Gruppo 13

Progetto:

Paninaro Next

Titolo del documento:

Sviluppo Applicazione

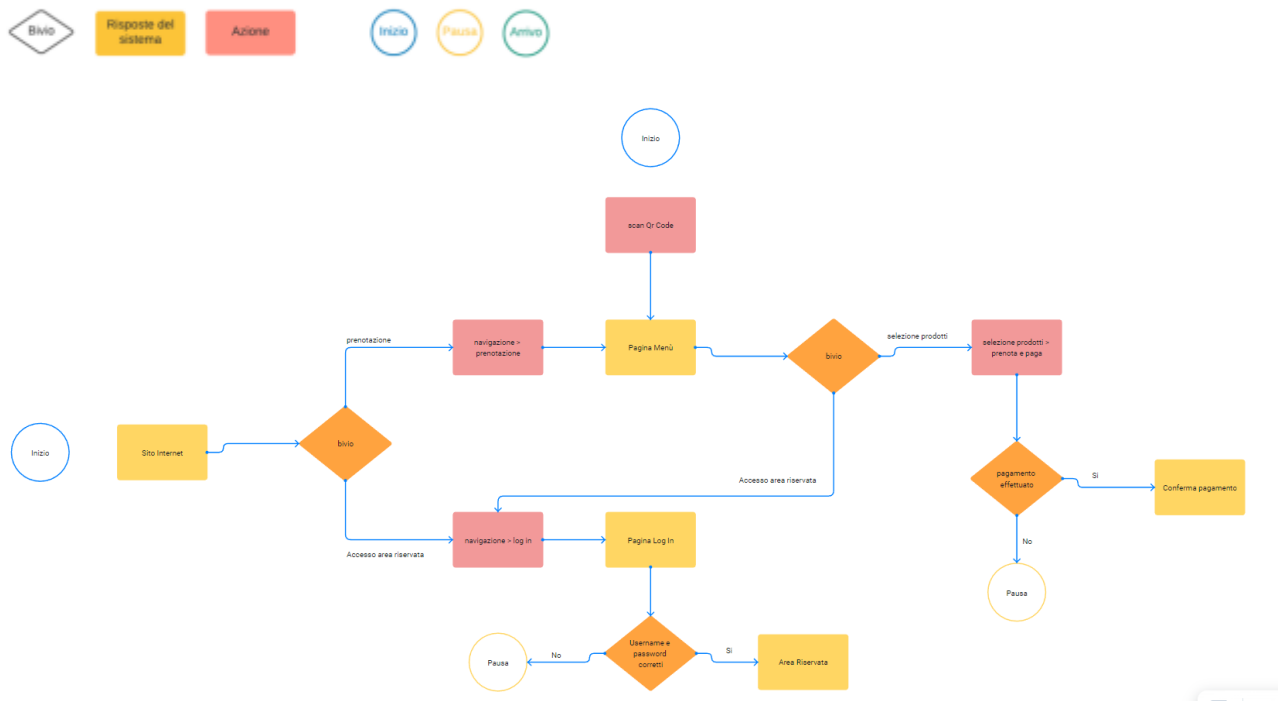
INDICE

Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte dell'applicazione Paninaro Next.
Partendo dalla descrizione degli user flow legate al ruolo del responsabile amministrativo dell'applicazione, il documento prosegue con la presentazione delle API necessarie (tramite l'API Model e il Modello delle risorse) per poter visualizzare, inserire e modificare sia i clienti che gli ingredienti nel database.
Per ogni API realizzata, oltre ad una descrizione delle funzionalità fornite, il documento presenta la sua documentazione e i test effettuati. Infine una sezione è dedicata alle informazioni del Git Repository e il deployment dell'applicazione stessa.

1. User Flows

In questa sezione del documento di sviluppo riportiamo gli "user flows", descritti dalla figura sottostante, con una legenda.



