

Report AWS: Gelateria

Alberto Antonelli - Andrea Pasero

August 31, 2024

Contents

1	Introduzione	3
2	Requisiti e analisi dei target users	4
2.1	Requisiti Funzionali	4
2.1.1	Autenticazione	4
2.1.2	Gestione dei Prodotti	4
2.1.3	Gestione degli Ordini	4
2.1.4	Applicazione di filtri sui Prodotti	5
2.1.5	Visualizzazione Prodotti	5
2.1.6	Effettuazione di un Ordine	5
2.1.7	Elenco Ordini	5
2.1.8	Visualizzazione delle statistiche	5
2.2	Requisiti Non Funzionali	5
2.3	Possibile scenario	6
2.3.1	Utente	6
2.3.2	Gestore	6
3	Design	7
3.1	Modellazione del dominio	7
3.2	Autenticazione	8
3.3	Profilo "utente"	9
3.4	Profilo "admin gestore"	10
4	Architettura	18
4.1	Architettura del sistema	18
4.2	Architettura del repository	18
5	Tecnologie	19
5.1	Tecnologie back-end	19
5.2	Tecnologie front-end	20
6	Test	22
6.1	Jest	22
6.2	Postman	23
7	Deployment	28
8	Conclusioni	29

1 Introduzione

L'obiettivo principale del progetto è la realizzazione di un'applicazione web avanzata e user-friendly che consenta al gestore di una gelateria di gestire in modo efficiente e organizzato gli ordini dei clienti. Questa piattaforma online offrirà al gestore un sistema intuitivo per aggiungere e aggiornare i diversi gusti di gelato disponibili, nonché le torte di gusto singolo, assicurando una gestione flessibile e dinamica del catalogo dei prodotti.

Grazie a questa applicazione, il gestore potrà gestire gli ordini ricevuti in maniera più snella, sia quelli effettuati per la consegna a domicilio che per il ritiro in negozio. Il sistema integrerà funzioni di notifica e aggiornamento dello stato degli ordini, permettendo al gestore di monitorare in tempo reale le richieste dei clienti e ottimizzare il processo di preparazione e consegna.

Dall'altra parte, i clienti potranno accedere alla piattaforma per esplorare l'ampia offerta della gelateria, visualizzare i prodotti e i loro dettagli come prezzo e disponibilità. Successivamente potranno effettuare ordini scegliendo il ritiro a domicilio o in negozio.

Inoltre, l'applicazione prevederà funzionalità avanzate per migliorare l'esperienza utente, come la visualizzazione dei 3 prodotti più ordinati che spesso sono anche quelli che vengono maggiormente richiesti.

2 Requisiti e analisi dei target users

La fase iniziale dell'elaborato ha previsto più sessioni di brainstorming al fine di elencare in maniera chiara e succinta tutti i requisiti funzionali e non funzionali.

2.1 Requisiti Funzionali

Le sottosezioni che seguono descrivono i requisiti funzionali.

2.1.1 Autenticazione

- **Login:** l'applicativo richiederà agli utenti di effettuare l'accesso mediante i campi email e password.
- **Registrazione:** l'applicativo permetterà agli utenti di effettuare una registrazione utilizzando un'email non ancora utilizzata e una password. Questa fase prevederà che l'utente registrato sia sempre un utente-cliente.

2.1.2 Gestione dei Prodotti

- **Elenco dei prodotti:** permette agli utenti di visualizzare l'elenco dei prodotti disponibili.
- **Visualizzazione di un prodotto:** permette al gestore di visualizzare nello specifico le caratteristiche di un prodotto.
- **Modifica di un prodotto:** permette al gestore di modificare la quantità disponibile di un prodotto.
- **Inserimento di un prodotto:** permette al gestore di inserire un prodotto.
- **Rimozione:** permette al gestore di rimuovere un prodotto dall'elenco, solo se non è presente in nessun ordine per finalità statistiche.

2.1.3 Gestione degli Ordini

- **Elenco degli ordini:** permette al gestore di visualizzare l'elenco degli ordini effettuati dai clienti e a quest'ultimo di vedere tutti i suoi ordini.
- **Visualizzazione di un ordine:** permette all'utente di visualizzare i dettagli di un ordine.
- **Modifica di un ordine:** permette al gestore di aggiornare lo stato di un ordine, anche di rifiutarlo.
- **Rimozione di un ordine:** permette al gestore di cancellare un ordine.

2.1.4 Applicazione di filtri sui Prodotti

Possibilità di applicare filtri come la distinzione tra gelato e torta.

2.1.5 Visualizzazione Prodotti

L'utente può visualizzare informazioni relative ad un prodotto, come specificate dal gestore in fase di creazione.

2.1.6 Effettuazione di un Ordine

L'utente può scegliere i prodotti desiderati, selezionare quantità, visualizzare il totale dell'ordine e decidere se ritirarlo o consegna al domicilio.

2.1.7 Elenco Ordini

L'utente ha a disposizione l'elenco degli ordini effettuati, con i relativi dettagli, per tracciare lo stato degli ordini.

2.1.8 Visualizzazione delle statistiche

L'admin può consultare le statistiche di ordini e prodotti.

