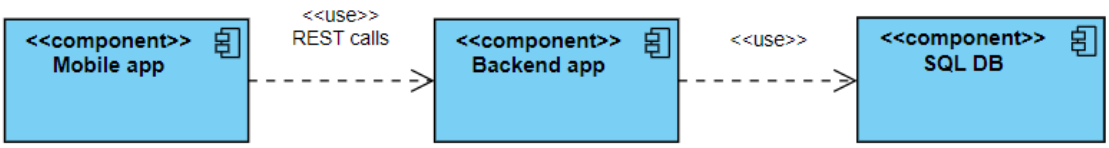
Ridesharing App

The development team of company X has received a new project. They must implement a ridesharing system, so the end users can book a trip using someone's shared car. The end users will interact with the system via a mobile app. The mobile app will communicate with a back-end app using REST. The data will be stored in an SQL database.



The first requirements for the back-end app are to provide endpoints that can facilitate:

- Adding a driver
- Updating a driver
- Getting the list of drivers by name or city

The following business rules need to be in place for each endpoint:

- Adding a driver: the request body and response body are mapped according to the table below:

Field name	**Present on the request**	**Mandatory on request**	**Present on response**	**Other validations**
id	no	no	yes	PK, auto-generated on DB level
name	yes	yes	yes	String with max length of 100
email	yes	yes	yes	String with max length of 100 Unique (you cannot add 2 drivers with the same email)
city	yes	yes	yes	String with max length of 100

- Updating a driver: driverId is in the path, the request body and response body are mapped according to the table below:

Field name	**Present on the request**	**Mandatory on request**	**Present on response**	**Other validations**
id	yes	yes	yes	Cannot be updated
name	yes	yes	yes	String with max length of 100
email	yes	yes	yes	String with max length of 100 Unique (you cannot add 2 drivers with the same email)
city	yes	yes	yes	String with max length of 100

- Getting the list of drivers by name or city: the name and / or city is provided in the url of the request. The response should contain all details for each driver.

Unit tests should be implemented for the service methods, with a coverage of at least 70% per line.

The type of the SQL DB is chosen by the development team.