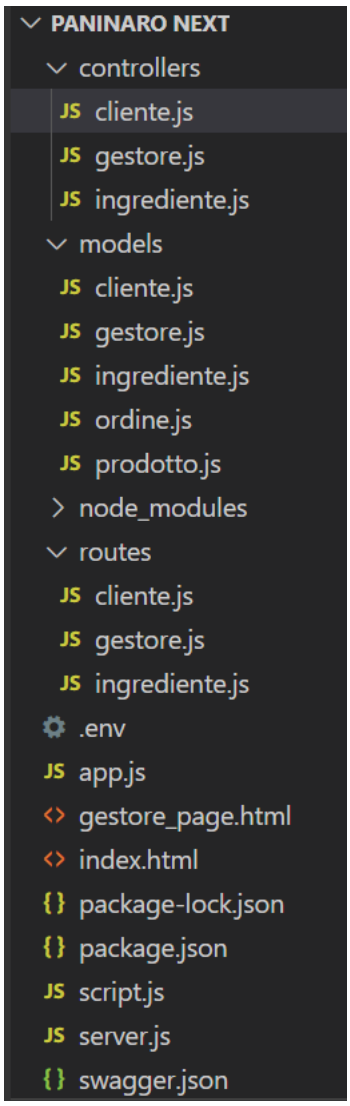
Si nota come un utente cliente possa accedere all'applicazione sia cercandola con un motore di ricerca, sia scannerizzando un apposito QR code. D'altro canto l'amministratore può accedere alla piattaforma tramite le sue credenziali per entrare nell'area riservata.

2. Application Implementation and Documentation

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

2.1. Project Structure

La struttura del progetto è presentata in figura:



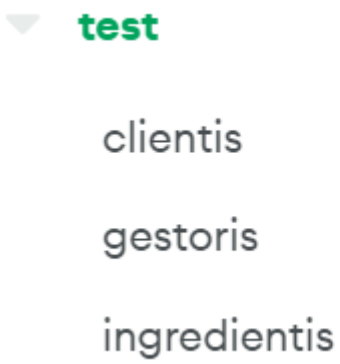
2.2. Project Dependencies

I seguenti moduli Node sono stati utilizzati e aggiunti al file Package.Json

- Express:
- Express-file upload
- MongoDB
- Cors
- Body-parser

2.3. Project Data or DB

Per la gestione dei dati utili all'applicazione abbiamo definito le strutture dati come illustrato in figura.



Per rappresentare i vari tipi di dati:

Per la tabella Clienti

```
_id: ObjectId('64cbd4103f43701a99ab824b')
conto: 30
nr_conto: 3
```