2.2 Requisiti Non Funzionali

Tra i requisiti non funzionali emersi i principali sono:

- **Error Handling:** Eventuali errori devono essere nascosti all'utente finale, mediante una pagina che espone la non autorizzazione all'accesso e il redirect alla home.
- **Autorizzazione:** Le informazioni gestite all'interno della piattaforma devono essere erogate tenendo conto dei permessi dell'utente che le richiede. Dovranno essere quindi utilizzati dei meccanismi di autorizzazione per implementare questo comportamento.
- **Responsive Design:** Utilizzo di un layout mobile first per la web app, ma flessibilità tale da essere utilizzato anche da dispositivi con diverse dimensioni.
- **User-Friendly:** Utilizzo di interfacce semplici, per risultare il più possibile user-friendly.
- **API RESTful:** Le API dovranno essere implementate seguendo il modello RESTful e non dovranno esporre informazioni sensibili.

2.3 Possibile scenario

2.3.1 Utente

1. L'utente si registra e/o effettua il login.
2. L'utente entra nella visualizzazione prodotti.
3. L'utente sceglie uno o più prodotti selezionandolo a seconda della quantità desiderata, inserendoli nel carrello.
4. L'utente procede alla visualizzazione del carrello, qui può eliminare un prodotto e/o l'intero contenuto del carrello (torniamo al passo 3).
5. L'utente finalizza l'ordine scegliendo se riceverlo a domicilio o ritirarlo in negozio.
6. L'utente da questo momento in poi può visualizzare i prodotti ordinati e seguire l'avanzamento dell'ordine dalla sezione notifiche.
7. L'utente può utilizzare la chat (in qualsiasi momento) per fare domande ai gestori dell'applicazione.

2.3.2 Gestore

1. Il gestore aggiunge e modifica la quantità dei prodotti.
2. Il gestore riceve un ordine, modificandone l'avanzamento.
3. Il gestore risponde alle domande generiche del cliente attraverso la chat.
4. Il gestore contatta il cliente attraverso per informazioni su un ordine specifico.

3 Design

3.1 Modellazione del dominio

- **User:** rappresenta un'istanza di credenziali, identificato dall'email.
- **Message:** rappresenta un messaggio inviato da un utente ad un altro.
- **Notification:** rappresenta una notifica associata ad un ordine.
- **Order:** rappresenta un ordine effettuato da un utente.
- **Product:** rappresenta un prodotto disponibile per l'ordine.
- **Ingredient:** rappresenta un ingrediente utilizzato nella produzione dei prodotti.
- **Stock:** rappresenta il movimento di ingredienti in magazzino.

User

Campo	Tipo
username	String, required, unique
password	String, required
email	String, required, unique
ruolo	String, enum: [], required

Message

Campo	Tipo
from	ObjectId, ref: 'User', required
to	ObjectId, ref: 'User', required
content	String, required
timestamp	Date, default: Date.now

Notification

Campo	Tipo
username	String, required
notifyDate	Date, required
message	String, required
read	Boolean
orderId	ObjectId, ref: 'Order', required

Order

Campo	Tipo
creationDate	Date, required
closingDate	Date, optional
shippingDate	Date, optional
status	String, enum: [], required
note	String, optional
orderType	String, enum: [], optional
products	Array of Objects
user	ObjectId, ref: 'User', required

Product

Campo	Tipo
name	String, required
description	String, required
price	Number, required
disponibilita	Number, default: 0
type	String, enum: [], required

Ingredient

Campo	Tipo
name	String, required, unique
disponibilita	Number, required
measure	String, enum: [], required
mediumPrice	Number, required

Stock

Campo	Tipo
ingredientId	ObjectId, ref: 'Ingredient', required
movementQuantity	Number, required
typeMovement	String, enum: [], required
price	Number, required
insertDate	Date, required

3.2 Autenticazione

Arrivati alla pagina iniziale (figura 1), si avrà la possibilità di effettuare il login e/o la registrazione. La schermata di login e quella di registrazione sono analoghe: è necessario inserire username, email e la password per registrarsi. Una volta creato l'account, automaticamente verrà associato il profilo da "utente".

