Урок 11 > Дизайним интернет-магазин

> Оглавление

> Общие сведения

Для описания дизайн-системы продукта удобно пользоваться онлайн-сервисами, например:

- Excalidraw
- Miro

Что должно быть в хорошем дизайне

1. Раздел Scope Refinement - здесь формулируется перечень того что конкретно будет реализовано в проекте.
2. Раздел Functional Requirements (Функциональные требования)

3. Раздел Non-functional Requirements (Нефункциональные требования)
4. Раздел с оценкой нагрузки новой системы
5. Раздел Высокоуровневый дизайн
6. Раздел Компонентный дизайн
7. Когда готов компонентный дизайн, имеет смысл подобрать и описать базы данных для конкретных модулей и компонентов
8. Описание масштабируемости системы

> Границы проекта (Scope refinement)

В разделе формируются рамки функциональности для конкретного проекта. В качестве примера онлайн-магазина, можно описать, что это будет площадка, где заказчик систему будет размещать свой товар и некий функционал, позволяющий пользователю посмотреть перечень товаров, ознакомится с деталями позиции, положить товары в корзину и оплатить их. Можно прописать что в функционале не будет возможности пользователю зарегистрироваться и разместить на данной площадке свои товары для реализации (не marketplace).

Также, пропишем, что другие части бизнеса, к примеру, бухгалтерия, логистика, доставка товаров и т.д. выходят за рамки текущей системы.

Подобные вещи нужно зафиксировать и согласовать с заказчиком в первую очередь:

- У заказчика будет ясная картина того, как данная система поможет в его предприятии
- У команды разработки будут четкие границы реализации будущей системы

> Функциональный требования (Functional Requirements)

В разделе перечисляются все функциональные требования системы. Так, для онлайн-магазина список требований может быть таким:

1. Главная страница сайта с витриной товаров
2. Поиск нужного товара
3. Просмотр страницы товара
4. Корзина и оплата заказа
5. Управление заказами

Конечно от проекта к проекту требования могут различаться и дополнять текущий перечень, но приведенный список это минимальный набор.

> Нефункциональные требования (Non-functional Requirements)

Основные требования к интернет магазину это:

1. Отзывчивость системы
2. Высокая доступность
3. Консистентность

> Оценка нагрузки (Capacity Estimation)

Для крупного интернет-магазина возьмем следующие метрики трафика:

1. MAU (Month Active Users) - 100M. 100 миллионов активных пользователей в месяц
2. DAU (Daily Active Users) - 10M. 10 миллионов активных пользователей в день
3. 100M goods

Принимая во внимание, что наш сервис это интернет-магазин, становится очевидным, что пользователи будут просматривать много контента - карточки товаров и изображения этих товаров. Возьмем в расчет, что на каждой карточке товара присутствует 10 изображений, и примем, что активный пользователь ежедневно просматривает полностью 10 карточек и совершает 1 заказ

4. 100 images views per active user daily

5. 1 order per active user daily

Расчетные показатели по загрузке

1. 10M пользователей * 100 images / 100000 секунд в дне= 10000 RPS — чтение
2. 10K / 100 = 100 RPS — запись. Формирование заказов

Сетевой трафик

1. 10000 секунд * 500KB (вес 1 изображения в карточке товара) = 5GB/s
2. 5GB/s * 100000/s *400 day = 200PB трафика

Нагрузка на хранилище

100M товаров * 10 изображений в карточке * 500KB = 500TB. При этом, если предусмотреть масштабирование, можем прибавить 20%.

Нагрузка на вычислительные мощности

1. Чтение / запись <= 10 instances
2. Сетевой трафик >= 40 instances
3. Хранилище <= 100 SSD ~ 20 physical instances or ~ 100 cloud instances