Per la tabella Ingredienti

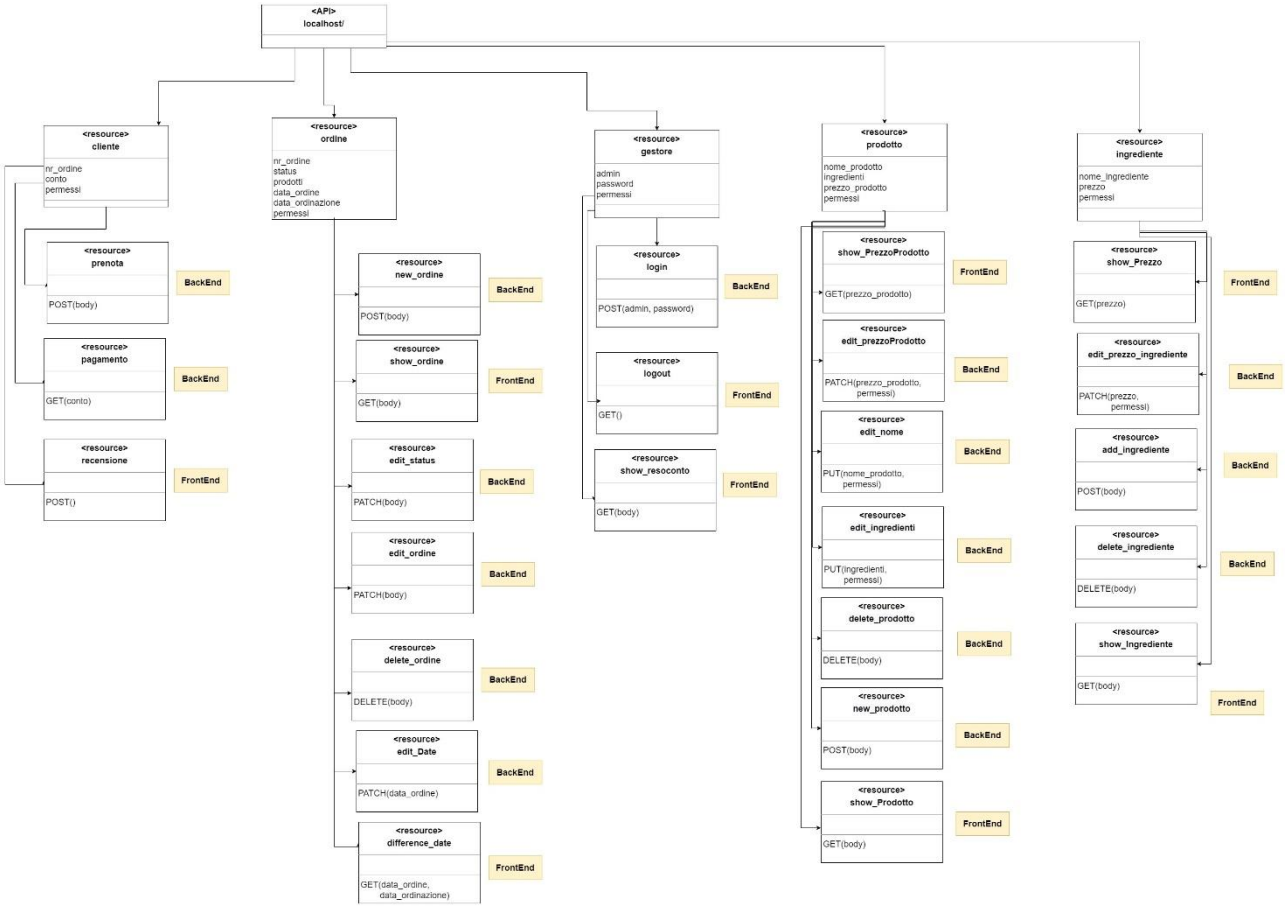
```
_id: ObjectId('64cc128c759c22083e1d3c45')
nome: "pecorino"
prezzo: 5
```

Per la tabella Gestori

```
_id: ObjectId('64ef21be32365a78163eb07d')
admin: "admin"
password: "password"
permessi: true
```