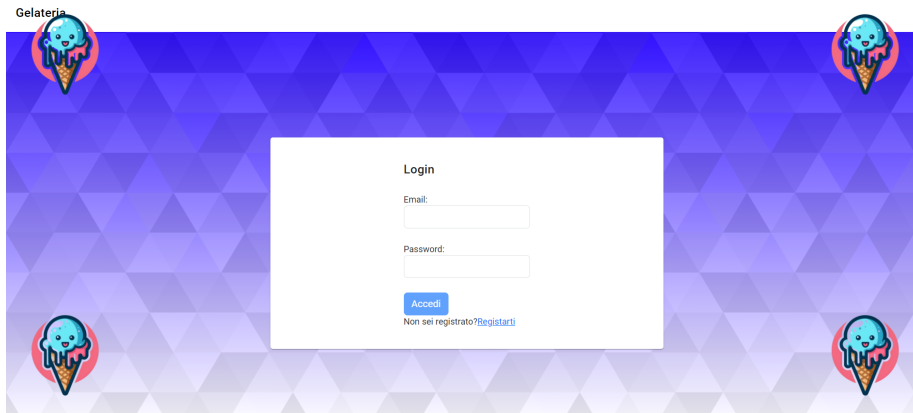


Figure 1: Login

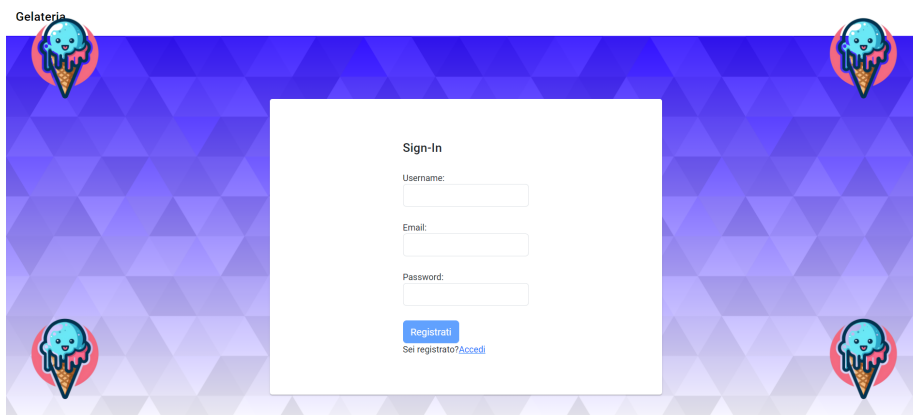


Figure 2: Register

L'idea di base è quella del riutilizzo del codice e dei componenti, visto che la maggior parte delle schermate risultano utili e necessarie sia per l'admin che per l'utente.

3.3 Profilo "utente"

Per il profilo "utente", vengono visualizzati i seguenti opzioni di menu:

- Home
- Prodotti
- Ordini
- Profilo

- Messaggi
- Esci/Logout

Inoltre vengono mostrate le icone per accedere al carrello, alla chat e alle notifiche.

La Home mostra 6 prodotti BestSeller sulla base dei prodotti più ordinati da tutti gli utenti, mentre nel menù Prodotti è possibile visitare la totalità dei prodotti ed aggiungerli al carrello. Il profilo invece mostra i 3 prodotti più ordinati e tutti gli ordini fatti dall'utente (figure 3, 4, e 5).

Qui si può notare il riutilizzo dei componenti.

Il menù ordini mostra gli ordini fatti dal utente loggato, divisi per lo stato dell'ordine (figura 6).

Il menù notifiche mostra tutti gli aggiornamenti dei vari ordini dell'utente, ognuna di esse può essere eliminata o impostata come letta/non letta (figura 7).

La figura 8 e 9 mostrano il carrello, la quantità di ogni prodotto e la possibilità di eliminarli.

3.4 Profilo "admin gestore"

Per il profilo "admin", vengono visualizzati i seguenti opzioni di menu:

- Home
- Prodotti
- Ordini
- Ingredienti
- Esci/Logout

Inoltre viene mostrata l'icona per la chat. Questa permette di comunicare con qualsiasi utente, mentre l'utente può interagire con un solo Admin anche se possono essere autenticati più di uno (figura 10). L'admin, rispetto all'utente, ha alcune funzionalità in più su ordini e prodotti come la possibilità di aggiungere quest'ultimi e di modificarne quantità e descrizione. Per quanto riguarda gli ordini può modificarne lo stato e contattare il cliente. Infine, l'admin ha la possibilità di visionare le statistiche per capire l'andamento della sua attività commerciale (figura 17).

