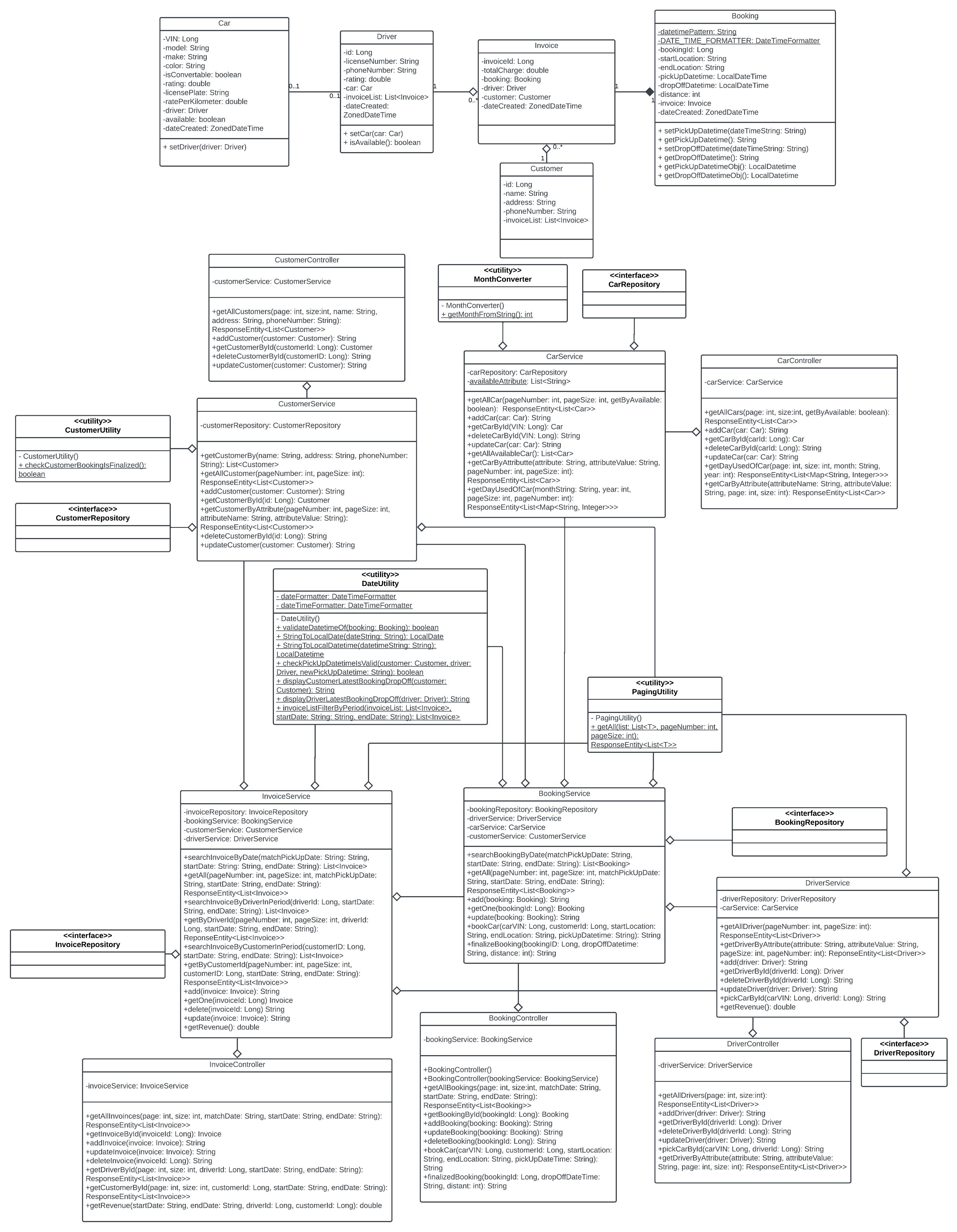
****

**IMPLEMENTATION RESULT:**

We established multiple APIs to conduct Create – Read – Update – Delete (CRUD) operations on all the system’s required entities: Car, Booking, Customer, Driver, and Invoice, to meet the business requirements. By applying PagingAndSortingRepository to our repositories, we implement pagination on all data returned from APIs so that it can be presented to users in pages for Read operations on each entity. Furthermore, we implement search APIs on each entity to search the data based on specific attributes, as well as statistical APIs to access each entity’s statistical data.

The APIs for admin will have an endpoint called “admin”, while the APIs for customer do not include it. The description of admin and customer’s APIs can be briefly explained as below (the detailed description of the APIs is included in README.md file):

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Entity** | **Method** | **Usage** |
| Admin | Booking | GET | Get all bookings filtered by a specific date or by a period. The data returned will be paginated. |
| POST | Create a new booking or finalize an existing booking. |
| PUT | Update booking |
| DELETE | Delete an existing booking |
| Invoice | GET | Get all invoices filtered by a specific date or by a period. The admin can also get the revenue of all invoices by a period, by a customer or driver. The data returned will be paginated. |
| POST | Create a new invoice |
| PUT | Update invoice |
| DELETE | Delete an existing invoice |
| Car | GET | Get the number of days a car was used within a month. The data returned will be paginated. |
| POST | Create a new car |
| PUT | Update a car |
| DELETE | Delete an existing car |
| Driver | GET | Get all drivers, get driver by ID or by attributes. The data returned will be paginated. |
| POST | Create a new driver |
| PUT | Update a driver |
| DELETE | Delete an existing driver |
| Customer | GET | Get all customer, get customer by ID or by attributes. The data returned will be paginated. |
| POST | Create a new customer |
| PUT | Update a customer |
| DELETE | Delete an existing customer |
| Customer user | Booking | GET | Get a booking by ID |
| POST | Book a car |
| Car | GET | Get all cars, get car by ID, by availability or by attributes. The data returned will be paginated. |
| Driver | Driver | GET | Pick a car for a driver |

To implement the pagination for all entities, we implemented our services to return an object of ResponseEntity. This object will take the input parameters including page number and size from HTTP requests and perform pagination on the data returned from the database. The data returned will be paginated based on the page number and size that users have input in the HTTP request parameters.

For the implementation of search APIs, we retrieve all the data of an entity from the database and filter them by the attributes and conditions that users have specified in the HTTP request parameters. The data returned will also be paginated by returning a ResponseEntity object with the page number and size included in HTTP request parameters.

In the implementation of statistical APIs for invoices, we obtain a list of invoices by a period, by a certain customer or driver, then compute the total of all their total charges to produce a revenue that is required by the admin. However, to use the statistical API for cars, we first establish a map with the key being a car’s license plate and the value being the number of days in a month that the car was driven. We iterate through all bookings created in a certain month, as provided by the admin in the HTTP request parameter, to determine how many days a car was driven during that month. The returned data is also paginated with ResponseEntity object.

**LIMITATION AND KNOWN BUGS:**

1. No validation of the date time format in request body:

Although there is the validation process of the date time format in HTTP request parameters, the date time in request body of HTTP request is not validated but return a “Bad Request” status to the client side. This is due to not implementing a custom handler exception for the controllers to provide more meaningful and insightful warnings. However, we can resolve this problem in the future by creating an exception handler to interpret the invalid input more thoroughly to the client side.

1. Implementing LocalDatetime instead of ZoneDatetime:

Because we apply LocalDatetime for pick – up and drop – off date time attributes of the Booking entity, our system may have some issues if it is used in different time zones in the world. We can repair this issue by applying ZoneDatetime to the attributes of date time instead so the date time can be determined more accurately and consistently according to different time zones in the world.