Web Applications 2023/2024

PL8S

by WA001

- Antonutti Manuel
- Carlesso Daniel
- Frigione Luigi
- Shams Mahshid
- Ursino Nicola

UNIVERSITÀ DEGLI STUDI DI PADOVA

DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

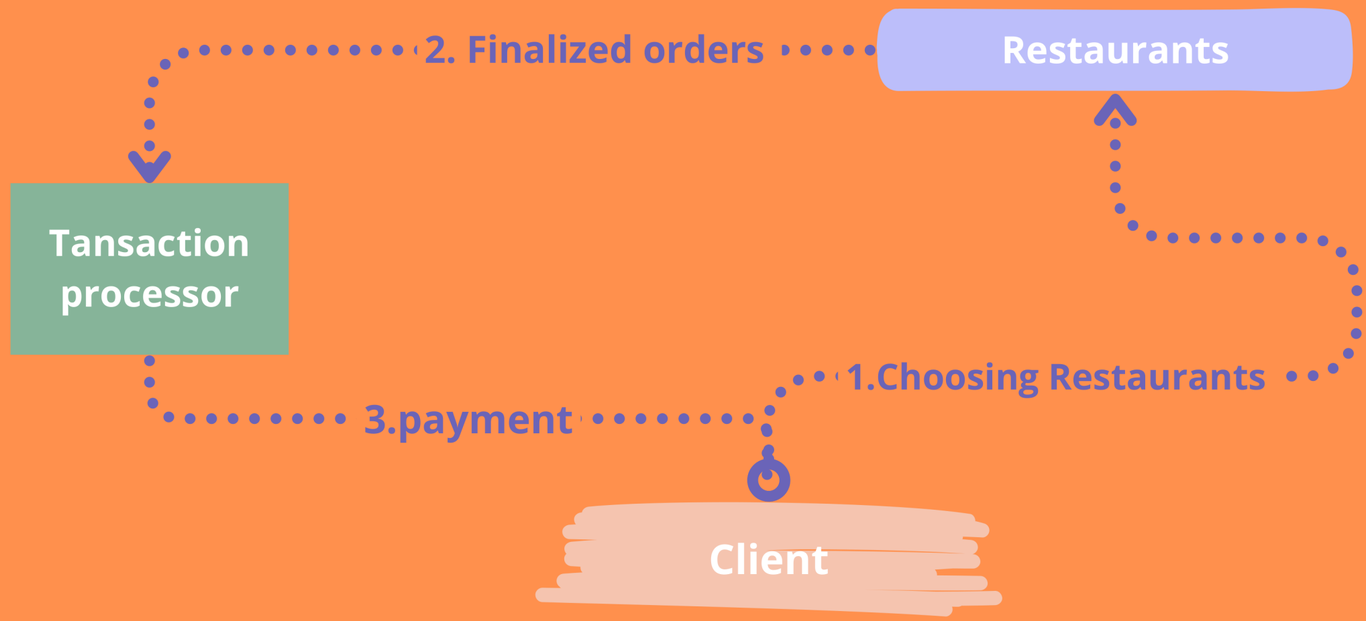
Application Objective

The proposed web application aims to revolutionize the dining experience by providing a centralized system for managing extensive records of restaurants, managers, dishes, and orders.

- real-time menu changes
- precise order monitoring
- Queue Avoidance

Objective
Advantages

This system not only streamlines restaurant operations but also elevates the overall customer experience by leveraging technology and data management.



- Operational Transparency
- Enhanced User Convenience
- Improved Service Quality
- Data-Driven Insights
- Increased Business Efficiency

User Roles

Guest (unregistered)

- Browse restaurants and menus
- Create a profile (register)

