seguente sezione, vengono descritti i design pattern adottati per il backend. Seguendo la descrizione fornita inizialmente possiamo descrivere l'utilizzo di:

Facade Pattern

Fornisce un'interfaccia statica a classi che sono disponibili nel contenitore di servizio di Laravel, rendendo l'uso delle classi di servizio più semplice; sono state largamente usate nella comunicazione con il database;

```
1 // Models/Client.php
2 class Client extends Model
3 {
4     use HasFactory;
5
6     protected $fillable=['id','user','nome'];
7
8     public function allergie(): BelongsToMany
```

```
    {
        return $this->belongsToMany(Allergeni::class);
}

    protected $table='clients';

    public function user() {
        return $this->belongsTo(User::class, 'user');
}
}
```

Repository Pattern

Separazione della logica di accesso ai dati dal business logic, creando un livello di astrazione per le operazioni CRUD e altre query di database.

```
// Controllers/Api/ClientController.php
class ClientController extends Controller
{
    public function index() {
        $client=Client::get()->all();
       return response()->json($client,200);
}

    public function show($id) {
        /** @var Client $client */
        $client = Client::where('id',$id)->first();
         return response()->json([
             'id' => $client['id'],
             'nome' => $client['nome'],
             'user' => $client['user'],
         ]);
}

    public function store(ClientRequest $request) {
        ...
}

    public function update(UpdateClientRequest $request) {
        ...
}

    public function destroy(string $id) {
        ...
}
}
```

Factory Pattern

Utilizzato per creare oggetti senza dover specificare la classe esatta dell'oggetto che verrà creato. Utilizzato per generare istanze di modelli in fase di testing o seeding del database.

```
// database/factories/RistoratoreFactory.php
class RistoratoreFactory extends Factory
{
    protected $model = Ristoratore::class;

    public function definition()
    {
```

```
8           return [
9               'user' => User::factory(),
10              'nome' => $this->faker->unique()->company,
11              'cucina' => 'Italiana',
12              'indirizzo' => $this->faker->unique()->address,
13              'telefono' => $this->faker->unique()->numerify('##########'),
14              'capienza' => $this->faker->numberBetween(10, 200),
15              'orario' => '19:30 - 22:30'
16          ];
17  }
18  }
```

4 Requisiti soddisfatti

4.1 Requisiti funzionali

Di seguito la specifica per i requisiti funzionali, i quali descrivono le funzionalità del sistema, le azioni che il sistema può compiere e le informazioni che il sistema può fornire.
Ogni requisito presenta un codice identificativo univoco, così definito:

- **RFO**: Requisito Funzionale Obbligatorio;
- **RFF**: Requisito Funzionale Facoltativo;
- **RFD**: Requisito Funzionale Desiderabile.

Tabella 6:

Stato	**Codice**	**Descrizione**
Non soddisfatto	RFO01	L'utente non riconosciuto deve poter registrare un nuovo account
Non soddisfatto	RFO02	L'utente non riconosciuto deve visualizzare un messaggio d'errore se la registrazione non va a buon fine
Non soddisfatto	RFO03	L'utente non riconosciuto deve poter effettuare il login
Non soddisfatto	RFO04	L'utente non riconosciuto deve visualizzare un messaggio se il login non va a buon fine
Non soddisfatto	RFO05	L'utente autenticato deve poter modificare le informazioni del suo account
Non soddisfatto	RFO06	L'utente non riconosciuto deve poter effettuare il logout
Non soddisfatto	RFO07	L'utente autenticato deve poter creare un profilo di tipo cliente
Non soddisfatto	RFO08	L'utente autenticato deve poter creare un profilo di tipo ristoratore
Non soddisfatto	RFO09	L'utente autenticato deve visualizzare un messaggio d'errore se la creazione del profilo di tipo ristoratore non va a buon fine
Non soddisfatto	RFO10	L'utente autenticato deve poter cancellare un profilo
Non soddisfatto	RFO11	L'utente deve poter selezionare un qualunque profilo afferente al suo account
Non soddisfatto	RFO12	Il cliente deve poter modificare le informazioni relative al suo profilo
Non soddisfatto	RFO13	Il ristoratore deve poter modificare le informazioni relative al suo profilo
Non soddisfatto	RFO14	Il cliente e il ristoratore devono poter uscire dal loro profilo
Non soddisfatto	RFO15	L'utente non riconosciuto e il cliente devono poter visualizzare la lista di ristoranti disponibili
Non soddisfatto	RFO16	L'utente non riconosciuto e il cliente devono poter ricercare un ristorante attraverso il nome
Non soddisfatto	RFO17	L'utente non riconosciuto e il cliente devono poter applicare dei filtri alla ricerca di un ristorante

Continued on next page

Tabella 6: (Continued)