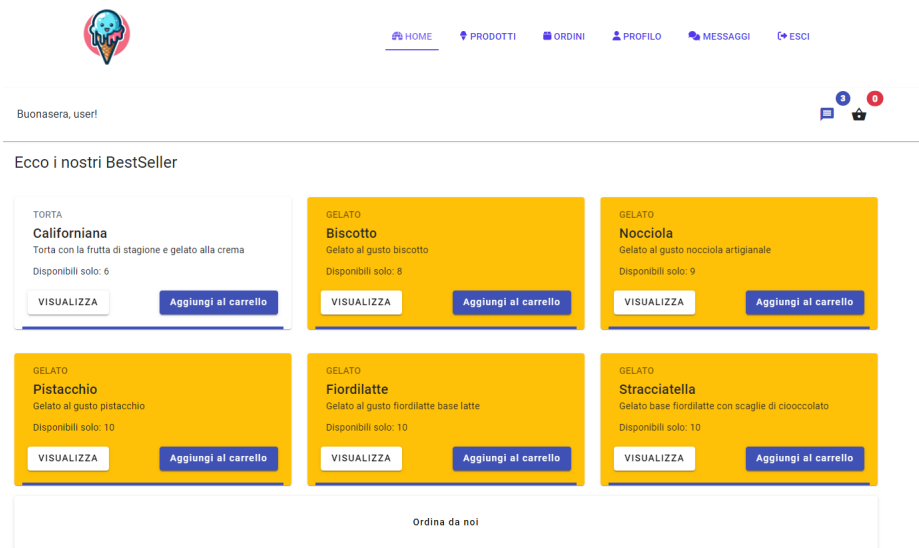


Figure 3: Home

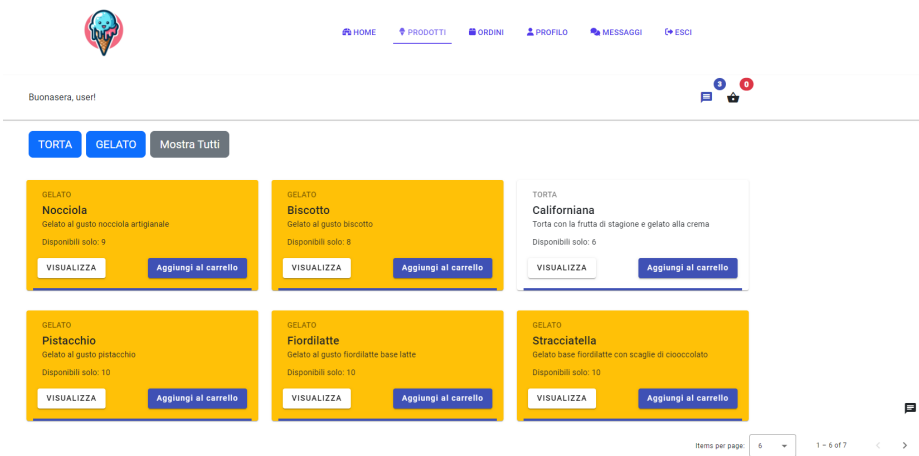


Figure 4: Prodotti

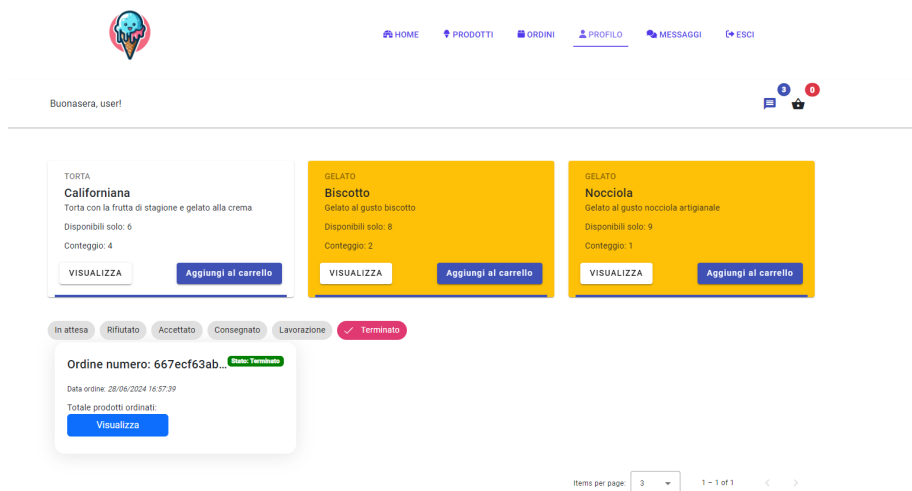


Figure 5: Profilo utente

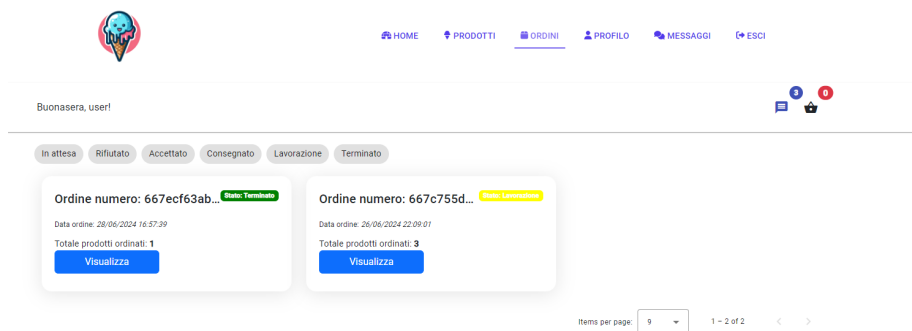



Figure 6: Ordini effettuati dall'utente



HOME PRODOTTI ORDINI PROFILO MESSAGGI ESCI

Buonasera, user!

Queste sono le tue notifiche


Tutte Non lette

N° Ordine	Messaggio	Letto?	Data		
667ecf63ab5a3431aa19d249	Il prodotto è nel seguente stato: terminato	<input checked="" type="checkbox"/>	08/07/2024 15:47:46	Visualizza	
667ecf63ab5a3431aa19d249	Il prodotto è nel seguente stato: terminato	<input checked="" type="checkbox"/>	08/07/2024 15:47:57	Visualizza	
667ecf63ab5a3431aa19d249	Il prodotto è nel seguente stato: terminato	<input type="checkbox"/>	08/07/2024 15:48:07	Visualizza	
667ecf63ab5a3431aa19d249	Il prodotto è nel seguente stato: terminato	<input checked="" type="checkbox"/>	08/07/2024 15:48:08	Visualizza	
667ecf63ab5a3431aa19d249	Il prodotto è nel seguente stato: terminato	<input checked="" type="checkbox"/>	08/07/2024 15:48:11	Visualizza	

Items per page: 5

1 - 5 of 7

Figure 7: Notifiche



HOME PRODOTTI ORDINI PROFILO MESSAGGI ESCI

Buonasera, user!

ECCO IL TUO CARRELLO


Svuota carrello

<div>GELATO</div> <div>Nocciola</div> <div>Gelato al gusto nocciola artigianale</div> <div>Quantità ordinata: 1</div> <div>VISUALIZZA Rimuovi prodotto</div>	<div>GELATO</div> <div>Biscotto</div> <div>Gelato al gusto biscotto</div> <div>Quantità ordinata: 1</div> <div>VISUALIZZA Rimuovi prodotto</div>	<div>TORTA</div> <div>Californiana</div> <div>Torta con la frutta di stagione e gelato alla crema</div> <div>Quantità ordinata: 1</div> <div>VISUALIZZA Rimuovi prodotto</div>
--	--	--

< Torna alla lista dei prodotti

> Procedi all'ordine

Figure 8: Carrello



HOME PRODOTTI ORDINI PROFILO MESSAGGI ESCI

Buonasera, user!

