Урок 2

Компания Make Cats Free предоставляет помощь котам-тестировщикам (клиентам) в выполнении рутинных задач, то есть **организация услуг** (создание, назначение, выполнение, оплата).
Ключевое конкурентное преимущество (в будущем) — **инновационная система матчинга** (подбора воркера под заказ).
Получается всё, что касается жизненного цикла «заказа услуги» и «подбора исполнителя», является сердцем бизнеса. (Это и есть наш бизнес домен)

Поддомены

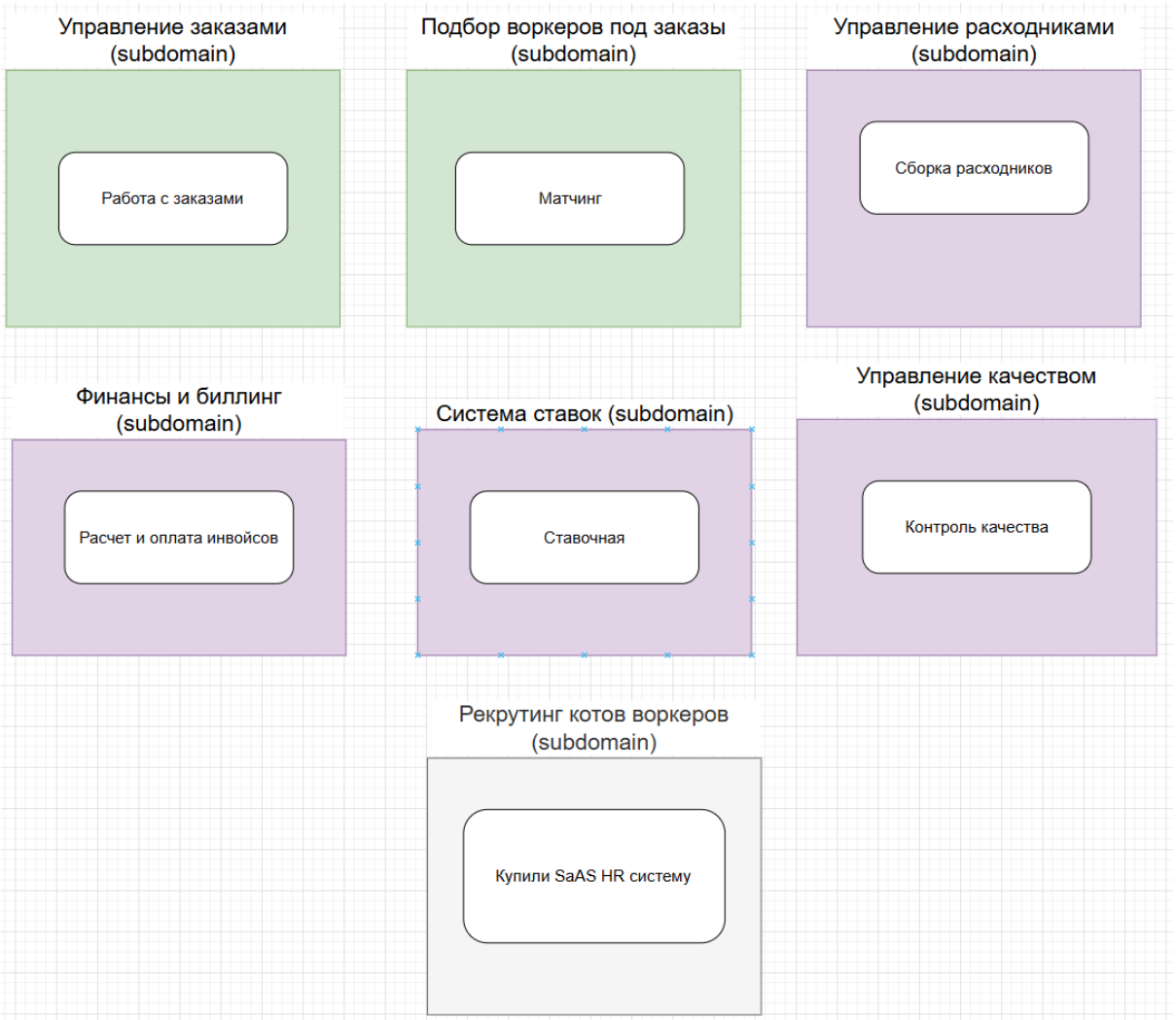
1) Управление заказами
    a) Это ядро всего процесса, где формируется основная ценность: клиент создаёт задачу, которая должна быть выполнена.
2) Подбор воркеров под заказы
    a) Главная фича компании, где планируются эксперименты и частые доработки, поэтому решил выделить в отдельный поддомен.
3) Управление расходниками
    a) Здесь потенциально сложный отдельный бизнес процесс по сборке расходников под заказы, заказ fur tune печенья (есть сомнения что оно должно быть в отдельном контексте, например вдруг бизнес захочет переделать механизм доставки печенья). Сборкой занимаются менеджеры отдела расходников.
4) Рекрутинг котов воркеров
    a) Здесь решается своя бизнес задача по найму воркеров, не связанная с заказами, а также потенциально здесь возможен поток больших данных.
5) Финансы и биллинг
    a) Выделил работу с деньгами в отдельный поддомен, потому что тут используются свои термины, отдельный свой бизнес процесс, хоть и имеющий тесную связанность с заказами, а также здесь происходит интеграция с внешними платежными системами. Тут есть доступ у менеджеров (начисление премии воркеру) и думаю доступ должен быть только у фин менеджера.
6) Система ставок
    a) Решил выделить в отдельный поддомен, т.к. для компании это второплановая задача.
7) Управление качеством
    a) Здесь работу над исследованием качества выполнения заказа и исследовании причин провала заказов, а также формирования гипотез по улучшению бизнеса занимаются менеджеры отдела качества.