Customer

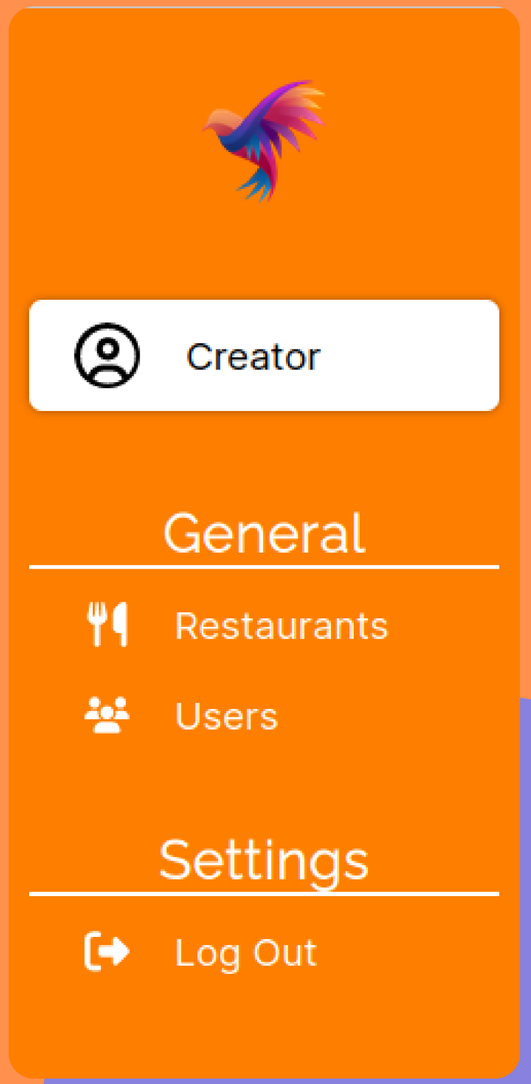
- Add and remove dishes to/from the cart
- Place an order
- Update profile

Manager

- Add, edit and delete restaurants
- Add, edit and delete dishes
- Update profile

Admin

- Update profile
- Add and delete users
- Edit and delete restaurants



Resource Filters

- **Auth**

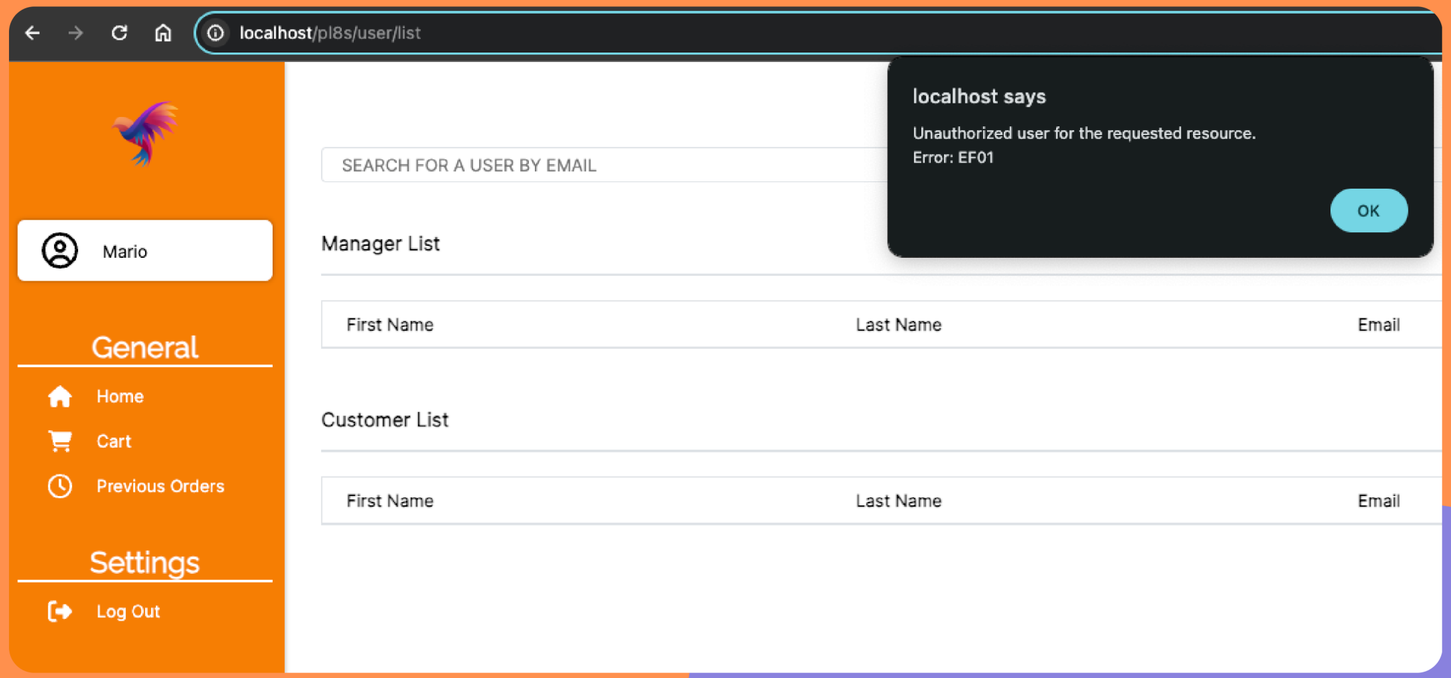
  Filters allow to authenticate a user and eventually authorize it for accessing the requested resource.

