Level 0. Research + DB

. 0.1 Разобраться с шаблоном проектирования БД для хранения и обработки иерархических структур Nested Set

- выбран формат ранжирования с левым и правым краями;
- время - 30 минут.

. 0.2 Написать Хранимые процедуры для MySQL или PostgreSQL реализующие методы для Nested Set:

- Приведение данных к nested set через parent_id
(не готово в виду длительной реализации задачи)

- Добавление узла в дерево:
(package “postgres”, f_add_book_category.sql)

- Удаление узла из дерева:
(package “postgres”, f_remove_book_category.sql)

- Перенос узла внутри дерева:

(package “postgres”, f_move_book_category.sql)

Level 1. Go BackEnd

. 1.1 Написать на языке GoAPI сервис реализующий методы:

(реализация сервиса регистрирующего endpoints: service/RestService.go)

- Нормализация дерева (bool)
(не готово)

- Добавление узла в дерево (id)
(handler/RestHandler/AddBookCategory)

- Удаление узла из дерева (bool)
(handler/RestHandler/RemoveBookCategory)

- Перенос узла внутри дерева (bool)
(handler/RestHandler/MoveBookCategory)

- Получить всех родителей узла (id[])
(handler/RestHandler/GetNodeParents)

- Получить всех прямых потомков узла (id[])
(не готово)

- Получить всех потомков узла (id[])
(handler/RestHandler/GetNodeChildren)

- Получить всех соседей узла с общим родителем (id[])
(не готово)

- Получить уровень вложенности узла (int)
(не готово)

. 1.2 ЗадокументироватьAPIвSwagger.io
(не готово)

. 1.3 Подготовить Postman проект тестирования API

https://d325b419-6dbb-474a-9b4f-95125f121976.mock.pstmn.io

(postman collection inside the project, package “postman”)

. 1.4 Упаковать сервис в Docker контейнер и выгрузить в приватный репозитарий на Docker Hub
(не готово. проблем с реализацией нет, но времени потрачено слишком много на выполнение задания в целом)

. *1.5 Написатьтесты*
(нужно их предварительно изучить для GO).

Level 2. JS FrontEnd

. 2.1 Написать интерфейс для визуализации и управления деревом в связке с предыдущим API сервисом. Первоначальный рендеринг дерева и обновление после всех управляющих действий должно быть асинхронно. Функционал:

- Отобразить дерево

- Добавление узла в дерево

- Удаление узла из дерева

- Перенос узла внутри дерева Drag & Drop

- Отображение ошибок

- Восстановление дерева после ошибок

. 2.2 Адаптация интерфейса под отображение и управления на мобильных устройствах

Общие положения

2.3 Google в помощь
2.4 Если не умеете в SQL, Go или JS – быстро нахвататься сколько сможете и

зафиксировать источники откуда хватали
2.5 Для 1.1 не использовать готовые библиотеки
2.6 Задания 1.2, 1.4 и 2.2 опциональны и являются бонусными
2.7 Комментируем код особенно в 0.2 и 1.1
2.8 В задании 2.1 используем любой JS фреймворк
2.9 Фиксируем время, затраченное на выполнение каждого задания

2.10 Фиксируем замечания и комментарии в ходе выполнения заданий

Разработкой во Frontend не обладаю, но при некотором потраченном времени можно освоиться. Также, приступая к реализации не было что-либо известно про Golang. Также Postgres на минимальном уровне.

Не все задания получилось выполнить из-за длительного выполнения задания в целом. Сами задания понятные, не сложные и вполне решаемые, но было потрачено значительное время на выявление проблемы остановки сервиса после запуска - не правильное использование логгера. Привычка с других языков программирования.

В целом впечатление положительное от задания.

Источники;

- по работе с Golang / Postgres в основном официальные.