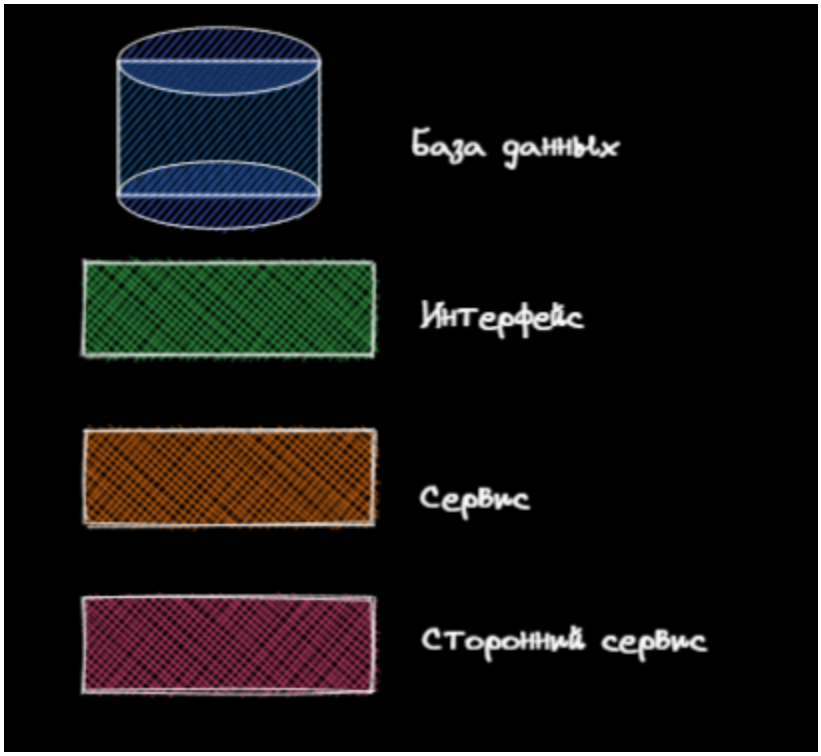
---

Стоимость

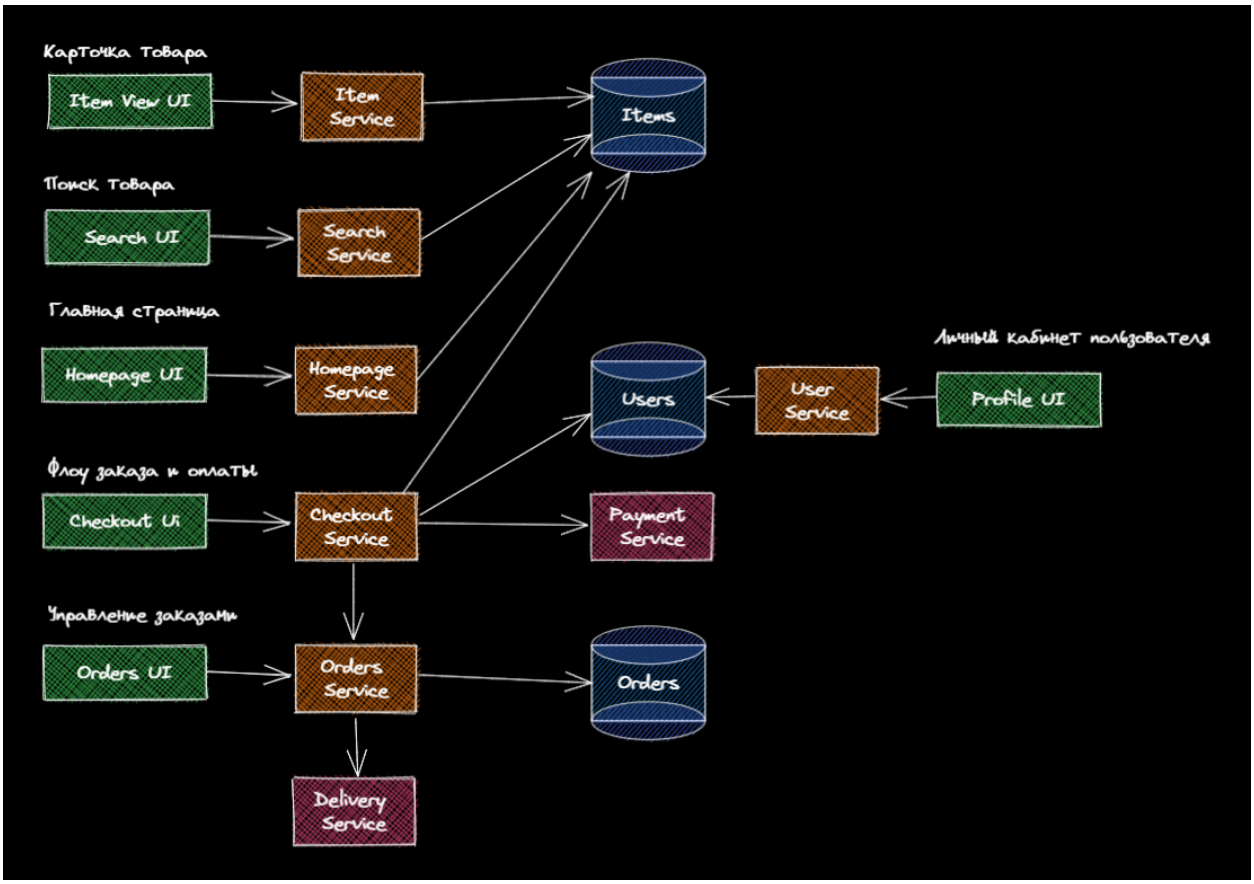
1. $0.1 / gb * 200 000 000 GB = $20M
2. 600TB * $300 = $180 000

