**URL Shortener Project Documentation**

**Overview**

The URL Shortener project provides a service to shorten long URLs and store them in a PostgreSQL database. It also allows users to retrieve the original URL using the shortened version, delete URLs, and view the details of a URL by its shortened version.

**Project Structure**

1. **dbconnection Package**: Handles database connection.
2. **models Package**: Contains data structures and functions for interacting with the database.
3. **shortner Package**: Includes logic for generating shortened URLs.
4. **controllers Package**: Manages HTTP request handling and business logic.
5. **routes Package**: Defines the API routes.
6. **main.go**: Entry point of the application.

**Functionality**

**1. Database Connection (dbconnection Package)**

* **Connect()**: Establishes a connection to the PostgreSQL database.

**Steps**:

* + Constructs a connection string.
  + Opens a connection using sql.Open.
  + Checks the connection with DB.Ping.
  + Logs errors if any occur.

**2. Models (models Package)**

* **UrlData Struct**: Represents a URL record with fields for ID, original URL, shortened URL, and creation timestamp.
* **CreateURL(u UrlData) (int, error)**: Inserts a new URL record into the database.

**Steps**:

* + Defines an SQL INSERT query with placeholders.
  + Executes the query with QueryRow and Scan to get the newly created record's ID.
  + Returns the ID and any errors encountered.
* **GetURLByID(id int) (UrlData, error)**: Retrieves a URL record by its ID.

**Steps**:

* + Defines an SQL SELECT query with a placeholder for ID.
  + Executes the query with QueryRow and Scan to retrieve the record.
  + Returns the UrlData and any errors encountered.
* **DeleteURL(id int) error**: Deletes a URL record by its ID.

**Steps**:

* + Defines an SQL DELETE query with a placeholder for ID.
  + Executes the query with Exec.
  + Returns any errors encountered.
* **GetURLByShortURL(shortUrl string) (UrlData, error)**: Retrieves a URL record by its shortened URL.

**Steps**:

* + Defines an SQL SELECT query with a placeholder for the shortened URL.
  + Executes the query with QueryRow and Scan to retrieve the record.
  + Returns the UrlData and any errors encountered.

**3. URL Shortening Logic (shortner Package)**

* **GenerateShortURL(urlToBeShort models.UrlData) string**: Creates a shortened version of the original URL.

**Steps**:

* + Uses MD5 hashing to generate a hash from the original URL.
  + Encodes the hash in hexadecimal format.
  + Returns the first 8 characters of the encoded string as the shortened URL.

**4. Controllers (controllers Package)**

* **CreateShortURL(originalUrl string) string**: Creates and saves a shortened URL.

**Steps**:

* + Creates a UrlData object with the original URL, generated shortened URL, and current timestamp.
  + Calls models.CreateURL to save the URL to the database.
  + Returns the shortened URL or an empty string if an error occurs.
* **ShortenURLHandler(w http.ResponseWriter, r \*http.Request)**: Handles HTTP requests to shorten URLs.

**Steps**:

* + Parses the JSON request body to get the original URL.
  + Calls CreateShortURL to generate and save the shortened URL.
  + Responds with the shortened URL or an error message.
* **RedirectToOriginalUrl(w http.ResponseWriter, r \*http.Request)**: Redirects to the original URL based on the shortened URL.

**Steps**:

* + Extracts the shortened URL from the request path.
  + Calls models.GetURLByShortURL to retrieve the original URL.
  + Redirects the user to the original URL or responds with an error if the URL is not found.
* **DeleteURLHandler(w http.ResponseWriter, r \*http.Request)**: Handles URL deletion requests.

**Steps**:

* + Extracts the shortened URL from the request path.
  + Calls models.GetURLByShortURL to retrieve the URL record.
  + Calls models.DeleteURL to delete the record from the database.
  + Responds with success or error status.
* **GetURLHandler(w http.ResponseWriter, r \*http.Request)**: Retrieves URL details by its shortened version.

**Steps**:

* + Extracts the shortened URL from the request path.
  + Calls models.GetURLByShortURL to get URL details.
  + Responds with the URL data or an error message if not found.

**5. Routes (routes Package)**

* **Router() \*mux.Router**: Sets up the API routes using Gorilla Mux.

**Routes**:

* + POST /shorten: Creates a shortened URL.
  + GET /redirect/{shortUrl}: Redirects to the original URL.
  + DELETE /delete/{shortUrl}: Deletes a URL by its shortened version.
  + GET /url/{shortUrl}: Retrieves URL details by its shortened version.

**6. Main (main.go)**

* **main()**: Entry point of the application.

**Steps**:

* + Prints a welcome message.
  + Calls dbconnection.Connect() to establish a database connection.
  + Starts the HTTP server on port 8080 with routes.Router().

**Running the Application**

1. **Install Dependencies**: Ensure all required Go packages are installed, including github.com/lib/pq, github.com/gorilla/mux, etc.
2. **Setup Database**: Create the PostgreSQL database and urldata table if not already created.
3. **Start Application**: Run go run main.go to start the server.
4. **Test Endpoints**: Use tools like Postman or curl to test the API endpoints.