Тестовое задание для обучающихся Go: Система управления библиотекой

Разработайте REST API на Go для управления коллекцией книг и авторами, которые их пишут. API будет выполнять операции CRUD с книгами и авторами, хранящимися в базе данных PostgreSQL. Система должна быть контейнеризирована с помощью Docker, состоящего из двух отдельных контейнеров: одного для веб-приложения и одного для базы данных.

Функционал:

- Интерфейс: приложение должно соответствовать набору архитектурных правил REST, реализуя RESTful API на основе протокола HTTP и его методов GET, POST, PUT и DELETE;
- Формат данных: используйте JSON для передачи данных в приложение;
- Подключение к базе данных: установите соединение с базой данных PostgreSQL, развернутой в Docker-контейнере;
- Операции CRUD: реализуйте функциональность операций, используя стандартный пакет database/sql Go;
- Транзакции: реализуйте транзакцию для операции, которая включает одновременное обновление книги и соответствующего автора, например обновление названия книги и биографии автора за один раз. Это следует выполнять только в том случае, если оба обновления пройдут успешно;
- Обработка ошибок: корректные ответы на ошибки в различных сценариях сбоев, например, книга не найдена, передана некорректная дата рождения, ошибки подключения к базе данных и др. Необходимо корректно использовать коды ответов HTTP для таких сценариев.

Спецификация API:

Книга должна иметь структуру:
- ID: уникальный идентификатор книги;
- Название: Название книги;
- AuthorID: ссылка на автора через его уникальный идентификатор;
- Год: Год издания книги;
- ISBN: Уникальный ISBN книги.

Автор должен иметь структуру:
- ID: уникальный идентификатор автора;
- Имя: имя автора;
- Фамилия: фамилия автора;
- Биография: краткое описание деятельности автора;
- Дата рождения: дата в формате yyyy-mm-dd.

Ваше приложение должно предоставлять следующие RESTful эндпоинты:

Для книг:
- POST/books — Добавить новую книгу;
- GET /books — Получить все книги;
- GET /books/{id} — Получить книгу по ее идентификатору;
- PUT /books/{id} — обновить книгу по ее идентификатору;
- DELETE /books/{id} — Удалить книгу по ее идентификатору.

Для авторов:
- POST/authors — Добавить нового автора;
- GET /authors — Получить всех авторов;
- GET /authors/{id} — получить автора по его идентификатору;
- PUT /authors/{id} — обновить автора по его идентификатору;
- DELETE /authors/{id} — удалить автора по его идентификатору.

Транзакционное обновление:
- PUT /books/{book_id}/authors/{author_id} — одновременно обновить сведения о книге и авторе.

Развертывание:

- Опишите Dockerfile для образа приложения Go;
- Настройте Docker-контейнер PostgreSQL, используя готовый образ с Docker-hub;
- Используйте docker-compose для поднятия обоих контейнеров, гарантируя, что они могут взаимодействовать.

Документация:

Предоставьте файл README.md с инструкциями по сборке и запуску приложения. Запуск должен осуществляться через инструкции в Makefile, примеры ключей: make build, make run.

Критерии оценки:

- Функциональность: Приложение должно отвечать всем указанным выше требованиям;
- Качество кода: код должен быть чистым, имена идентификаторов и пакетов должны соответствовать принятым стандартам. Структура должна соответствовать примеру clean-architecture с лекций, должны использоваться интерфейсы для разделения слоев в приложении. Код должен быть читаемым, где необходимо могут быть комментарии;
- Обработка ошибок: приложение должно корректно обрабатывать ошибки и реагировать на них. Создайте свои типы ошибок и используйте функции для работы с ними, которые посчитаете необходимыми.