> Высокоуровневый дизайн

На схеме высокоуровневого дизайна визуализируем каждое из функциональных требований с точки зрения пользовательского интерфейса и бэкенд-сервиса отвечающего за логику и хранилище данных.



Легенда

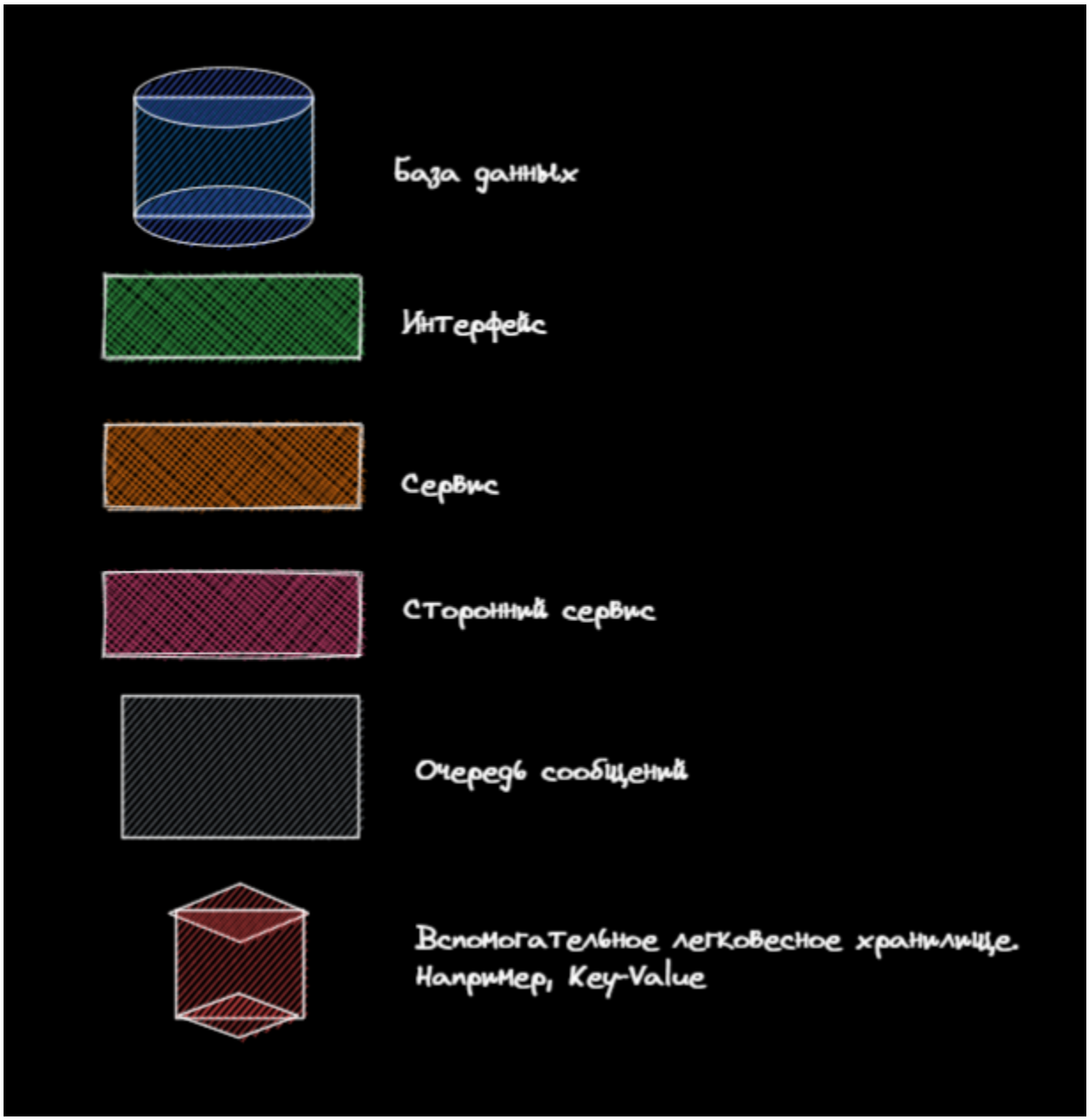


Высокоуровневый дизайн интернет-магазина