- **Security**

  Filters protect important resources from being accessed by everyone.

- **Our implementation**

  Admin, Manager, User, Restaurant, Dish, ListRestaurant, Stripe



JSON Web Token

In its compact form, a JSON Web Token consists of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

Encoded using the HMAC SHA256 algorithm.

Stored client-side and sent in the *Authorization* header with *Bearer* schema.

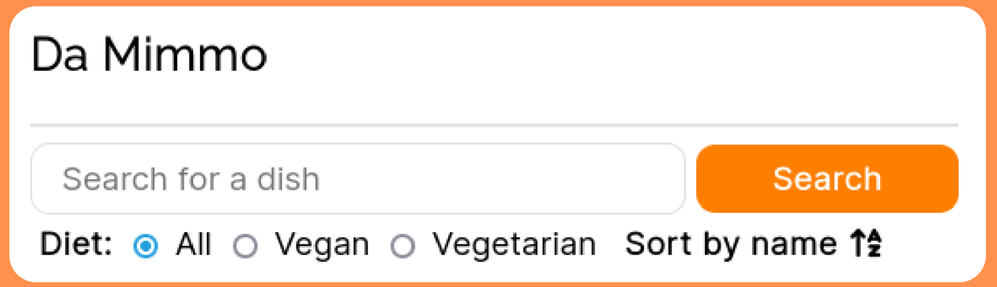
About JWT

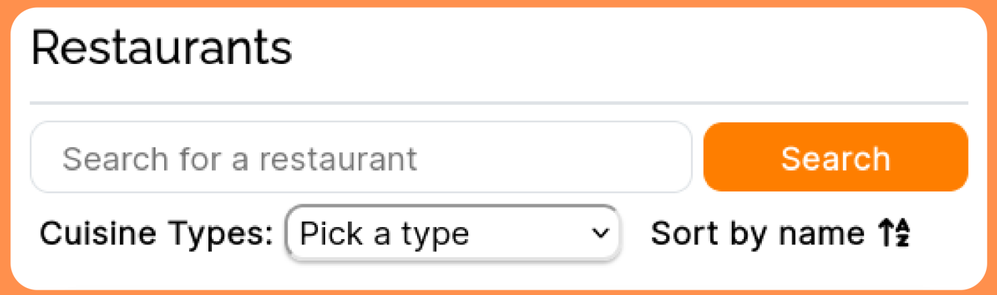
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

```
String jwt = Jwts.builder() JwtBuilder
        .header() BuilderHeader
            .add( k: "app",  v: "PL8S")
            .and() JwtBuilder
        .claim( s: "uid", user.getUser_id())
        .claim( s: "rol", user.getRole())
        .claim( s: "str", user.getStripe_id())
        .claim( s: "dat", exp_date)
        .signWith(key)
        .compact();

return jwt;
```

Search with filters





Name field

- can be blank or partial
- case insenstive

Diet

- **vegan** (only vegan ingredients)
- **vegetarian** (vegan or vegetarian ingredients)

