MSc_group_c

ITU MiniTwit Report Skeleton

May 2024

1 Introduction

2 System perspective

This section presents the system.

2.1 Design and architecture

The system is refactored with the Go programming language.

2.1.1 Module diagram

An overview of the modules of the codebase is presented by the following package diagram. The package diagram models the internal structure of the codebase from the src folder (not infrastructure). Though it should be noted that within the handlers folder, the classes auth.go, message.go and user.go are presented, and its dependencies. This is to depict the complexity of that modules (since it is the biggest and most central module in the system).

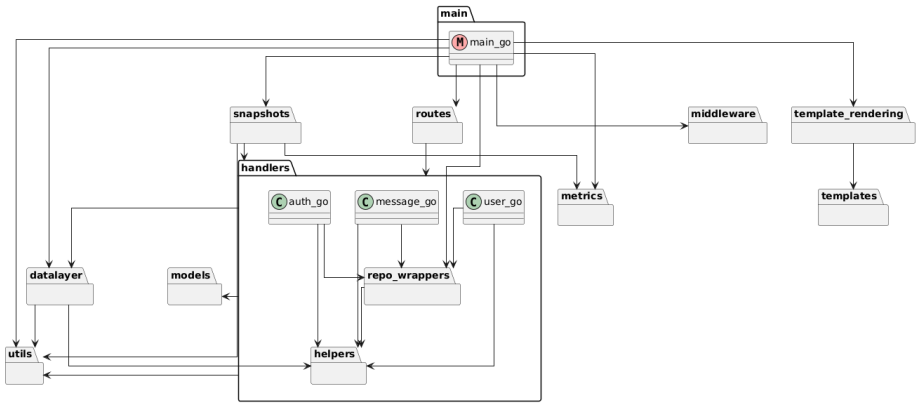

Figure 1: Module diagram

In the diagram it can be seen, that the main.go file orchestrates the system. It (in this context) has the responsibility for: 1. Render the template (frontend) 2. Initialize a new instance of the database object 3. Setup middleware 4. Setup routes, which have the responsibility of exposing the endpoints that further orchestrates to the handlers module for the logic of the API.

2.2 Dependencies

2.3 System interactions

2.4 Current state of the system

2.4.1 SonarQube analysis summary

The following table summarizes key code quality metrics from SonarQube analysis:

Metric	Value
Lines of Code (LOC)	1,591
Code Duplication	4.1%
Security Hotspots	8
Overall Rating	A (Excellent quality)
Cyclomatic Complexity	216 (handlers: 151)
Technical Debt	~1 hour 7 minutes

2.4.2 Code climate

The following table summarizes key code quality metrics from Code Climate analysis:

Metric	Value
Lines of Code (LOC)	1,912
Code Duplication	0 %
Overall Rating	A (Excellent quality)
Complexity	299 (handlers: 196)
Technical Debt	~1 day 2 hours

2.4.3 Overall assesment

Both tools show a high complexity in the handlers module

2.5 Deployment

2.5.1 Allocation viewpoint

3 Process perspective

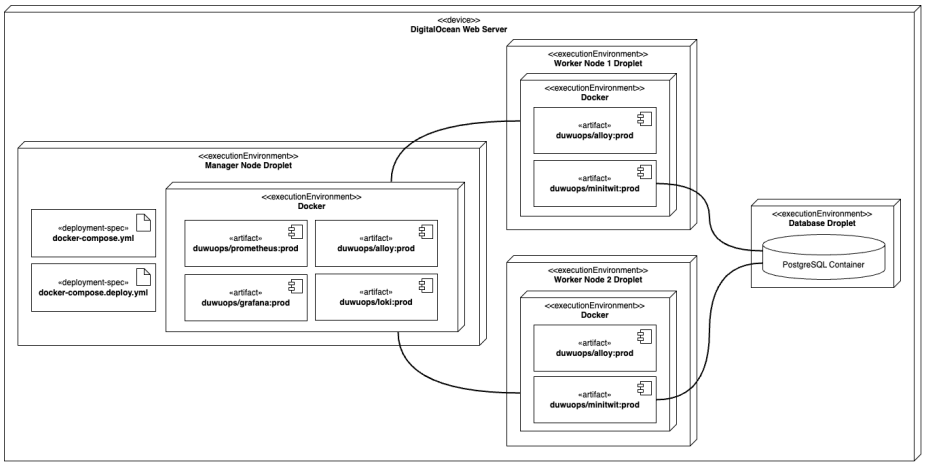

Figure 2: Deployment diagram