Техническое задание

Проект: Аналог «E-Каталог» (агрегатор цен по магазинам)

Версия ТЗ: v2 (с парсингом цен и витриной предложений)

1. Цель работы

Разработать веб-приложение — аналог «E-Каталог», которое позволяет искать товары и сравнивать цены в нескольких магазинах (минимум 3 источника), показывая самое выгодное предложение и остальные предложения ниже.

2. Основной сценарий пользователя

- Пользователь выбирает город (например: Екатеринбург).
- В шапке сайта есть динамическое поле поиска (подсказки по мере ввода).
- Пользователь выбирает товар из подсказок и переходит на страницу товара.
- На странице товара сверху отображается лучшее предложение (минимальная итоговая цена).
- Ниже показывается список остальных магазинов, отсортированный по цене (от дешёвых к дорогим).

3. Источники цен (магазины)

Приложение должно собирать цены минимум из трёх магазинов. Примеры: «М.Видео», «DNS», «Ozon», «Ситилинк» и др. Конкретный набор магазинов выбирается студентом и фиксируется в проекте.

- Минимум 3 источника (магазина).
- Каждый источник должен отдавать/давать возможность получить: название товара, цена, город (или регион), ссылка на товар, наличие (если доступно), скидка (если есть).

4. Парсинг/получение данных по магазинам

Получение данных может быть реализовано одним из способов (выбрать любой):

- Парсинг HTML страниц магазинов (requests + парсер HTML).
- Публичный API магазина (если доступен).
- Внешний сервис/фид (если законно доступен и без закрытых ключей).

Важно: необходимо соблюдать корректную частоту запросов и не выполнять агрессивный скрейпинг. Для учебного проекта допускается ограничение ассортимента (например, 30–100 товаров в базе).

5. Данные, которые нужно хранить в MySQL

- Товары: id, название, бренд (опц.), категория (опц.), изображение (URL/файл), краткое описание (опц.).
- Магазины: id, название, сайт/домен.

- Города: id, название.
- Предложения (offers): товар_id, магазин_id, город_id, цена_обычная, цена_со_скидкой (если есть), размер_скидки (если есть), ссылка, наличие (опц.), дата_обновления.

6. Функциональные требования (пользовательская часть)

- Выбор города (влияние на выдачу цен).
- Поиск товаров в поле ввода с подсказками (autocomplete).
- Страница товара:
- — блок «Лучшее предложение» (самая низкая итоговая цена по выбранному городу);
- — блок «Другие магазины» (таблица/список остальных предложений, сортировка по цене).
- Отображение полей предложения: магазин, цена, цена со скидкой/скидка, наличие (если есть), ссылка «Перейти в магазин».

7. Административная часть (минимум)

- Авторизация администратора.
- Управление списком магазинов (включить/выключить источник).
- Управление списком городов.
- Запуск обновления цен вручную (кнопка «Обновить предложения»).
- Просмотр логов обновления (успех/ошибки по источникам).

8. Обновление цен

- Цены должны обновляться по расписанию или вручную из админки.
- Минимум: обновление вручную + фоновая задача (cron/планировщик) 1 раз в сутки.
- При обновлении сохранять дату/время обновления для каждого предложения.

9. Технические требования

- Frontend: HTML/CSS/JS или React.
- Backend: Node.js или PHP.
- База данных: MySQL.
- Адаптивность: Desktop / Tablet / Mobile.
- Код проекта: понятная структура, README с инструкцией запуска.

10. Ограничения

- Оформление заказа/оплата не требуются.
- Достаточно ограниченного ассортимента (например 30–100 товаров), но с реальными ценами из выбранных источников.

- Допускается упрощение категорий и характеристик товаров.

11. Результат работы

- Исходный код проекта (архив или репозиторий).
- Развёрнутое приложение (локально или на сервере).
- База данных с тестовыми товарами и предложениями.
- Демонстрация страницы товара с корректным блоком «Лучшее предложение» и списком остальных магазинов.

12. Критерии оценки

- Рабочий поиск с динамическими подсказками.
- Корректная агрегация предложений минимум из 3 источников.
- Учет города (цены/наличие меняются в зависимости от города, если источник это поддерживает).
- Правильная логика выбора «лучшего предложения» и сортировки остальных.
- Стабильность обновления и наличие логов/обработки ошибок.
- Качество интерфейса и адаптивность.