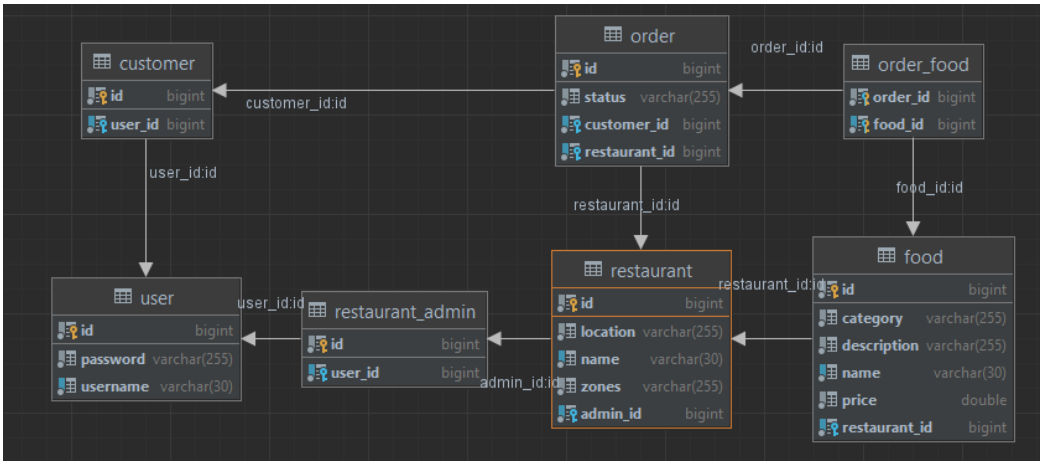
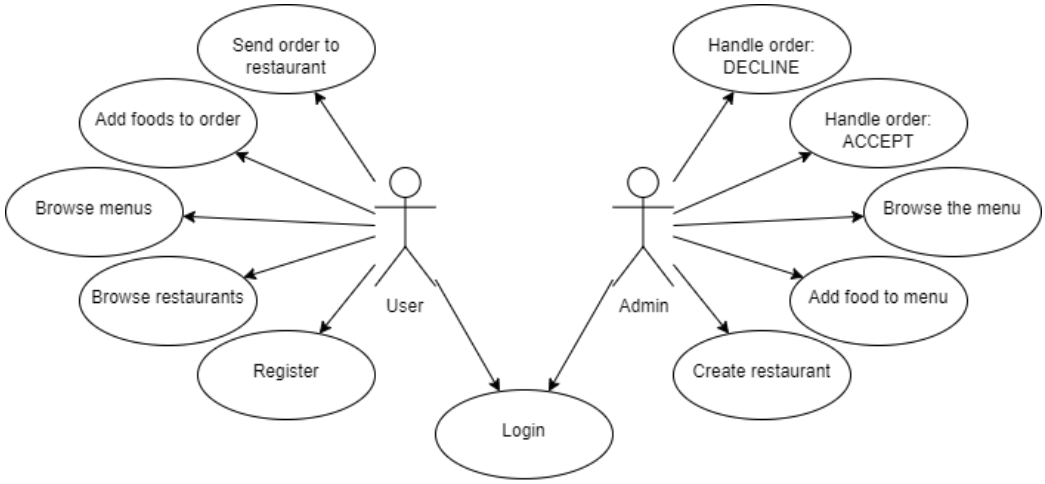
# Assignment 2 Documentation

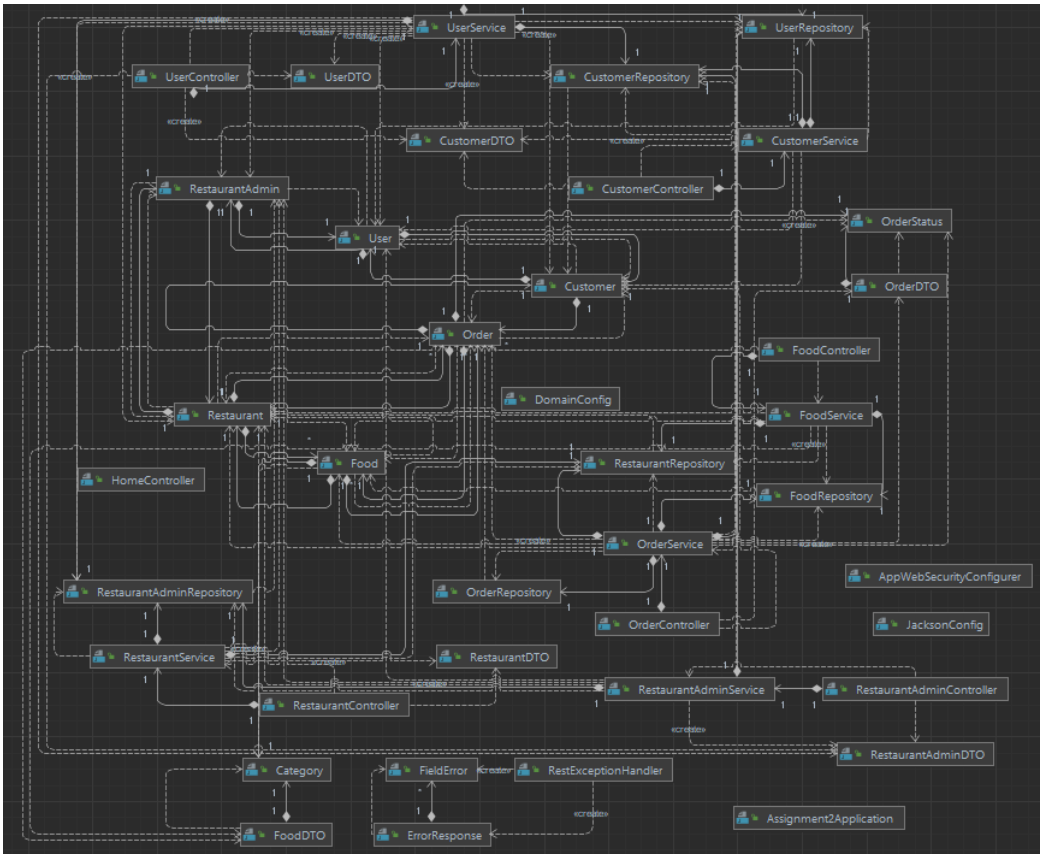
# Database diagram



# Use case diagram



# Class diagram



# Design patterns

```java
public FoodDTO create(final FoodDTO foodDTO) { // builder pattern
    final Food food = Food.builder()
            .name(foodDTO.getName())
            .description(foodDTO.getDescription())
            .price(foodDTO.getPrice())
            .category(foodDTO.getCategory())
            .restaurant(restaurantRepository.getById(foodDTO.getRestaurant()))
            .build();
    mapToEntity(foodDTO, food);
    foodDTO.setId(foodRepository.save(food).getId());
    return foodDTO;
}
```

```java
public void update(final Long id, final int advance) { //State DP
    final Order order = orderRepository.findById(id)
            .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND));
    switch (order.getStatus()) {
        case PENDING -> {
            if (advance != 0) order.setStatus(OrderStatus.ACCEPTED);
            else order.setStatus(OrderStatus.DECLINED);
        }
        case ACCEPTED -> order.setStatus(OrderStatus.IN_DELIVERY);
        case IN_DELIVERY -> order.setStatus(OrderStatus.DELIVERED);
    }
    orderRepository.save(order);
}
```