Non soddisfatto	RFO18	L'utente non riconosciuto e il cliente devono poter visualizzare le informazioni relative ad un particolare ristorante, il suo menù e le relative recensioni
Non soddisfatto	RFO19	Il cliente deve poter effettuare una prenotazione presso un ristorante
Non soddisfatto	RFO20	Il cliente deve poter consultare la lista delle sue prenotazioni
Non soddisfatto	RFO21	Il cliente deve poter annullare una sua prenotazione
Non soddisfatto	RFO22	Il cliente deve poter visualizzare le informazioni relative ad una sua prenotazione in particolare
Non soddisfatto	RFO23	Il cliente deve poter invitare altri clienti a partecipare ad una sua prenotazione
Non soddisfatto	RFO24	Il cliente deve poter accettare l'invito ricevuto, a partecipare ad una prenotazione
Non soddisfatto	RFO25	Il cliente deve poter effettuare un'ordinazione nel contesto di una sua prenotazione
Non soddisfatto	RFO26	Nell'effettuare l'ordinazione, il cliente deve poter apportare modifiche alle pietanze del ristorante, modificandone la quantità o rimuovendo ingredienti
Non soddisfatto	RFF01	Nell'effettuare l'ordinazione, il cliente deve poter aggiungere ingredienti da una pietanza
Non soddisfatto	RFO27	Il cliente deve poter annullare un'ordinazione
Non soddisfatto	RFO28	Il cliente deve poter selezionare la modalità secondo la quale dividere il conto con gli altri clienti afferenti alla stessa prenotazione
Non soddisfatto	RFF02	Il cliente dopo aver scelto la modalità di divisione "equa" può selezionare altri profili e pagare per loro
Non soddisfatto	RFF03	Il cliente dopo aver scelto la modalità di divisione "proporzionale" può selezionare pietanze ordinate da altri profili e pagarle
Non soddisfatto	RFF04	Il cliente deve visualizzare un messaggio d'errore se la suddetta modalità è già stata scelta da un altro cliente afferente alla stessa prenotazione
Non soddisfatto	RFF05	Il cliente deve poter effettuare il pagamento tramite l'app
Non soddisfatto	RFO29	Il cliente deve visualizzare un messaggio d'errore nel caso il pagamento non sia andato a buon fine
Non soddisfatto	RFO30	Il cliente deve poter rilasciare feedback e una recensione relativi alla sua esperienza in un ristorante
Non soddisfatto	RFO31	Il cliente deve ricevere una notifica quando una richiesta di prenotazione è accettata o rifiutata
Non soddisfatto	RFF06	Il cliente e il ristoratore devono ricevere una notifica in concomitanza della ricezione di un messaggio
Non soddisfatto	RFO32	Il cliente deve ricevere una notifica quando un cliente da egli invitato a partecipare ad una prenotazione, accetta l'invito
Non soddisfatto	RFO33	Il ristoratore deve poter accettare la richiesta di prenotazione da parte di un cliente

Continued on next page

Tabella 6: (Continued)

Non soddisfatto	RFO34	Il ristoratore deve poter rifiutare la richiesta di prenotazione da parte di un cliente
Non soddisfatto	RFO35	Il ristoratore deve poter visualizzare le prenotazioni presso il suo ristorante
Non soddisfatto	RFF07	Il ristoratore deve poter visualizzare le prenotazioni presso il suo ristorante implementate come un calendario con le prenotazioni raggruppate per giorno
Non soddisfatto	RFF08	Il ristoratore deve poter vedere il dettaglio degli ingredienti necessari per ogni giornata
Non soddisfatto	RFO36	Il ristoratore deve poter visualizzare i dettagli relativi ad una singola prenotazione
Non soddisfatto	RFO37	Il ristoratore deve poter vedere gli ingredienti necessari per soddisfare gli ordinati afferenti ad una singola prenotazione
Non soddisfatto	RFO38	Il ristoratore deve poter vedere le ordinazioni effettuate dai clienti che partecipano alla stessa prenotazione
Non soddisfatto	RFO39	Il ristoratore deve poter visualizzare lo stato del pagamento del conto relativo ad una prenotazione cioè se e quante ordinazioni o quote rimangono da pagare
Non soddisfatto	RFO40	Il ristoratore deve poter modificare il menù di un suo ristorante aggiungendo, modificando e/o eliminando le pietanze e le sezioni del menù stesso
Non soddisfatto	RFO41	Il ristoratore deve poter modificare la lista degli ingredienti presenti nelle pietanze del menù, aggiungendone di nuovi od eliminandone di già presenti
Non soddisfatto	RFO42	Il ristoratore deve, in fase di aggiunta di un nuovo ingrediente, poter associare specifici allergeni al suddetto ingrediente
Non soddisfatto	RFO43	Il ristoratore deve poter visualizzare le recensioni relative al suo ristorante
Non soddisfatto	RFO44	Il ristorante deve poter gestire manualmente il pagamento del conto relativo ad una prenotazione
Non soddisfatto	RFO45	Il ristoratore deve ricevere una notifica quando un cliente conferma un'ordinazione
Non soddisfatto	RFO46	Il ristoratore deve ricevere una notifica quando un cliente annulla una prenotazione
Non soddisfatto	RFO47	Il ristoratore deve ricevere una notifica quando è effettuata una nuova richiesta di prenotazione presso il suo ristorante
Non soddisfatto	RFO48	Il ristoratore deve ricevere una notifica quando un cliente rilascia una nuova recensione relativa ad un suo ristorante
Non soddisfatto	RFO49	Il ristoratore deve ricevere una notifica quando un cliente effettua un pagamento tramite l'app
Non soddisfatto	RFF09	Il ristoratore e il cliente devono poter visualizzare la/le loro chat
Non soddisfatto	RFF10	Il cliente deve poter aprire un canale di comunicazione tramite chat verso un ristoratore