Tipologia Servizio

Ritiro in Negozio

Lascia delle note per noi

Invia Ordine

Figure 9: Conferma dell'ordine

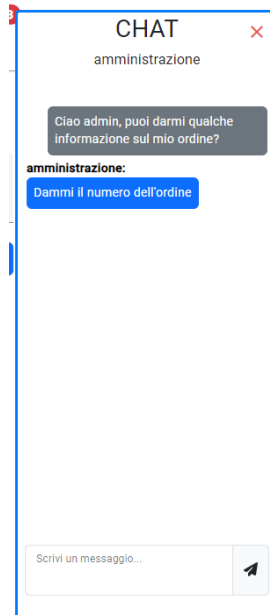


Figure 10: Chat tra cliente e admin

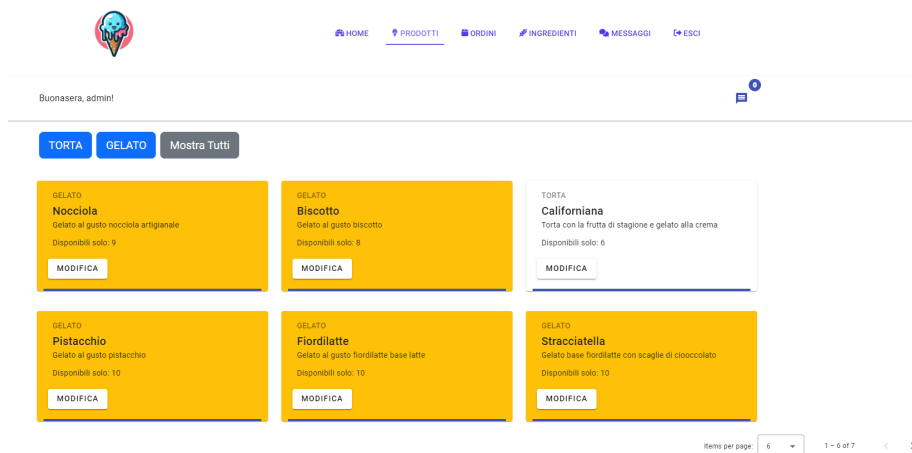


Figure 11: Prodotti: visualizzazione admin

Buonasera, admin!

Ingredienti

Zucchero - 10.00 Kg - €1.00	
Latte - 10.00 L - €1.50	
Nocciola - 10.00 Kg - €13.30	
Pistacchio - 10.00 Kg - €17.40	

Aggiungi un nuovo ingrediente

Nome


Disponibilità

Misura:

Prezzo Medio

[Aggiungi Ingrediente](#)

Figure 12: Ingredienti



[HOME](#) [PRODOTTI](#) [ORDINI](#) [INGREDIENTI](#) [MESSAGGI](#) [ESCI](#)

Buonasera, admin!



GELATO

Nocciola

Gelato al gusto nocciola artigianale


Disponibili solo: 9

[Elimina prodotto](#) [Modifica descrizione](#)

[Upload quantità](#)

Figure 13: Modifica prodotto



[HOME](#) [PRODOTTI](#) [ORDINI](#) [INGREDIENTI](#) [MESSAGGI](#) [ESCI](#)

Buonasera, admin!

0

ID Ordine 667ecf63ab5a3431aa19d249

Tipo di consegna: ritiro

Stato dell'ordine
terminato

Data creazione ordine: 28/06/2024 16:57:39
Data fine ordine: 08/07/2024 15:48:15
Data spedizione ordine: 04/07/2024 19:19:04

Note:
mi va il gelato

Contatta il cliente

Prodotti


Totale prodotti: 1

Totale: 15.00€

Torna alla lista

Salva Modifiche

Figure 14: Modifica stato ordine



[HOME](#) [PRODOTTI](#) [ORDINI](#) [INGREDIENTI](#) [MESSAGGI](#) [ESCI](#)

Buonasera, admin!