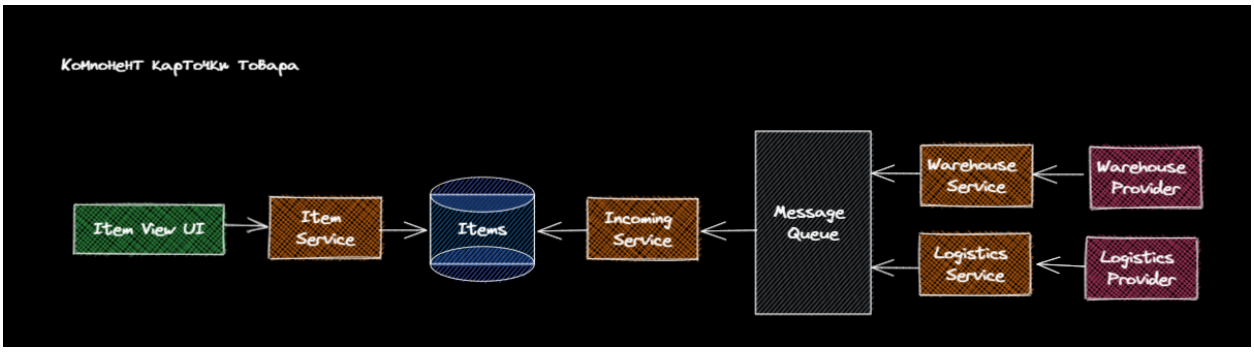
> Компонентный дизайн

Компонентный дизайн представляет собой более детальную прорисовку модулей, подробную прорисовку взаимодействия с внешними сервисами, относительно нашей системы и взаимодействия отдельных модулей между собой.

