Möglichkeiten PHP Cloud-Anwendungen mit kompilierten Programmiersprachen zu verbinden

Systemdokumentation

Verfasser: Martin Bende
Matrikelnummer: 3293456

Fachhochschule Aachen
University of Applied Sciences
Fachbereich Elektrotechnik und Informationstechnik
Studiengang Informatik

1. Einführung

Diese Dokumentation liefert eine detaillierte Anleitung zur Integration einer Laravel Applikation mit der Programmiersprache Go auf verschiedene Weise. Die Kombination dieser beiden Technologien kann die Leistung und Effektivität der Webapplikation verbessern, in dem die Skalierbarkeit und die Fähigkeit zur Verarbeitung großer Datenmengen erhöht.
Die Dokumentation wird einen Überblick über die verschiedenen Integrationsmethoden geben. In folgender Dokumentation wird gezeigt, wie GO als separate Microservice oder als Teil der Laravel Applikation eingesetzt werden kann.
Anforderungen
Folgende Anforderungen müssen erfüllt werden, um die Anwendung starten zu können:

- Docker: 4.15.0^
- Min. Speicherplatz: 3 GB (durch die gespeicherte Daten in der Datenbank kann dieser Zahl höher werden)
- Min. RAM: 4 GB

2. Architektur

Die implementierte Webanwendung ist durch die angewendete Architektur auf jedem System verfügbar, das Docker unterstützt. Die Anwendung besteht aus mehreren Komponenten, die in verschiedenen Docker-Container ausgeführt werden (*s. Abbildung 2.1)*. Diese Container werden unter einem gemeinsamen Network (sail-network) und in einem Image (practical-project) zusammengeführt, um miteinander kommunizieren zu können und eine Einheit zu bilden.

