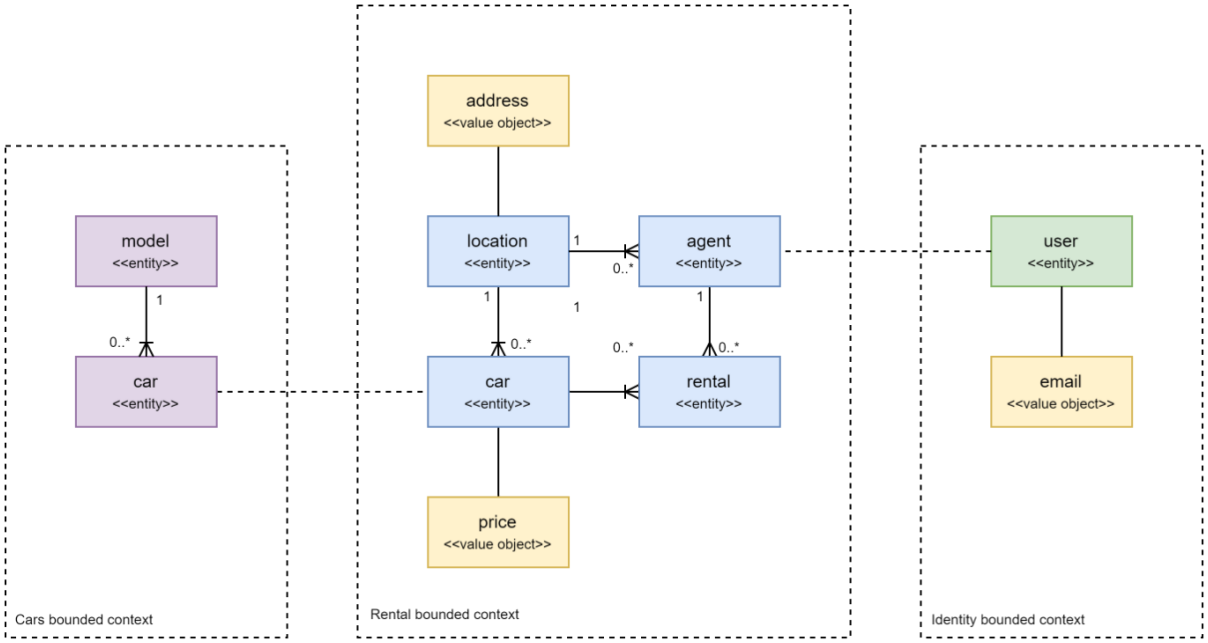
The lotters assignment

Domain

I identified 3 bounded context, each one having the specific entities and value objects. Between the Cars bounded context, bounded context that contains all the information regarding the presentation of a car, and the rentals bounded context, we have a common concept, that will share the same identity, but with totally different properties. The creation of a car is the responsibility of the cars bounded context, and the rental bounded context will just receive a notification that a new car was created or deleted, and from there is the rentals bounded context responsibility to ensure that the car is created.

In the same way a common concept between the rentals bounded context and identity bounded context is the user. The responsibility to create a user is for identity, and rentals bounded context will create based on a notification an agent (user in rentals).

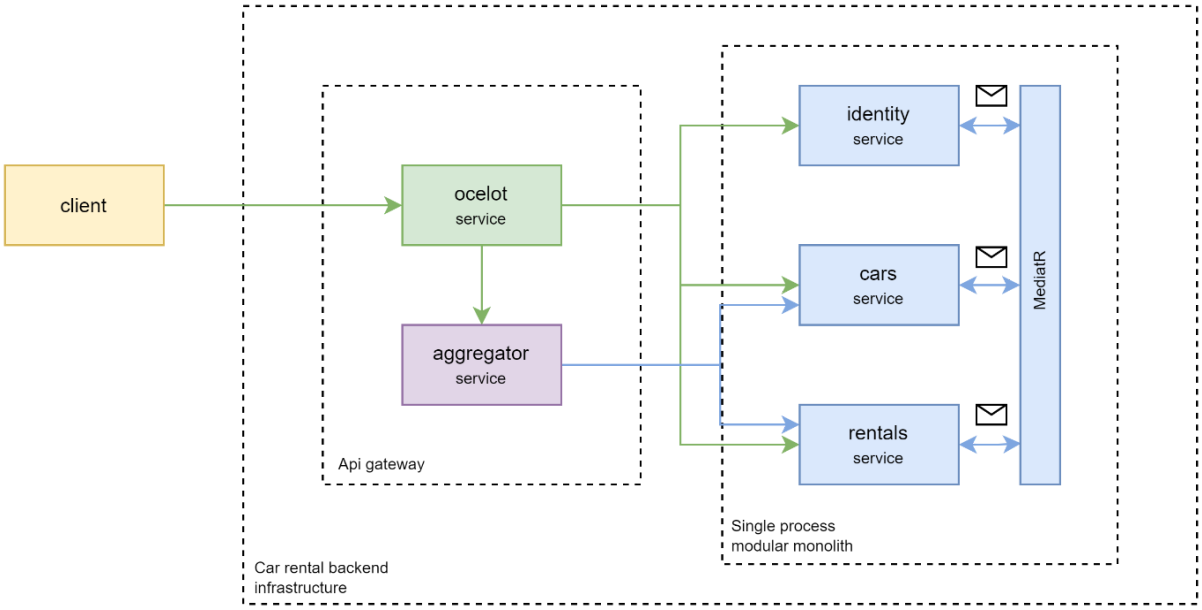


Architecture

For the architecture of the system I choose a modular monolith, with multiple modules for each bounded context to keep it simple, and an API gateway to expose the API to the world.

The communication between different modules I used MediatR, to follow the query, command, query pattern, and to be easy to decouple in different services, by switching from MediatR to a service bus as RabbitMq or Kafka.

The API gateway is composed by two services, due to the limitation of a service that used ocelot, to not have any other controller. The service that contains the ocelot file configurations is just a revers proxy to redirect the calls from the client to the specific service or the aggregator. The aggregator is used to compose complex responses with data from multiple services, and in this way to not let the user to handle all the round trips to get all the necessary data.


client
ocelot
service
aggregator
service
Api gateway
identity
service
cars
service
rentals
service
MediatR
Single process
modular monolith
Car rental backend
infrastructure