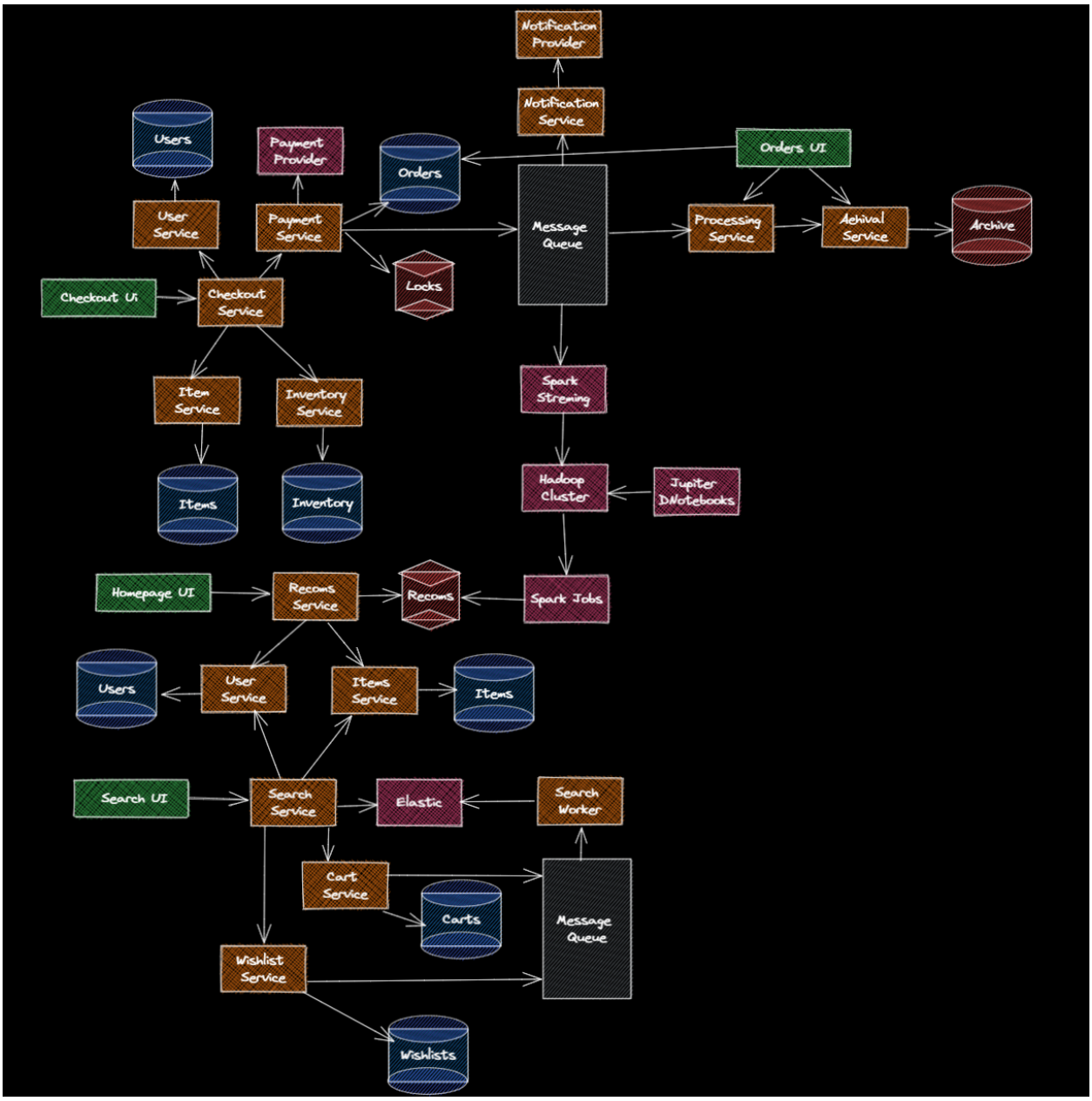
К внешним сервисам относим не только сервисы контрагентов, но и используемый компанией софт, который не является частью описываемой системы.


База данных
Интерфейс
Сервис
Сторонний сервис
Очередь сообщений
Вспомогательное легковесное хранилище.
Например, Key-Value

Легенда



Карточка товар



Дизайн остальных компонент

> Выбор хранилищ под систему

Items — Хранит в себе информацию о карточках товаров. Карточки имеют описательную структуру и у каждой карточки могут быть конкретные

характеристики, подходящие только к ней. Поэтому, здесь лучше всего подойдет документная база данных типа **MongoDB**.

Users — Содержит информацию о пользователях. Данные унифицированы и к ним должны применяться ACID требования. Потому здесь лучше использовать реляционные базы данных, например MySQL.

Orders — Содержит информацию о всех активных заказах за какой-то некий отрезок времени. С этими заказами пока ведется работа внутренних служб и пользователей, поэтому под это хранилище лучше также выбрать MySQL.

Archive — Содержит информацию о исполненных заказах. Для таких заказов может подойти колоночная база, например, Cassandra.

Locks, Inventory, Recoms — К этим хранилищам подойдут база данных в типа key-value, например Redis. Locks хранит информацию о том что единица товара заблокирована пользователем (находится на стадии оформления заказа) и не доступна другим пользователям. Inventory хранит информацию о количественных остатках товаров на складе. Recoms это список рекомендаций для конкретного пользователя.

Carts — Корзины пользователей. У каждого пользователя может быть только одна корзина, поэтому, изначально можем использовать MySQL.