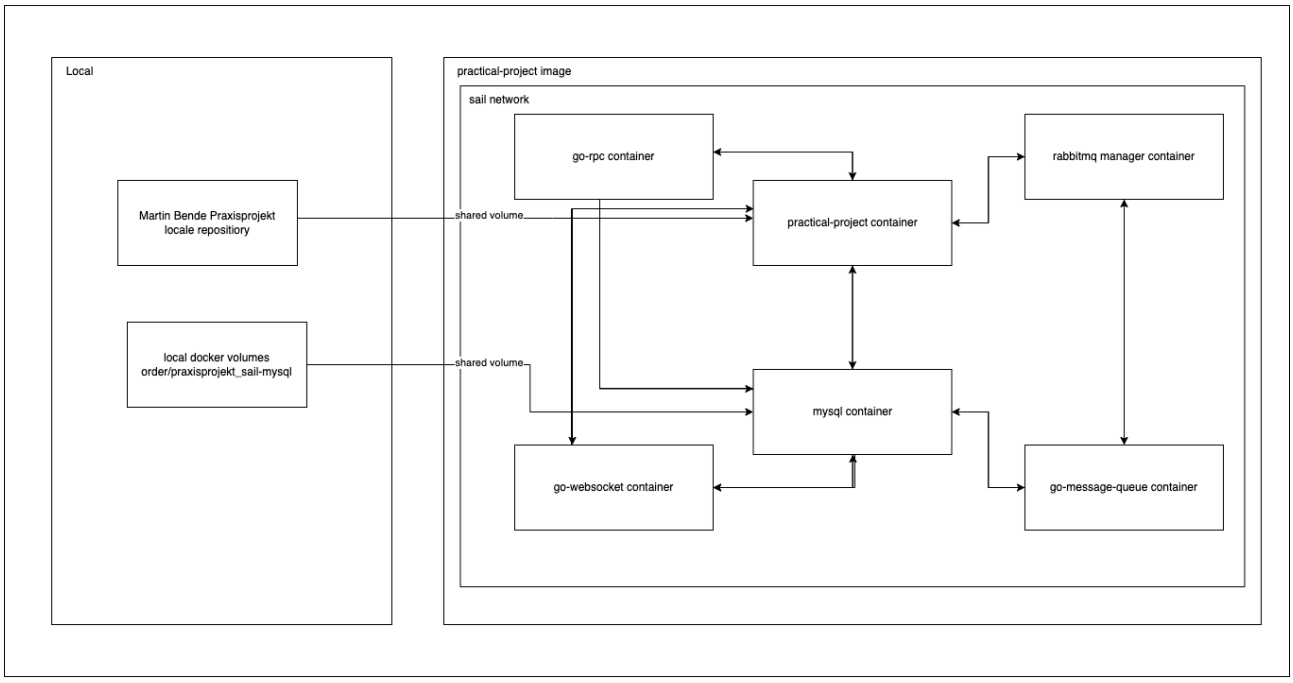

Abbildung 2.1

Pratical-project Container

In diesem Container läuft die Webanwendung selbst, der die Codebasis und die Abhängigkeiten der Anwendung enthält. Durch einem geteiltem Volume hat der Container Zugriff auf den Quellcode. Der Container wird auf dem Port 80 bereitgestellt, und kann von dem Host-Rechner über das Port 80 zugegriffen werden. Der Container wird durch den offizielle Laravel Sail Dockerfile gebaut.

Mysql Container

In diesem Container läuft ein Mysql-Server, um der Webanwendung ein Datenbank-Server bereitzustellen. Der Container wird auf dem Port 3306 bereitgestellt, und kann von dem Host-Rechner über das Port 3306 zugegriffen werden. In der Datenbank gespeicherten Daten werden über einem geteiltem Volume auf dem Host-Rechner gespeichert, um auf diesen persistenten Daten später Zugriff zu gewährleisten. Der Container verwendet das offizielle Mysql-Server 8.0 Image.

RabbitMQ-Manager Container

Um eine Art von Verbindung zwischen der Webanwendung, und der Programmiersprache GO zu realisieren, wird ein Container mit der Service RabbitMQ bereitgestellt. Der Container wird auf dem Port 5672, sowie 15672 bereitgestellt. Auf dem RabbitMQ-Manager Webseite kann über das Port 15672 zugegriffen werden. Das Port 5672 wird für interne Kommunikationszwecke genutzt. Der Container verwendet das offizielle RabbitMQ3-Management Image.

GO-Message-Queue Container:

Um eine Art von Verbindung zwischen der Webanwendung, und der Programmiersprachen GO zu realisieren, wird ein Container mit einem GO-RPC-Server bereitgestellt. Der Container ist von Host-Rechner nicht erreichbar, da die interne Kommunikation über das RabbitMQ-Manager Container stattfinden. Der Container verwendet das offizielle Golang (Version 1.20) Image.

GO-RPC Container

Um eine Art von Verbindung zwischen der Webanwendung, und der Programmiersprachen GO zu realisieren, wird ein Container mit einem GO-RPC-Server bereitgestellt. Der Container wird auf dem Port 6001 bereitgestellt, und kann von dem Host-Rechner über das Port 6001 zugegriffen werden. Der Container verwendet das offizielle Golang (Version 1.20) Image.

GO-Websocket Container

Um eine Art von Verbindung zwischen der Webanwendung, und der Programmiersprachen GO zu realisieren, wird ein Container mit einem GO-Websocket-Server bereitgestellt. Der Container wird auf dem Port 6002 bereitgestellt, und kann von dem Host-Rechner über das Port 6002 zugegriffen werden. Der Container verwendet das offizielle Golang (Version 1.20) Image.

3. Funktionsweise

In der Webanwendung ist gewährleistet, die in GO geschriebene Anwendung auf 4 verschiedene Wege zu testen. Zwischen der verschiedenen GO Applikation gibt nur ein einziger Unterschied: Die Methodik, wie die Applikation von der Außenwelt zugänglich ist. Dafür wurden verschieden Packages sowohl in PHP als auch in GO verwendet. Die angewendete Methodik und Packages werden in folgendem beschrieben.