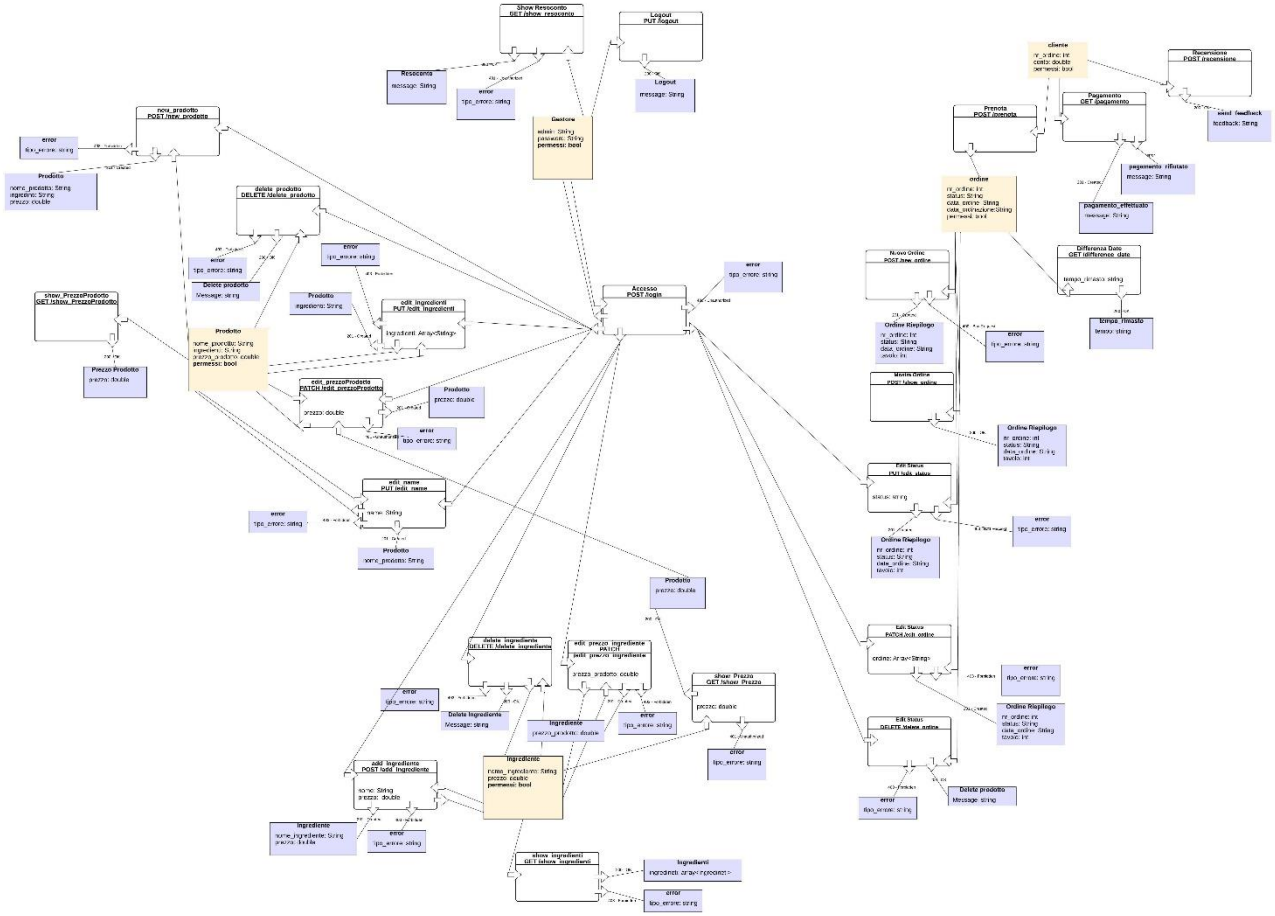
2.4. Project APIs

2.4.1. Resources Extraction from the Class Diagram



2.4.2. Resources Models

Ecco i Resources models:



2.5. Sviluppo API

Ecco le varie API sviluppate:

Routers	HTTP Methods	Description
/cliente/:nr_ordine	DELETE	Elimina l'ordine associato al cliente
/cliente	GET	Mostra tutti i clienti
/cliente/:nr_ordine	GET	Mostra un cliente specifico associato al <nr_ordine>
/cliente	POST	Aggiunge un nuovo cliente
/cliente	PATCH	Modifica l'ordine del cliente
/ingrediente/:nome	DELETE	Elimina l'ingrediente <nome>
/ingrediente	GET	Mostra tutti gli ingredienti
/ingrediente/:nome	GET	Mostra un ingrediente specifico <nome>
/ingrediente	POST	Aggiunge un nuovo ingrediente
/ingrediente	PATCH	Modifica il prezzo dell'ingrediente
/gestore	POST	Accesso alla pagina gestore

2.5.1 Elenco Ingredienti/Clienti

Mostra un elenco di tutti gli ingredienti/clienti presenti nel database

```
// GET '/ingrediente'
const show_Ingredienti = (req, res) => {

    //find the specific ingrediente with that name
    Ingredienti.find({ }, (err, data) => {
        if (err || !data) {
            return res.status(400).json({ message: "Ingredienti not present." });
        }
        else return res.status(200).json(data); //return the ingredienti objects if found
    });
}
```

```
// GET '/cliente'
const show_Clienti = (req, res) => {
    Clienti.find({}, (err, data) => {
        if (err || !data) {
            return res.json({ message: "Clienti not present." });
        }
        else return res.status(200).json(data); //return the clienti objects if found
    });
}
```