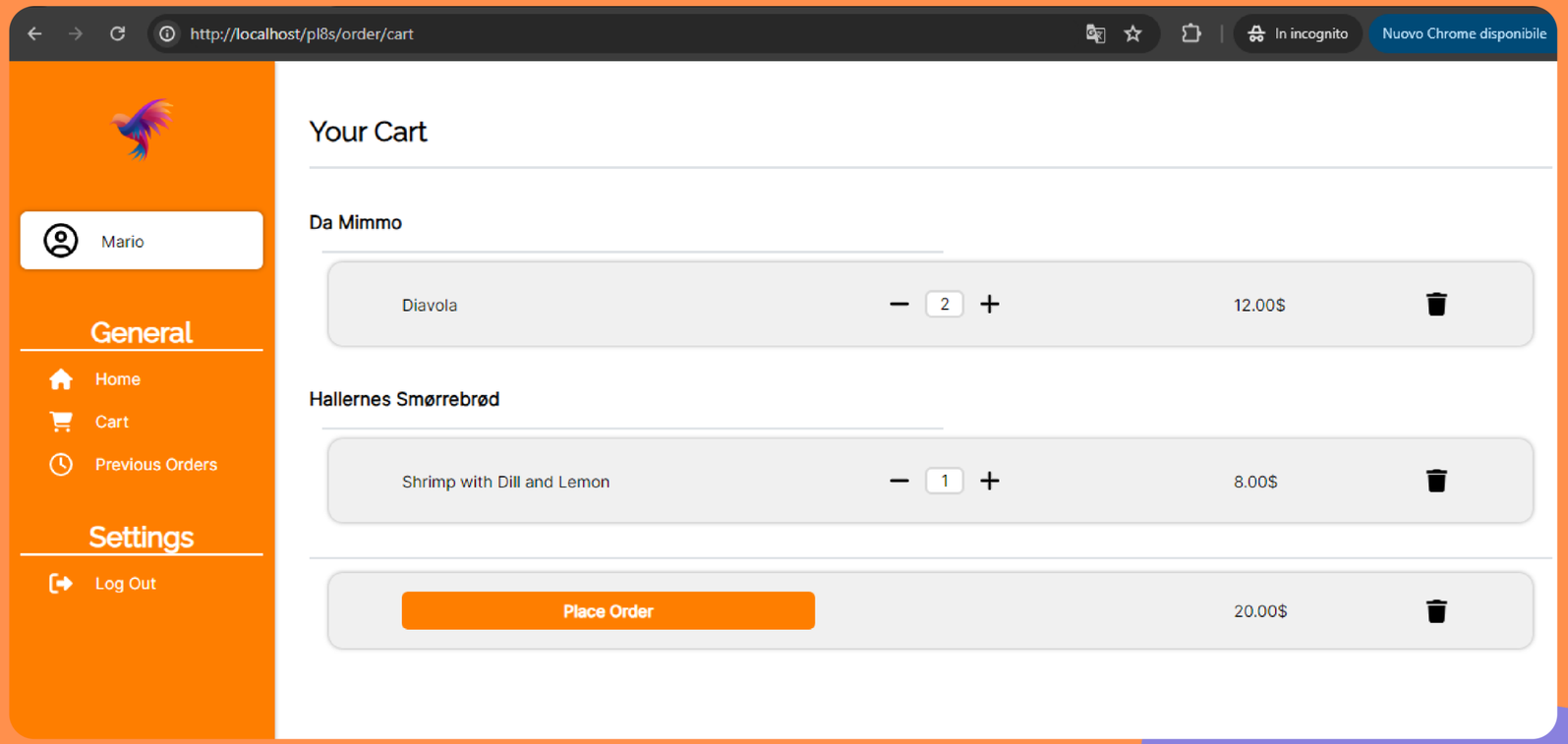
Cuisine types

- the select box (without duplicates) is automatically filled

Sort by name

- Sorting is handled at client side, no http requests required

Cart Page



- Change quantities
- Delete items
- Place order

- **Persistent cart**

The cart is designed to save each change to its state in the database, ensuring persistence even after the authentication token expires.

stripe

Main Features

Platform

A **fully integrated** suite of financial and payments products

Now

Connect

- Expanded features for more businesses
- **Facilitate** and **monetise**
- Send payouts around the world

Future

Flexibility

- Adapt to **customer preferences** with access to 100+ payment methods
- Charge customers in more than 135 currencies