FFI

Diese PHP Erweiterung ermöglicht das Laden von gemeinsam genutzten Bibliotheken (.DLL oder .so), den Aufruf von C-Funktionen und den Zugriff auf C-Datenstrukturen in reinem PHP, ohne dass tiefe Kenntnisse der Zend Extension API erforderlich sind und ohne dass eine dritte "Zwischen"-Sprache erlernt werden muss.
In Go kann eine Shared Library (auch Dynamically Linked Library oder kurz DLL genannt) mithilfe des "cgo" Packages erstellt werden. Das "cgo" Package ermöglicht es, C-Code in Go-Programme zu integrieren. Bei dem Start der Applikation (auf dem in Abschnitt 4), wird mithilfe von dem offiziellen Go Image (Version 1.20) ein C-Shared Bibliothek für das Betriebssystem des Pratical-project Container (s. Abschnitt 2) kompiliert. Auf dieser Weise kann die kompilierte Bibliothek in PHP geladen und als interne Funktion aufgerufen werden, die die Resultate zurückgibt.

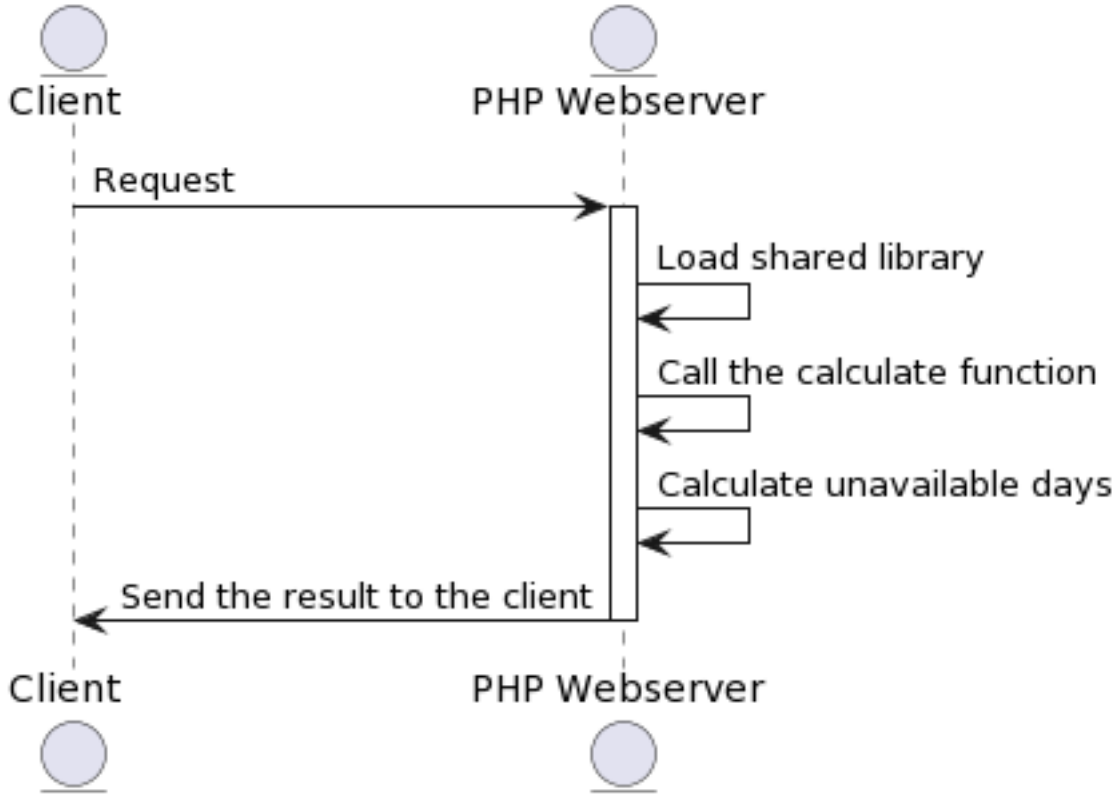

Abbildung 3.1

RPC

Remote Procedure Call ist eine Technik zur Realisierung von Interprozesskommunikation. Sie ermöglicht den Aufruf von Funktionen in anderen Adressräumen. Im Normalfall werden die aufgerufenen Funktionen auf einem anderen Computer als das aufrufende Programm ausgeführt.
In dem GO-RPC Container wird mithilfe von goridgeRpc Package ein RPC-Server gestartet. In der Laravel Webanwendung selbst wird eine RPC Call mithilfe von RoadRunner Package getätigt. Die eingegangene Request wird in der GO Anwendung verarbeitet und die entsprechende Funktion aufgerufen, die die Resultate zurückgibt.

