## ASSUMPTIONS

1. For Task 2 - It is kept mandatory to provide both a name and a phone number when creating a new person. The absence of these fields renders the data incomplete and not useful for identification or contact purposes. If either the name or phone number is missing in the request, the service will return an error message.

2. The request body provided for Task 2 does not have an age column, if it is not included, the code will insert a default value of 0 for age into the database. This design allows flexibility for the client to provide age when available but ensures that the person record can still be created without it.

3. The approach of allowing each service function independently opens a connection to the database when needed.

## APPROACH

### Task 1: Retrieve Person Details

1. **Receive Request:** A GET request is made to GET/person/:personID/info.
2. **Extract ID:** The personID is extracted from the URL.
3. **Fetch Details:** A SQL query is executed to retrieve the person's details from the database using the personID.
4. **Return Response:** The details are returned as a JSON response, or an error is sent if the person is not found.

### Task 2: Create a New Person

1. **Receive Request:** A POST request is made to POST /person/create with person details in JSON format.
2. **Validate Input:** The incoming data is validated for required fields.
3. **Insert Data:** SQL queries are executed to insert the new person into the person table, along with their phone and address details.
4. **Return Response:** A success message is returned as a JSON response, or an error is sent if any step fails.

## DEPENDENCIES

1) Gin Framework:
   - Package: **github.com/gin-gonic/gin**
2) MySQL Driver:
   - Package: **github.com/go-sql-driver/mysql**
3) Dotenv Package:
   - Package: **github.com/joho/godotenv**