I'll provide a high-level overview of how the Trimble Cars application could be structured, focusing on the backend API using Java, Spring Boot.

Database Design

database schema

- Car: Represents a car registered by the car owner.
- User: Represents both car owners and end customers.
- Lease: Represents a lease agreement between a car owner and an end customer.

API Endpoint

- POST /cars - Register a new car (Car Owner)
- GET /cars - View available cars for lease (End Customer)
- POST /leases- Start a new lease (End Customer)
- PUT /leases/{leaseId} - End a lease (End Customer)
- GET /leases/history- View lease history (Car Owner and End Customer)

Example Code

Car entity

```
@Entity
public class Car {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String make;
    private String model;
    private String status; // Ideal, OnLease, OnService

    @OneToMany(mappedBy = "car")
    private List<Lease> leases;

    // Getters and setters
}
```

And here's an example of how the Lease service could be implemented:

```
@Service
public class LeaseService {

    @Autowired
    private LeaseRepository leaseRepository;

    public Lease startLease(Car car, User customer) {
        Lease lease = new Lease(car, customer);
        return leaseRepository.save(lease);
    }

    public void endLease(Long leaseId) {
        Lease lease = leaseRepository.findById(leaseId).orElseThrow();
        lease.setEndDate(LocalDateTime.now());
        leaseRepository.save(lease);
    }
}
```

Testing

Unit tests would be written using JUnit to test the service layer. Functional testing of the APIs could be automated using Postman or a similar tool.

Logging

Proper log statements would be added throughout the application to track important events, such as lease starts and ends.

Optional Steps

Integrating with a 3rd party authentication service, such as Okta, could be done using Spring Security. Integrating with a portal like Auto Car or Overdrive would require API calls to fetch car models and variants.