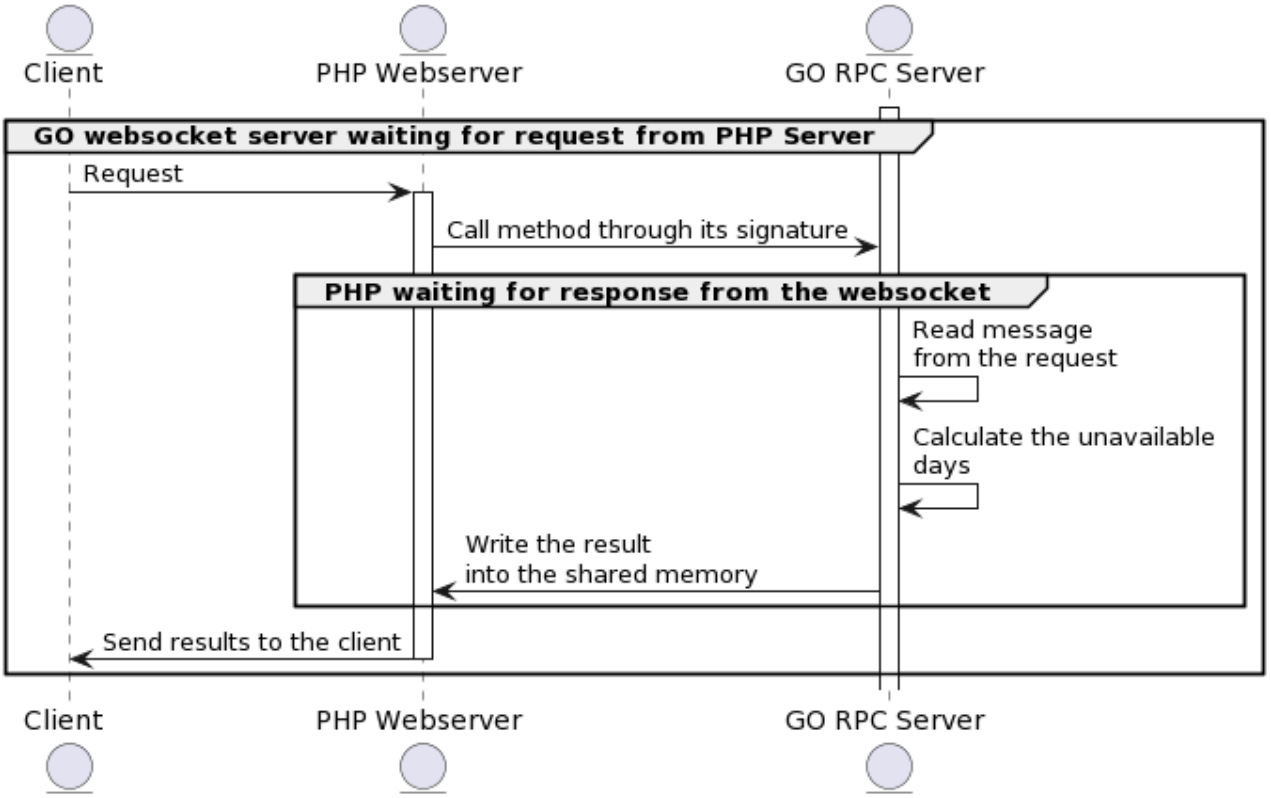

Abbildung 3.2

Websocket

Das Websocket-Protokoll ist ein auf TCP basierendes Netzwerkprotokoll, das entworfen wurde, um eine bidirektionale Verbindung zwischen einer Webanwendung und einem Websocket-Server bzw. einem Webserver, der auch Websockets unterstützt, herzustellen. Um eine bidirektionale Kommunikation zwischen Client und Server bereitzustellen, wird in dem GO-Websocket Container mithilfe von gorrila Websocket Package ein Websocket-Server aufgebaut. Auf der Seite von der Laravel Webanwendung wird mithilfe von textalk Websocket Package ein Client gestartet, der eine Message zu dem Websocket-Server sendet und blockiert, bis die Antwort von der Websocket-Server nicht empfangen werden kann. Der Websocket-Server verarbeitet die Request, ruft die entsprechende Funktion auf, und gibt die Resultate zurück

