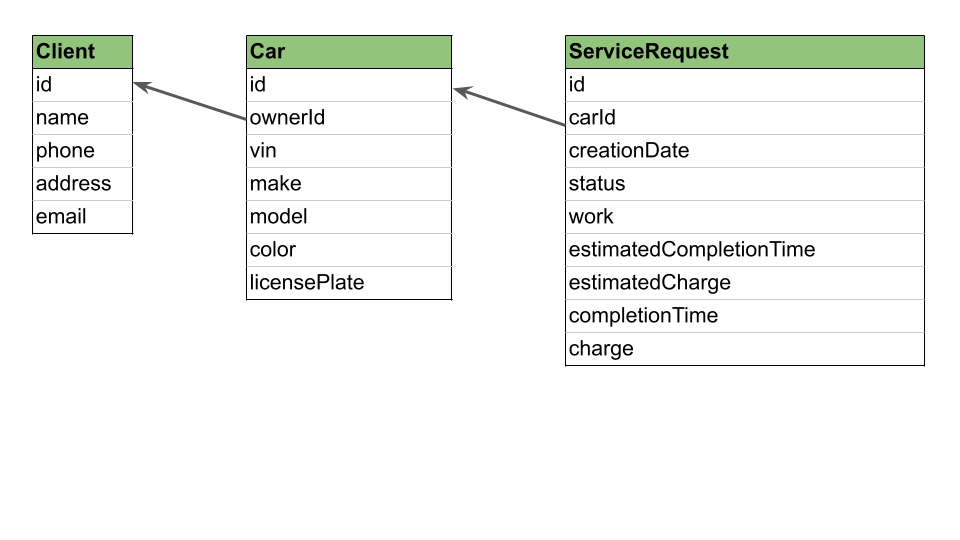
**Auto Repair Shop APIs - Design and Assumptions**

The scope of this system is to define basic CRUD operations for an Auto shop, where **Service Requests** need to be managed for **Clients** and their **Cars.** A client in the system can have multiple cars, and a car can have multiple service requests associated with it. The client that owns a car is referred to as the owner in Car object and referenced by ownerId in the car table.

This is a simplistic approach to solve this use case with some assumptions to mimic the real world solution at a smaller scale. For the scope of this assignment we assume 3 primary entities - Service Requests, Cars and Clients. Here’s the data model diagram -

Assumptions

1. The system is defined such that managing the service requests is the primary objective, however there are endpoints to access Clients and Cars as well.
2. When creating (or updating) a car the API expects a valid ownerId (existent) and when creating (or updating) a service request the API expects a valid carId (existent). This is done in this way, since when coupled with UI, that’s how the system will handle a service request creation.(i.e. Creating or looking up a client account first, then creating or looking up a car and then creating a service request).
3. For the purposes of the assignment, delete client operation will cascade delete car and service request associated with the client and delete car operation will cascade delete the service request associated with the car. In the real world, we don’t want to delete these objects. Ideally, we would want to set these objects as inactive by an update operation. However, the delete API is provided to simply offer the functionality.
4. The system has defined pagination only for Clients, this is just to showcase the pagination functionality. The page number is defined with zero based indexing, so the first records are retrieved with page = 0. It can be easily extended to other classes.