Fact

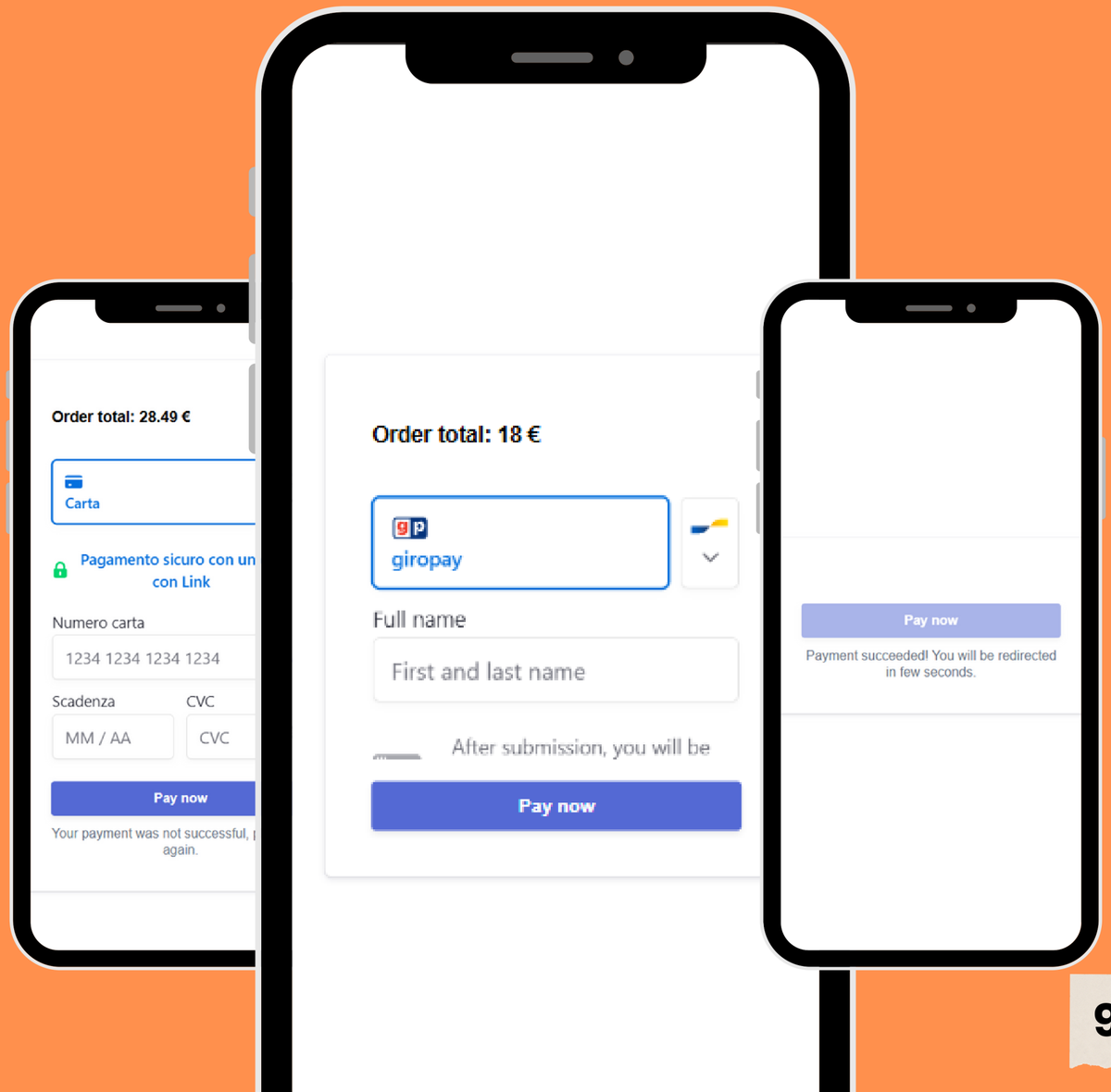
stripe

How to Buy

1. Visit our website.

2. Choose your meal.

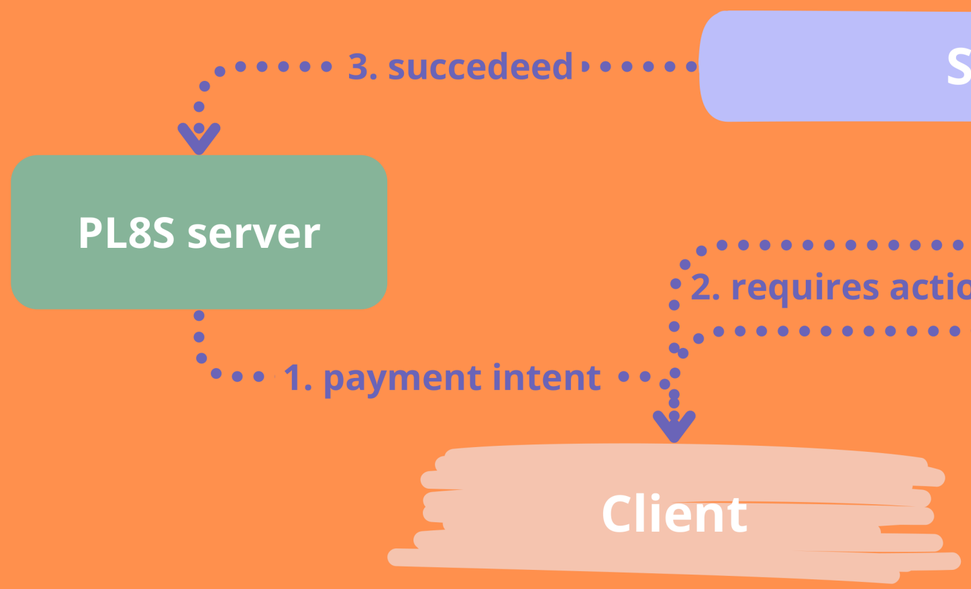
3. Enter payment details.