2.5.1 Trova Ingrediente/Cliente

Trova un ingrediente/cliente in base al nome/nr_conto

```
// GET '/ingrediente/nome'
const show_Prezzo = (req, res) => {
    let input_nome = req.params.nome; //get the ingrediente name

    //find the specific ingrediente with that name
    Ingredienti.findOne({ nome: input_nome }, (err, data) => {
        if (err || !data) {
            return res.json({ message: "Ingrediente doesn't exist." });
        }
        else return res.status(200).json(data); //return the ingrediente object if found
    });
}
```

```
// GET '/cliente/nr_conto'
const show_Conto = (req, res) => {
    let input_nr_conto = parseInt(req.params.nr_conto); //get the ingrediente name
    var query = { nr_conto: input_nr_conto };

    //find the specific cliente with that number
    Clienti.findOne(query, (err, data) => {
        if (err || !data) {
            return res.json({ message: "Cliente doesn't exist." });
        }
        else return res.status(200).json(data); //return the cliente object if found
    });
}
```

2.5.1 Modifica Ingrediente/Cliente

Modifica il prezzo di un ingrediente dato il nome, e il conto di un cliente dato il nr_conto

```
// PATCH '/ingrediente'
const edit_prezzo_ingrediente = (req, res) => {
    let input_nome = req.body.nome; //get the ingrediente name
    let input_prezzo = req.body.prezzo; //get the ingrediente name


    if(input_prezzo == []){
        return res.status(400).json({ message: "Null value."});
    }
    else{
    //find the specific ingrediente with that name
    Ingredienti.findOne({ nome: input_nome }, (err, data) => {
        if (err || !data) {
            return res.status(400).json({ message: "Ingrediente doesn't exist." });
        }
        else{
            Ingredienti.replaceOne({ nome: input_nome }, {nome: input_nome, prezzo: input_prezzo}, (err, data) => {
                if (err || !data) {
                        return res.status(400).json({ message: "Something went wrong, please try again." });
                }else{
                    return res.status(201).json({ message: "Modified ingrediente."});
                }});
        }

    });
    }
}
```

```
// PATCH '/cliente'
const edit_conto_cliente = (req, res) => {
    let input_nr_ordine = req.body.nr_ordine; //get the conto number
    let input_conto = req.body.conto; //get the ingrediente name

    //find the specific ingrediente with that name
    Clienti.findOne({ nr_ordine: input_nr_ordine }, (err, data) => {
        if (err || !data) {
            return res.json({ message: "Cliente doesn't exist." });
        }
        else{
            Clienti.replaceOne({ nr_ordine: input_nr_ordine }, { conto: input_conto, nr_ordine: input_nr_ordine }, (err, data) => {
            if (err || !data) {
                    return res.json({ message: "Something went wrong, please try again." });
            }else{
                return res.status(201).json({ message: "Modified cliente."});
            }});
        }

    });
}
```

2.5.2 Creazione di un Ingrediente/Cliente

Crea un ingrediente/cliente

```
// POST '/ingrediente'
const add_ingrediente = (req, res) => {
    //check if the ingrediente name already exists in db
    Ingredienti.findOne({ nome: req.body.nome }, (err, data) => {

        //if ingr. not in db and nome,prezzo not null, add it
        if (!data && req.body.nome!=[] && req.body.prezzo!=[]) {
            //create a new ingr. object using the Ingrediente model and req.body
            const newIngrediente = new Ingredienti({
                nome: req.body.nome,
                prezzo: req.body.prezzo,
            })


            // save this object to database
            newIngrediente.save((err, data) => {
                if (err) return res.status(400).json({ Error: err });
                return res.status(201).json(data);
            })
            //if there's an error or the ingr. is in db, return a message
        } else {
            if (err)
                return res.status(400).json(`Something went wrong, please try again. ${err}`);
            else if(req.body.nome==[] || req.body.prezzo==[])
                return res.status(400).json({ message: "Null value."});
            else
            return res.status(400).json({ message: "Ingrediente already exists" });
        }
    })
};
```

```
// POST '/cliente'
const add_cliente = (req, res) => {
    //check if the cliente name already exists in db
    Clienti.findOne({ nr_conto: req.body.nr_conto }, (err, data) => {
        //if ingr. not in db, add it
        if (!data) {
            //create a new cliente object using the Ingrediente model and req.body
            const newCliente = new Clienti({
                nr_conto: req.body.nr_conto,
                conto: req.body.conto,
            })

            // save this object to database
            newCliente.save((err, data) => {
                if (err) return res.json({ Error: err });
                return res.status(201).json(data);
            })
            //if there's an error or the cliente is in db, return a message
        } else {
            if (err) return res.json(`Something went wrong, please try again. ${err}`);
            return res.json({ message: "Ingrediente already exists" });
        }
    })
};
```