0

Compila la form con i dettagli del nuovo gusto oppure della nuova torta

Nome del prodotto:*

Descrizione:*

Prezzo:*

Disponibilità:*

Tipo:*

Crea Prodotto

Figure 15: Creazione di un prodotto

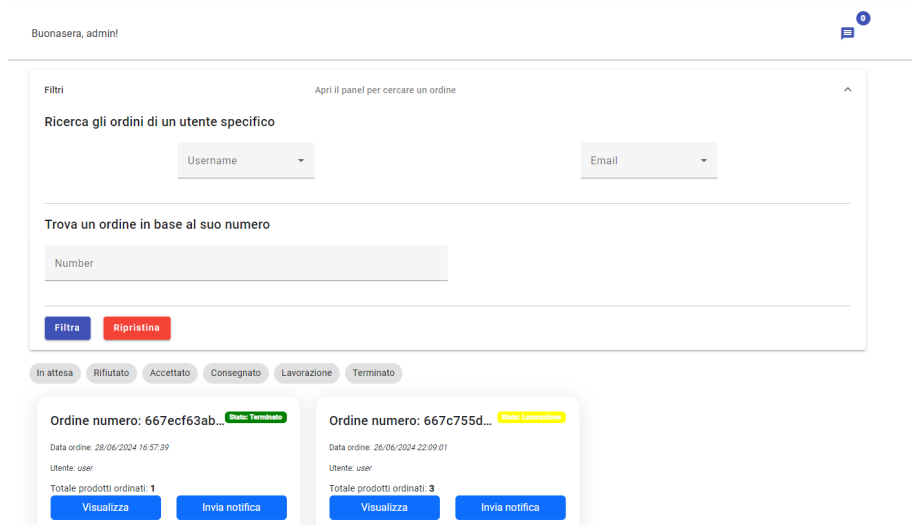


Figure 16: Ordini: visualizzazione Admin



Figure 17: Statistiche

4 Architettura

In questo capitolo verrà descritta l'architettura del sistema che verrà realizzato. In particolare verranno elencate e motivate quelle che sono le scelte principali che avranno un impatto sulla realizzazione dell'elaborato.

4.1 Architettura del sistema

Il sistema che verrà implementato prevede la realizzazione di due componenti software:

- Sistema front-end: Single-page web application servita tramite un hosting statico, che ospiterà le interfacce di interazione degli utenti con il sistema. Questa comunicherà con il Sistema back-end attraverso delle API HTTP.
- Sistema back-end: Applicazione back-end, che espone attraverso un'interfaccia HTTP l'insieme di operazioni necessarie all'esecuzione del sistema.

Architettura del back-end

Essa prevede la separazione della logica applicativa, da quella del modello, da quella delle modalità di erogazione del servizio (REST, SOAP, ecc.).

I livelli di separazione logica implementati sono:

- Model: Rappresenta l'integrazione tra un sistema ad oggetti come JavaScript e un database, relazionale e non. Permette di svolgere semplici operazioni CRUD, senza la necessità di utilizzare specifici Query Language.
- Service: Rappresenta una porzione della logica di business dell'applicazione. Utilizza e opzionalmente combina istanze specifiche dei modelli per implementare operazioni non banali, come per esempio l'autenticazione di un utente.
- Controller : Permette di collegare 1:1 operazioni di un servizio alla logica HTTP. Specifica quali servizi e quali operazioni devono essere chiamati, e come deve avvenire l'eventuale passaggio di parametri.
- Router : Permette di collegare la logica dei Controller ad una specifica rotta HTTP. Solitamente i Controller vengono raggruppati all'interno dei Router a seconda delle aree del dominio di cui si occupano.

4.2 Architettura del repository

Dato che sia la componente front-end che quella back-end dovranno essere realizzate utilizzando il linguaggio Typescript, le due componenti sono state organizzate all'interno di un monorepo, ovvero la tecnica di includere due o più applicazioni, anche indipendenti, all'interno dello stesso repository. In questo modo è possibile condividere agevolmente del codice, come per esempio la definizione dei tipi delle risorse.

5 Tecnologie

Lo sviluppo dell'intero sistema è basato sullo stack MEAN (MongoDB, Express, Angular e Node.js). Il vantaggio principale di MEAN è che permette di costruire applicazioni full-stack utilizzando un unico linguaggio, consentendo di riutilizzare il codice e di mantenerlo strutturalmente e sintatticamente coerente. La riusabilità può accelerare notevolmente il processo di sviluppo, consentendo di concentrarsi su attività più complesse.

MEAN è anche eccezionalmente scalabile, il che lo rende una scelta popolare per la realizzazione di applicazioni web su larga scala. L'architettura asincrona di Node.js consente una facile scalabilità orizzontale e MongoDB supporta lo sharding. È possibile distribuire l'archiviazione dei dati su numerosi server e aggiungere altri server ai propri cluster man mano che la domanda aumenta.

Inoltre, MEAN è ricco di librerie, framework e altri strumenti all'interno del gestore di pacchetti Node.js (npm). Questo ricco ecosistema permette di integrare facilmente le funzionalità nella propria applicazione senza doverle costruire da zero.

In questo capitolo verranno descritte le principali scelte tecnologiche effettuate.

5.1 Tecnologie back-end

Express

Express.js è un framework flessibile e leggero per lo sviluppo backend di applicazioni Node.js. Funge da middleware per garantire un'interazione fluida tra il client e il database. Offre inoltre solide funzionalità di routing e un gestore di errori predefinito.

Node.js

Node.js è un ambiente runtime JavaScript open-source e multiplatforma. Consente l'esecuzione di JavaScript lato server e fornisce un'architettura I/O non bloccante e guidata dagli eventi. La sua natura asincrona consente di gestire più richieste simultanee senza bloccare l'esecuzione di altro codice.

Socket.io

Socket.io è una libreria JavaScript che abilita la comunicazione bidirezionale e in tempo reale tra client e server. È costruita su Node.js e utilizza WebSockets, insieme ad altre tecnologie di fallback, per garantire la trasmissione efficiente dei dati in vari ambienti di rete.

Socket.io è particolarmente utile per applicazioni che richiedono aggiornamenti in tempo reale, come chat, notifiche live, giochi online e collaborazioni in tempo reale. La libreria gestisce automaticamente la riconnessione e la gestione degli eventi, offrendo un'API semplice e robusta.

