Тестовое задание на позицию Младший разработчик Java в МойСклад

Предметная область

В модели предметной области имеются **сущности** (в скобках указаны обязательные атрибуты):

- **Товар** (Артикул, Наименование, Цена последней закупки, Цена последней продажи).
- **Склад** (Наименование). На складе может храниться несколько товаров.

Документы:

- **Поступление** (Номер, Склад, Список товаров). Заводится при поступлении товара. Содержит список товаров, их количество и закупочные цены. В документе указывается склад, на который поступают товары.
- **Продажа** (Номер, Склад, Список товаров). Заводится при продаже товара. Содержит список товаров, их количество и цены продажи. В документе указывается склад, с которого списывают товары.
- **Перемещение** (Номер, Склад1, Склад2, Список товаров). Заводится при перемещении товара между складами. Содержит список товаров и их количество. В документе указывается склад, с которого списывают товары и склад, на который они поступают.

Постановка задачи (backend)

Необходимо реализовать серверную часть приложения по учету товаров на складе. Приложение должно включать API для просмотра, создания, редактирования и удаления сущностей.
На вход приложению поступают документы: API включает операции импорта (создания) и просмотра документов.
На выходе имеется возможность сформировать отчеты:

- Общий список товаров (артикул, наименование, цены закупки и продажи). В качестве опционального параметра может быть передан фильтр по имени товара.
- Остатки товаров на складах (артикул, наименование, остаток по всем складам). В качестве опционального параметра может быть передан фильтр по складу.

API для работы с сущностями оперирует форматом JSON.
Работу с документами также можно проводить в формате JSON, либо использовать другой человеко-читаемый формат (например CSV).
Формат выводимых отчетов на выбор: JSON, CSV, HTML (а можно и все три :)).

Функциональные требования

- Предусмотреть валидацию входных параметров. Программа не должна «упасть» на некорректных данных.
- Выполнение любой операции должно возвращать результат — ОК либо сообщение/код ошибки.

Нефункциональные требования

- Язык разработки: Java.
- Фреймворк: использование Spring нежелательно, лучше обойтись без него.
- Данные можно хранить в памяти (например, в виде списка), либо использовать для хранения PostgreSQL / H2 (будет плюсом).
- Сборка с помощью инструмента Maven без установки или настройки каких-либо дополнительных компонент.
- Выполненное задание должно быть размещено в публичном репозитории на Github / Bitbucket.
- Файл README должен содержать инструкцию по сборке, настройке, конфигурированию и развертыванию приложения (если необходимо).

Дополнительные возможности

К реализованному API можно составить небольшую документацию — список возможных команд с передаваемыми в них параметрами и возможными ответами на запрос. Файл с документацией может быть в формате Markdown, либо с использованием инструментов для документирования АПИ(API Blueprint, Swagger и подобные).

Реализованный функционал желательно покрыть юнит-тестами и интеграционными тестами.

Постановка задачи (frontend) — опционально

Плюсом будет создание UI для серверного приложения.
Язык разработки и фреймворк: на ваш выбор.
Дизайн страницы (страниц): на ваш выбор.
Желательно наличие возможности аутентификации.