2.5.3 Cancellazione di un Ingrediente/Cliente

Cancella un ingrediente/cliente dato il nome/nr_conto

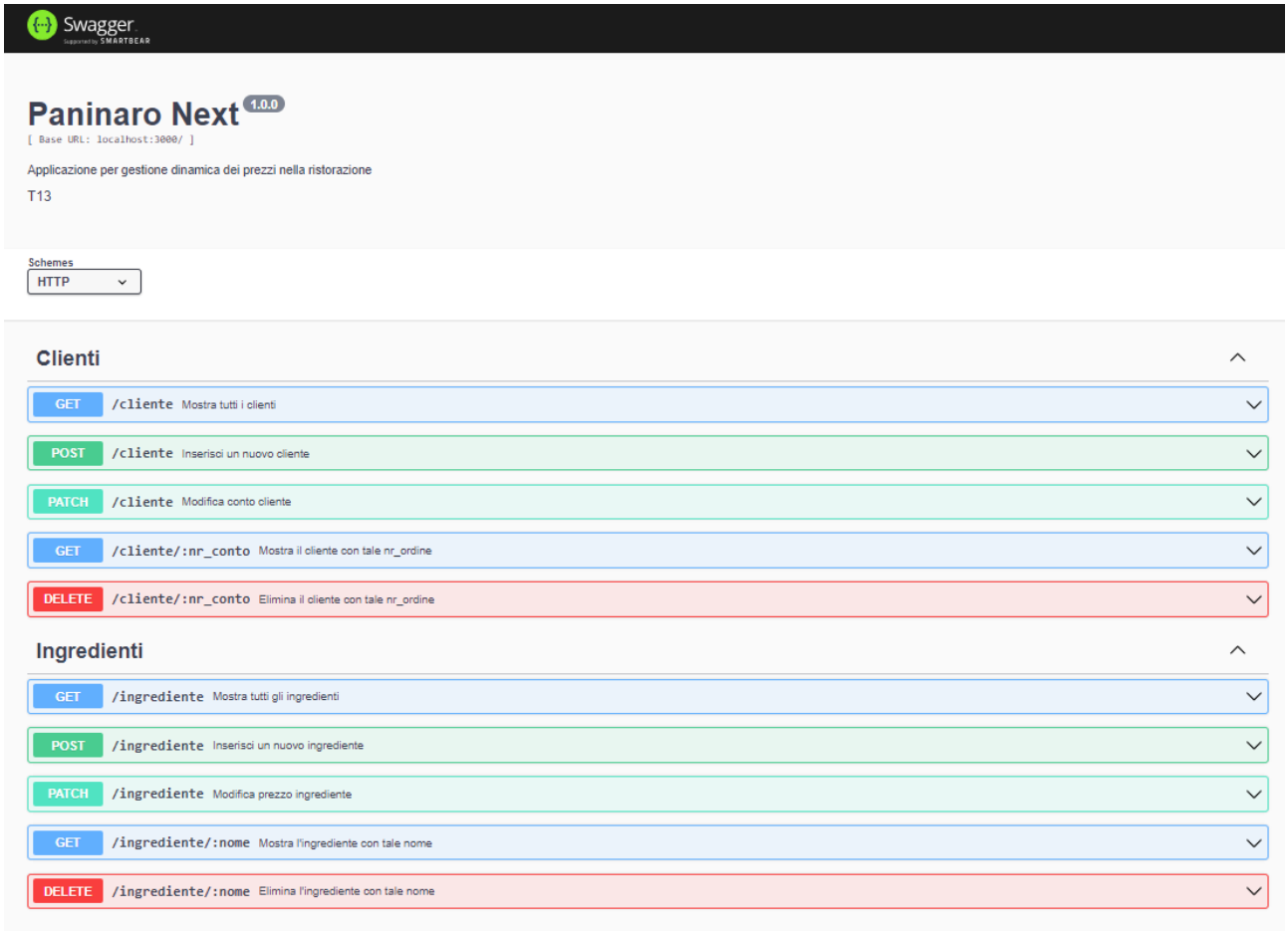
```
// DELETE '/ingrediente/nome'
const delete_ingrediente = (req, res) => {

    let input_nome = req.params.nome;
    var query = { nome: input_nome };

    Ingredienti.deleteOne(query, (err) => {
        if (err) {
            return res.json({ Error: err });;
        }
        else {
            res.status(200).json({ message: "DELETED 1 ingrediente" });
        }

    });

}
```

```
// DELETE '/cliente/nr_conto'
const delete_cliente = (req, res) => {

    let input_nr_conto = parseInt(req.params.nr_conto); //get the ingrediente name
    var query = { nr_conto: input_nr_conto };

    Clienti.deleteOne(query, (err) => {
        if (err) {
            return res.json({ Error: err });
        }
        else {
            res.status(200).json({ message: "DELETED 1 cliente" });
        }

    });

}
```

