GO-12 01: OpenAPI и синхронное REST взаимодействие

Описание:

В данном задании мы посмотрим OpenAPI Specification (Swagger) и то, как мы способны использовать похожую спецификацию для формирования контрактов между сервисами и генерации рутинного кода клиента/сервера, имплементирующего данный контракт.
Мы будем использовать библиотеку go-swagger, которая предоставляет инструменты для работы со Swagger'ом версии 2.0 (OpenAPI 2.0).
Существует уже OpenAPI версии 3.0 и инструменты (Swagger Codegen) для различных языков (в том числе для Go), позволяющие генерировать и клиент, и сервер. Но что касается именно Go - Swagger Codegen все-таки является инструментом, написанным на Java (так что, если вы его хотите сбилдить и запустить у себя - в его зависимости входят Maven и Jvm), и для широкого употребления в GO не особо подходит, так как не выделяется богатством функционала именно для Go. Поэтому наиболее популярным решением является go-swagger, хотя он и не поддерживает OpenAPI 3.0, он отлично использует преимущества языка - go generate для перегенерации, использование преимуществ утиной типизации в местах, где можно подменять реализацию, и т.д. (На официальном сайте можно найти достаточно информации - goswagger.io.
В этом задании мы с вами сгенерируем контракт сервера для конвертации набора изображений в PDF и попробуем взаимодействовать с ним через уже написанную имплементацию клиента.

Полезные ссылки:

- goswagger.io
- OpenAPI
- Swagger

Задание:

1. Форкните репозиторий module12 с кодом данного задания в группу с вашими репозиториями - golang_users_repos/<your_gitlab_id> и создайте в нем из ветки master ветку module12_01.
2. Внутри вы найдете docker-compose.yml, который запускает клиент, сервер и директории с клиентом и сервером, соответственно (перед эти нужно выполнить следующие ниже пункты и сгенерировать соответствующий код).

Если у вас не установлен docker-compose - установите.

3. Внутри директории server вы найдете файл api/api.yaml, содержащий описание нашего API.

Для визуализации вы можете использовать Swagger Editor или встроенный инструмент библиотеки swagger serve api/api.yml.

4. Сгеренерируйте сервер с помощью команды swagger generate server для нашего api внутри директории server, придерживаясь при этом нашего layout'a (вся логика должна лечь внутрь директории internal - для этого у команды есть соответствующий флаг: --server-package=internal/handler).

Так же используйте при генерации флаг -A "service-pdf-compose" для корректного наименования сервиса.

5. Обратите внимание - библиотека сгенерировала нам заглушку сервера вместе с валидацией, необходимыми сущностями для параметров, возможными вариантами ответов и т.д.
6. У вас появится единственный доступный для редактирования файл в директории internal/handler, где вы найдете обработчик endpoint'a, для которого требуется имплементация.
7. Используя заготовленную функцию, из пакета composer внутри директории pkg имплементируйте наш endpoint, так чтобы он возвращал нам сгенерированный из изображений PDF файл. Для ответов сервера используйте сгенерированные структуры.
8. Сгенерируйте клиент для сервиса-контроллера (директория controller) с помощью команды swagger generate client. Используя сгенерированный клиент, добавьте логику в эндпоинт Send в controller/internal/handlers/page.go таким образом, чтоб контроллер в html форме мог корректно принимать файлы, далее отправлять их на конвертацию в server и возвращать сконвертированные файлы клиенту.

Во избежание возникновения на стороне контроллера ошибок класса no consumer: используйте для клинета кастомный транспорт с явным добавлением consumer'a с mime application/pdf:

```
transport := httptransport.New(apiclient.DefaultHost+":8090",
apiclient.DefaultBasePath, []string{"http"})
transport.Consumers["application/pdf"] = runtime.ByteStreamConsumer()
```

9. Запустите контейнеры с помощью команды docker-compose up (используется docker-compose.yml из корня репозитория).
10. Используя визуальный интерфейс клиента, приложите файлы и проверьте корректность конвертации их сервером.
11. В ответ пришлите ссылку на merge request в ветку master вашего проекта ветки module12_01.