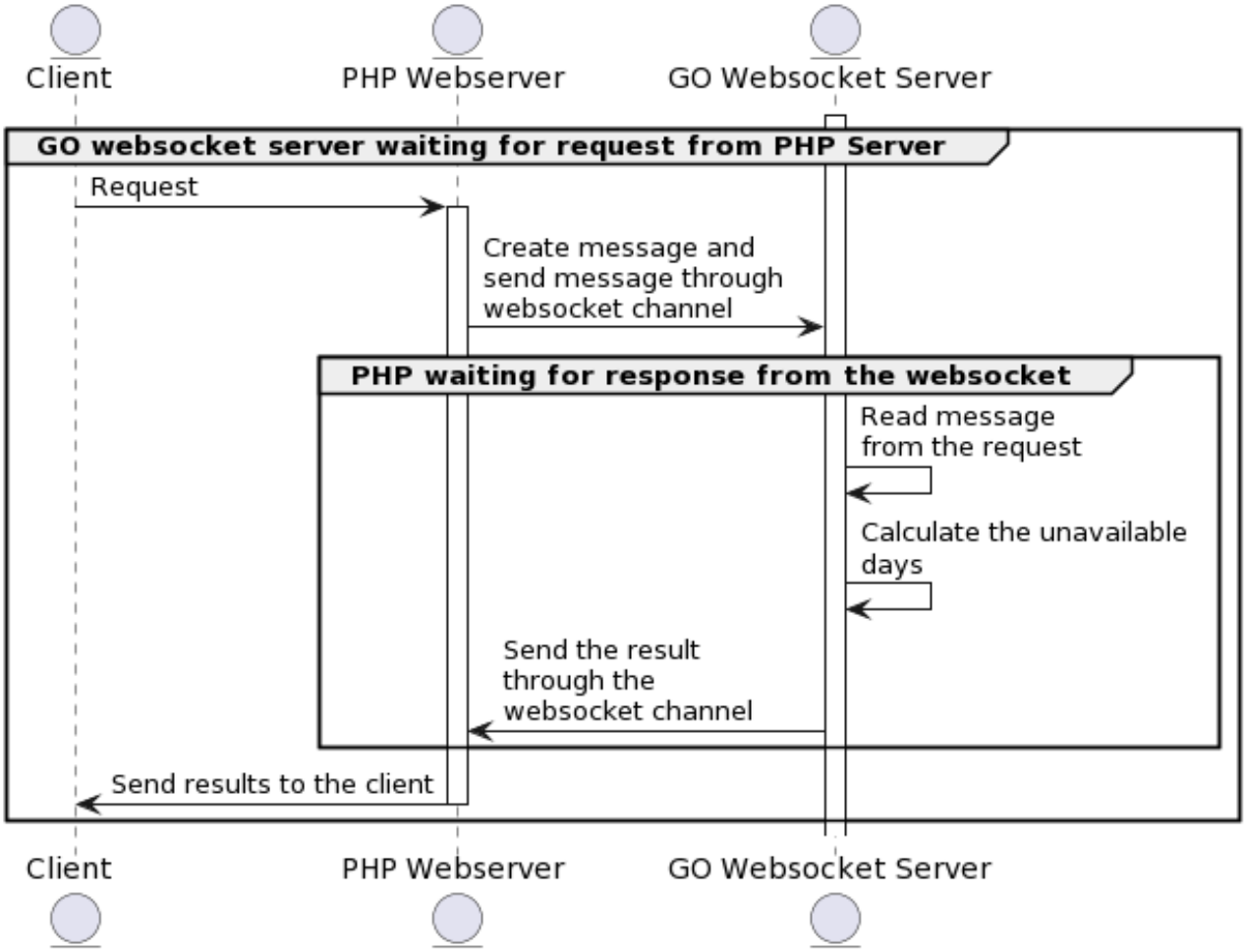

Abbildung 3.3

RabbitMQ

RabbitMQ ist eine Open Source Message Broker Software, die das Advanced Message Queuing Protocol implementiert. Bei dem Start der Applikation wird ein RabbitMQ-Manager in dem RabbitMQ-Manager Container
gestartet um den Nachrichtenaustausch zwischen dem Practical-Project Container und dem GO-Message-Queue Container zu ermöglichen. Beim Start der GO-Message-Queue Container wird 2 Message Queue (PHP-TO-GO und GO-TO-PHP) deklariert, um die bidirektionale Nachrichtenaustausch ohne Konflikten zu ermöglichen. Nachdem erfolgreichen Start des Containers lauscht die Go-Anwendung an dem PHP-TO-GO Queue, um Nachrichten zu empfangen. Wenn die Laravel Anwendung eine Nachricht auf dem PHP-TO-GO Queue push, lauscht an dem GO-TO-PHP Queue und wird