3. API documentation

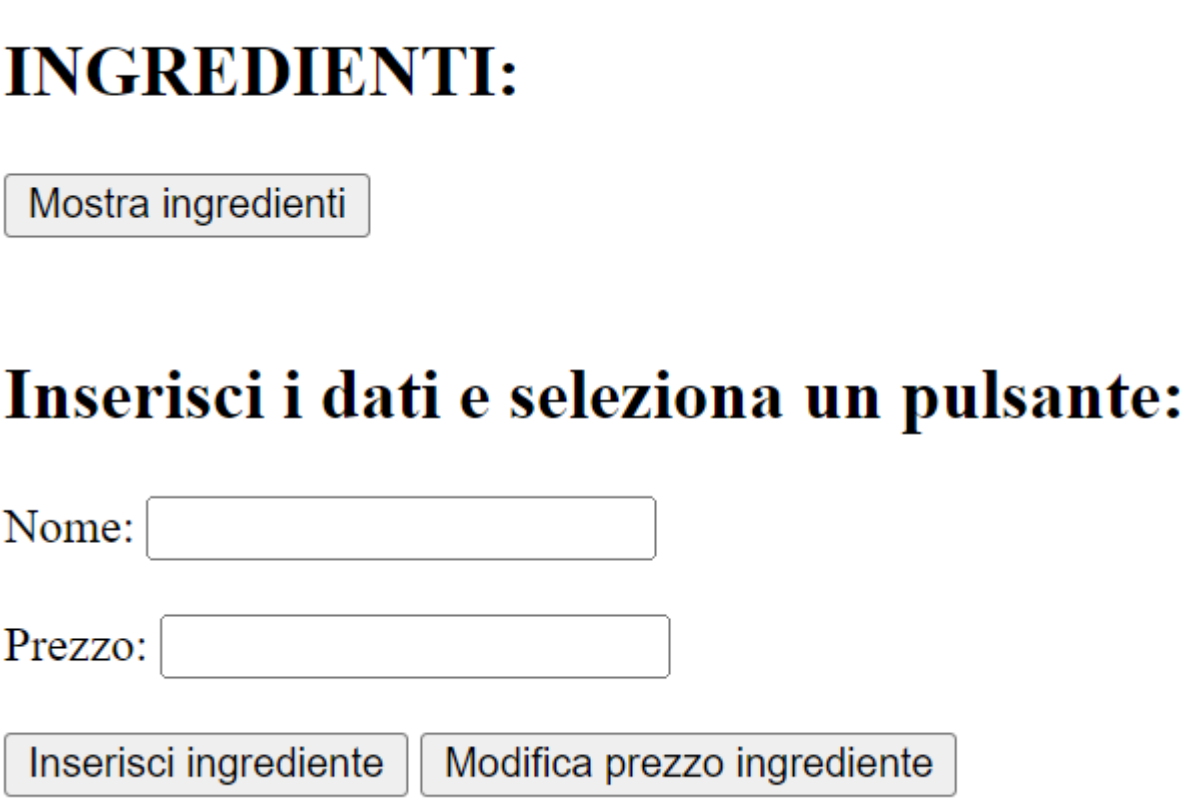
Le API Locali fornite dall'applicazione Paninaro Next e descritte nella sezione precedente sono state documentate utilizzando il modulo NodeJS chiamato Swagger UI Express. In questo modo la documentazione relativa alle API è direttamente disponibile a chiunque veda il codice sorgente.
Per poter generare l'endpoint dedicato alla presentazione delle API abbiamo utilizzato Swagger UI in quanto crea una pagina web dalle definizioni delle specifiche OpenAPI.
In particolare, di seguito mostriamo la pagina web relativa alla documentazione che presenta le 4 API (GET, POST, DELETE e PATCH) per la gestione dei dati della nostra applicazione.
La GET viene utilizzata per visualizzare i dati in una pagina HTML. La POST per inserire un nuovo dato nel nostro sistema. La DELETE per cancellare un dato dal nostro sistema. E la PATCH per modificare dati.
L'endpoint da invocare per raggiungere la seguente documentazione e':