Body-Parser

Body-Parser è un middleware per Express che analizza il corpo delle richieste HTTP, rendendo i dati facilmente accessibili tramite `req.body`. Supporta diversi formati, come JSON e URL-encoded.

Bcrypt

Bcrypt è una libreria per Node.js utilizzata per l'hashing sicuro delle password. Utilizza un algoritmo di hashing adattivo che rende più difficile la compromissione delle password attraverso attacchi di forza bruta.

JSON Web Tokens (JWT)

JSON Web Tokens (JWT) è una libreria per la creazione e la verifica di token basati su JSON. È comunemente utilizzata per l'autenticazione e l'autorizzazione tra client e server, garantendo un metodo sicuro per trasmettere informazioni tra le parti.

5.2 Tecnologie front-end

Angular

Angular è un framework JavaScript per lo sviluppo di applicazioni frontend. Offre funzionalità come il binding bidirezionale dei dati e l'iniezione di dipendenze per consentire visualizzazioni dinamiche, semplificando la creazione di interfacce utente complesse e interattive.

Angular Material

Angular Material è un'implementazione di Google Material Design in **Angular**. Si tratta quindi di una libreria di componenti dell'interfaccia utente (UI) per gli sviluppatori che usano **Angular**.

Bootstrap

Bootstrap è una popolare libreria CSS che facilita la creazione di interfacce utente reattive e moderne. Fornisce griglie, componenti pre-stilizzati e plugin JavaScript per velocizzare lo sviluppo frontend.

RxJS

RxJS (Reactive Extensions for JavaScript) è una libreria per la programmazione reattiva che utilizza osservabili per semplificare la gestione di eventi asincroni e stream di dati. È integrata in Angular per il binding dei dati e la gestione delle richieste HTTP.

JWT-decode

JWT-decode è una libreria che permette di decodificare JSON Web Tokens (JWT) nel frontend senza richiedere una firma segreta. È utile per estrarre informazioni memorizzate nel payload dei token JWT.

ApexCharts

ApexCharts è una libreria JavaScript per la creazione di grafici interattivi e reattivi. Supporta una vasta gamma di tipi di grafici ed è facile da integrare con framework frontend come Angular e React.

6 Test

Le funzionalità del sistema sono state testate sui browser Chromium e Firefox sia in versione desktop che mobile (mediante i tool di sviluppo integrati nel browser). I test hanno voluto verificare che le funzionalità, i task principali e la parte grafica funzionassero correttamente.

Durante la realizzazione del progetto, sono stati adottati due approcci principali per testare le funzionalità realizzate:

- Postman
- Jest

6.1 Jest

Il framework Jest permette di realizzare unit test su codice Javascript. È stato utilizzato per testare il controller del primo componente con cui ci si interfaccia, ossia quello relativo all'autenticazione. Nello specifico, si verifica se l'autenticazione avviene con successo con un utente registrato, e viceversa che non avvenga.

```
describe('POST /api/user/login', () => {
  it('should login an existing user', async () => {
    const credentials = {
      email: 'user@user.it',
      password: '123456'
    };

    const res = await request(app)
      .post('/api/user/login')
      .send(credentials)
      .expect(200);

    // Verifica che la risposta contenga il messaggio di login
    // riuscito e il token
    expect(res.body.isValid).toBe(true);
    expect(res.body.message).toBe('Login successful');
    expect(res.body).toHaveProperty('token');

    // Verifica che il token sia valido decodificandolo
    const decodedToken = jwt.verify(res.body.token, '
      chiaveSegreta');
    expect(decodedToken.username).toBe('user');
  });
});
```

Listing 1: Esempio di test per il login

```
it('should return 400 if user is not found', async () => {
  const credentials = {
    email: 'nonexistentuser@example.com',
    password: '123456'
  };
});
```

```

    const res = await request(app)
      .post('/api/user/login')
      .send(credentials)
      .expect(400);

    // Verifica che la risposta contenga il messaggio di errore
    // appropriato
    expect(res.body.isValid).toBe(false);
    expect(res.body.message).toBe('User not found');
  });

  it('should return 400 if password is incorrect', async () => {
    const credentials = {
      email: 'user@user.it',
      password: 'wrongpassword'
    };

    const res = await request(app)
      .post('/api/user/login')
      .send(credentials)
      .expect(400);

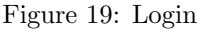
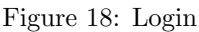
    // Verifica che la risposta contenga il messaggio di errore
    // appropriato
    expect(res.body.isValid).toBe(false);
    expect(res.body.message).toBe('Incorrect password');
  });

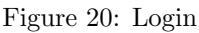
```

Listing 2: Altri test del Login

6.2 Postman

Nella realizzazione degli endpoint, si è verificato puntualmente che essi rispondessero nel modo corretto, effettuando simulazioni tramite Postman, con body di dati corrispondenti a quelli che avrebbe inviato la parte client.