blockiert. Die Nachricht wird seitens der GO Anwendung verarbeitet und die Resultate auf dem GO-TO-PHP QUEUE geschrieben. Wenn die Laravel Anwendung die Nachricht erhält, wird die Nachricht bestätigt und gelöscht, um die mehrmaligen Lesen zu vermeiden.

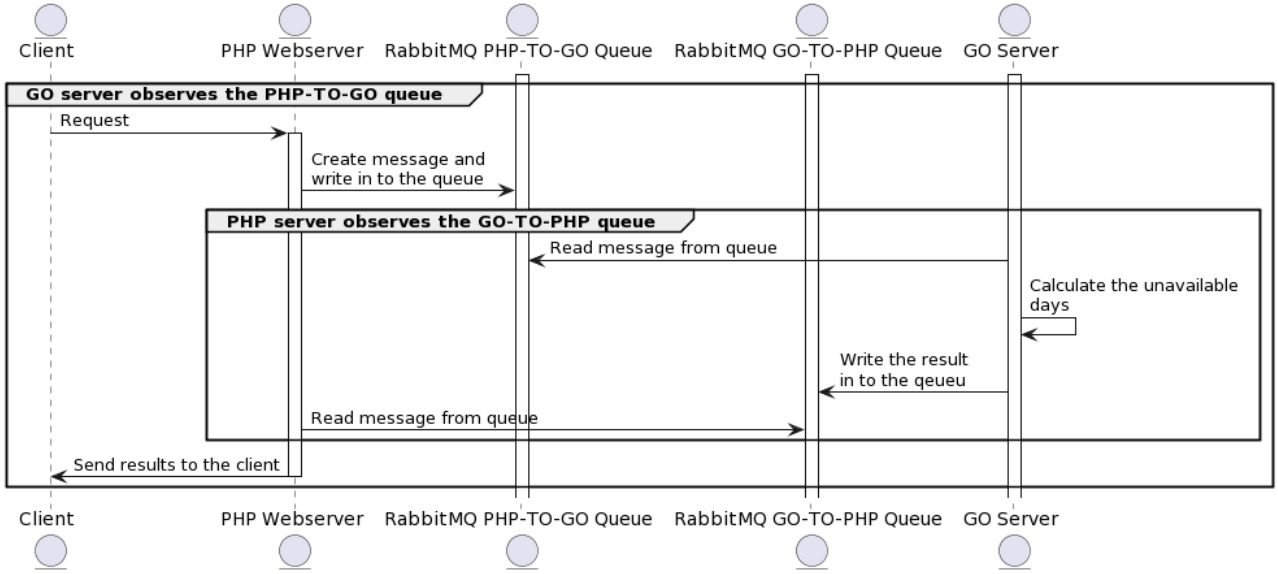

Abbildung 3.4

4. Installation

Um die Installation der Applikation zu vereinfachen, wird eine bash Datei bereitgestellt. Die Applikation kann mit folgendem Befehl im Terminal gestartet werden:

chmod 777 start-application.sh && ./start-application.sh

Bei dem ersten Start der Anwendung kann es auch mehrere Minuten in Anspruch nehmen, da alle Abhängigkeiten heruntergeladen werden müssen.

5. Wartung und Fragen

Bei Fragen oder Fehlermeldung folgende Person kontaktieren:
Martin Bende - bendemartin97@gmail.com