http://localhost:3000/api-docs



4. FrontEnd Implementation

Il FrontEnd fornisce le funzionalità di login, e di visualizzazione, inserimento e modifica degli ingredienti. Un pulsante serve per mostrare gli ingredienti, mentre il form per modificare il prezzo e aggiungere gli ingredienti, usando gli appositi bottoni.



5. GitHub Repository and Deployment Info

Le due repository di cui tener conto sono:

- **Deliverables** https://github.com/gruppoT13/Deliverables dove sono presenti tutti i deliverables;
- **Codice del progetto** https://github.com/gruppoT13/paninaro_next dove è caricato l'intero progetto

6. Testing

Ecco i vari casi di test fatti:

Nr.Test Case	Descizione	Test Data	Precondizioni	Dipendenze	Risultato Atteso	Risultato riscontrato
1	Accesso area riservata **Gestore**	<admin> non vuota e <password> rispettosa delle politiche di sicurezza	<admin> e <passord> già definite in precedenza nel sistema e accesso non ancora effettuato		Accesso alla pagina **Gestore**	Messaggio "Accesso" e apertura in un'altra pagina di **page_gestore**
1.1	Tentativo di accesso con credenziali errate	<admin> sconosciuta <password> qualunque			Messaggio di errore <Unauthorized>	Messaggio di errore "Accesso Negato"
2	Effettuare un ordine selezionando uno o più prodotti, effettuando infine il pagamento	<data_ordinazione> di valore maggiore di <data_ordine>, <prodotti> non nullo		Questo test andrebbe effettuato con modulo PayPall	Ricevuta dell'ordine effettuato a cui viene assegnato un <nr_ordine> univoco	non implementato
2.1	Tentativo di registrazione di un ordine con pagamento rifiutato	<data_ordinazione> di valore maggiore di <data_ordine>, <prodotti> non nullo		Questo test andrebbe effettuato con modulo PayPall	Messaggio di errore <pagamento_rifiutato>	non implementato
3	Cambiare lo <status> dell'ordine	ordine <nr_ordine> già inserita nel sistema	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di conferma del cambio stato	non implementato
4	Modifica Ordine	ordine <nr_ordine> già inserita nel sistema	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di conferma della corretta modifica	non implementato
5	Modifica Prezzo Ingredienti	<ingrediente> già inserito nel sistema	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di conferma della corretta modifica, con la visualizzazione del nuovo <prezzo_ingrediente>	Messaggio di conferma "Ingrediente modificato"
5,1	Tentativo di Modifica Prezzo Ingredienti	<ingrediente> non conosciuto	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di errore <ingrediente_inesistente>	Messaggio di conferma "Ingrediente modificato"
6	Aggiungere ingrediente	<ingrediente> non vuoto o <prezzo> non vuoto	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di conferma della corretta modifica	Messaggio di errore "Ingrediente non esiste"
6,1	Tentativo di aggiungere ingrediente rifiutato	<ingrediente> vuoto o <prezzo> vuoto	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di errore <dati_insufficenti>	Messaggio di errore "Valore nullo"
7	Modifica Prodotti	<prodotto> già inserito nel sistema	<permessi> = TRUE	Questo caso di test deve essere fatto dopo il caso di test nr 1	Messaggio di conferma della corretta modifica, con la visualizzazione del nuovo <prezzo_prodotto>	non implementato