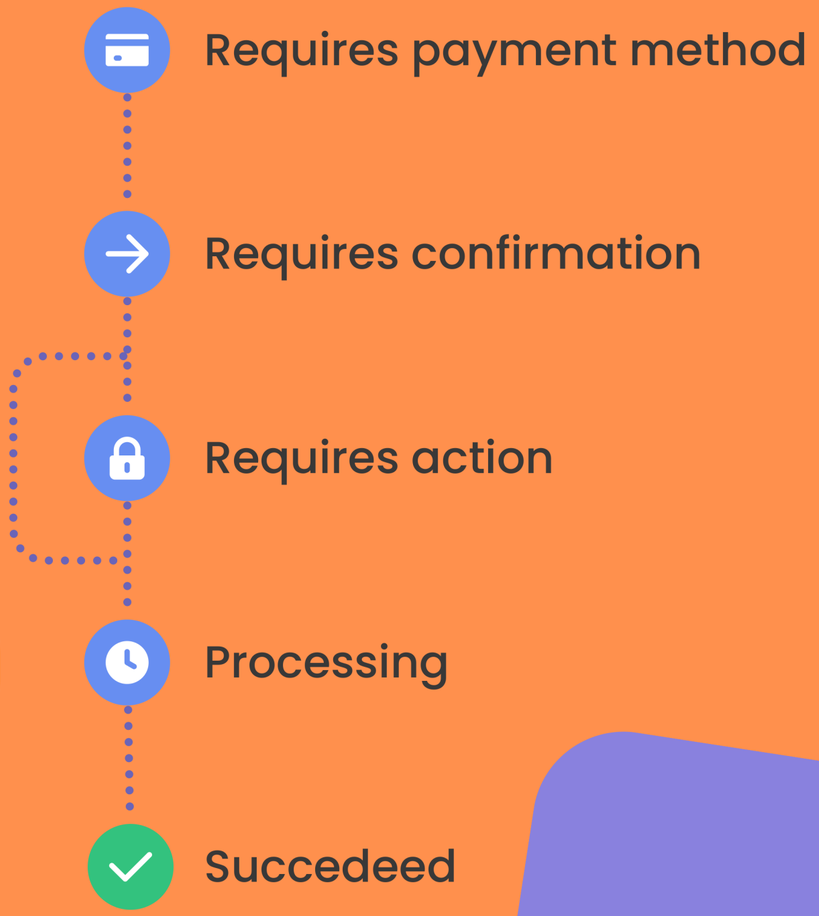


Stripe Filter

- Auth payment confirmation

The payment confirmation is a server-to-server communication.





REST API

URI	Method	Filter
/restaurants/name/{?name}/ cuisine_types/{?cuisine_type}	GET	None
/restaurant/cuisine-types	GET	None
/restaurant/create	POST	MRF
/restaurant/{?restaurant_id}	GET	None
/restaurant/manager	GET	LRMRF
/restaurant/update/{?restaurant_id}	PUT	RRF
/restaurant/delete/{?restaurant_id}	DELETE	RRF
/order	GET	URF
/order	PUT	URF
/order/dishes	GET	URF
/order/dishes	DELETE	URF
/order/previous	GET	URF
/order/payment	POST	SRF
/order/dishes/{?dish_id}	POST	URF
/order/dishes/{?dish_id}	PUT	URF
/order/dishes/{?dish_id}	DELETE	URF
/order/previous-orders/{?order_id}/ dishes	GET	URF

URI	Method	Filter
/dish	POST	DRF
/dish/{?dish_id}	PUT	DRF
/dish/{?dish_id}	DELETE	DRF
/dish/{?dish_id}/orders	GET	DRF
/dishes	GET	None
/user	PUT	URF
/user/register	POST	None
/user/create	POST	ARF
/user/login	GET	None
/user/logout	GET	URF
/user/cards	GET	URF
/user/list	GET	ARF
/user/email/{?user_email}	GET	ARF
/user/delete/{?user_id}	DELETE	ARF
/cuisine/types	GET	None

RESOURCE FILTERS

- Admin (ARF)
- User (URF)
- Dish (DRF)
- Manager (MRF)
- Restaurant (RRF)
- Stripe (SRF)
- ListRestaurantManager (LRMRF)

We also used the following servlets:

- RestDispatcherServlet: delegating the requests to the appropriate REST resource
- SelectDishServlet: selecting a dish from the database

LIVE DEMO