GET ▼ http://localhost:3000/api/product

Params Authorization **Headers (9)** Body Scripts Tests Settings

Headers 7 hidden

	Key	Value	Description
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaW...	
<input checked="" type="checkbox"/>	userId	6583645f340b9b6eedd3a7f8	

Body Cookies Headers (11) Test Results ⊕ Status: 200 OK Time: 152 ms Size: 1.2 KB

Pretty Raw Preview Visualize JSON ▼ ≡

```

1  [
2    {
3      "_id": "65e8ac19433b2b9e6bea4cee",
4      "name": "Amarena",
5      "description": "1",
6      "price": 1,
7      "disponibilita": 0,
8      "type": "GELATO",
9      "___v": 0
10   },
11   {
12     "_id": "65eb2d3d770a33dd478b74b3",
13     "name": "Torta al cioccolato",
14     "description": "Deliziosa torta al cioccolato fondente",
15     "price": 10,
16     "disponibilita": 0,
17     "type": "TORTA"
18   },
19   {
20     "id": "65eb2d57770a33dd478b74b5"
  
```

Figure 21: Login

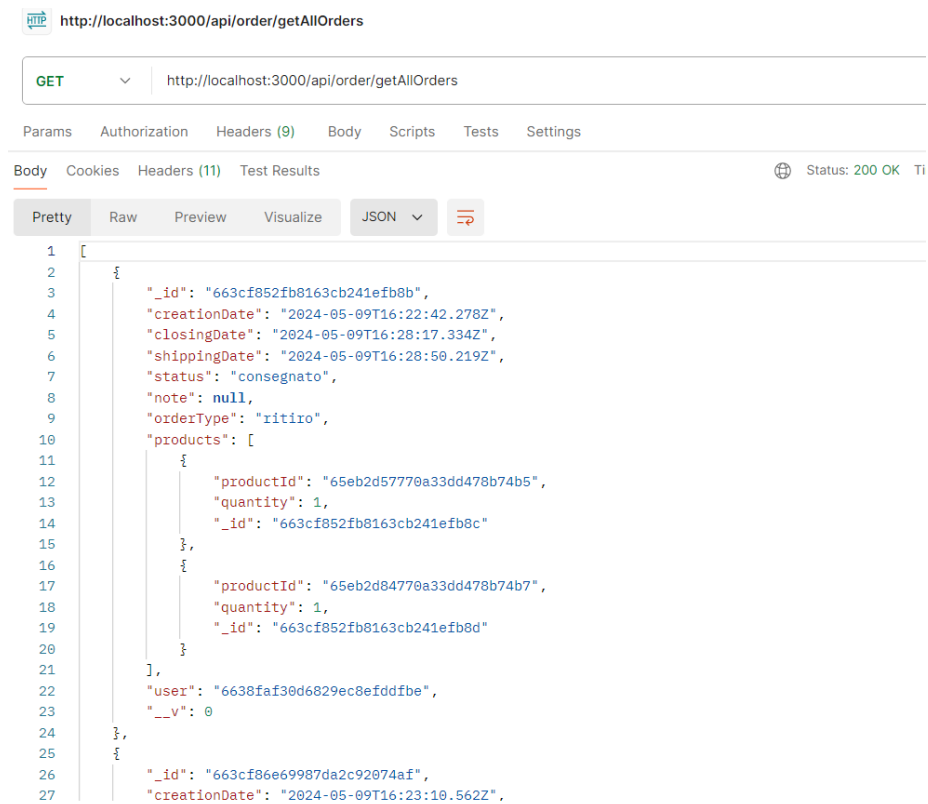


Figure 22: Login

7 Deployment

Dopo aver completato lo sviluppo delle API, è stato effettuato il deployment su Heroku, una piattaforma cloud che facilita il rilascio, la gestione e il ridimensionamento di applicazioni. Heroku è stato scelto per la sua semplicità d'uso e per l'integrazione fluida con il nostro stack tecnologico.

Le API sono state configurate e caricate correttamente, garantendo accessibilità e scalabilità. Attualmente, le API sono raggiungibili all'indirizzo <https://glacial-ocean-94844-f96f9789c382.herokuapp.com/api>.

Questo ci consente di rendere i servizi API disponibili in modo affidabile per gli utenti finali, sfruttando l'infrastruttura cloud offerta da Heroku.

In alternativa, per avviare il sistema bisogna lanciare l'applicazione Angular e il server Node.

I passi da fare sono i seguenti:

1. posizionarsi con un terminale dove si vuole scaricare il progetto e clonare il repository
2. `git clone https://github.com/Bilashos/Gelateria.git`
3. spostarsi nella directory backend
4. eseguire nel terminale `npm install`
5. spostarsi nella directory frontend con un'altra finestra del terminale
6. eseguire nel terminale `npm install`
7. avviare il server node (backend)
8. eseguire `npm start` per il frontend
9. collegarsi da browser attraverso il seguente indirizzo `https://localhost:4200`

Per l'utilizzo dell'applicativo si possono usare le seguenti credenziali:

- ADMIN. email: `admin@admin.it` pw: 123456
- USER. email: `user@user.it` pw: 123456

8 Conclusioni

Lo sviluppo di questo progetto ha permesso al team di arricchire il proprio bagaglio di conoscenze tecniche, dando la possibilità di mettersi alla prova con diverse tecnologie dello sviluppo web moderno che risultano essere molto apprezzate nel mondo del lavoro.

E' stato realizzato un sistema di prenotazione di una gelateria per torte e gusti con interazione con il cliente sia sottoforma di chat che messaggio diretto in caso di ordine.

Questo sistema è stato realizzato tramite una single page web application e un sottosistema back-end, che espone le proprie funzionalità mediante un'interfaccia HTTP.