Определение типа поддоменов

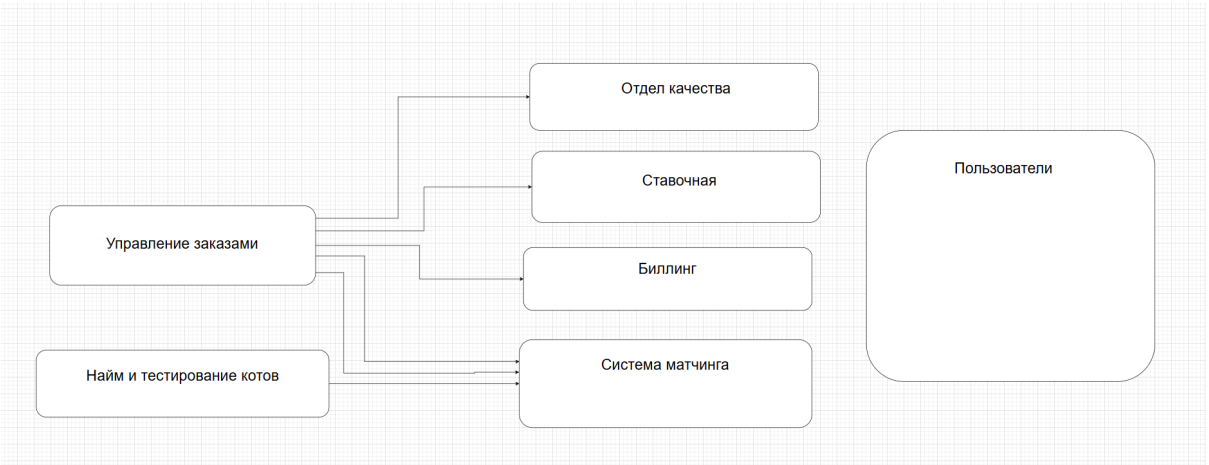
Вид поддомена	**Конкурентное преимущество**	**Сложность доменной модели**	**Изменчивость**	**Варианты реализации**	**Интересы проблемы**	**Предполагаемый вид поддомена**
Управление заказами	да	высокая	средняя	разработка	высокий	core
Подбор воркеров под заказы	да	высокая	высокая	разработка	высокий	core
Управление расходниками	нет	низкая	низкая	разработка	низкий	supporting
Рекрутинг котов воркеров	нет	высокая	низкая	возможно покупка готового решения	низкий	generic
Финансы и биллинг	нет	высокая	низкая	разработка	низкий	supporting
Система ставок	нет	низкая	низкая	разработка	низкий	supporting
Управление качеством	нет	низкая	низкая	разработка	низкий	supporting


High
Complexity
Low
Low
Business differentiation
High
Generic
Supporting
Core
Рекрутинг воркеров
Финансы и биллинг
Управление заказами
Подбор воркеров под заказы
Управление расходниками
Система ставок
Управление качеством

Определение Bounded Contexts



Вот моя упрощенная схема из ES



У меня куда то потерялся домен с пользователями (где заводятся клиенты и менеджеры, это я пытался решить проблему данных клиентов и менеджеров, т.к.

неизвестно как они попадают в систему, я выделил для них отдельный контекст в ES)
По расхождениям - вроде бы все +- совпало. (Ну на самом деле после разбора первой домашки я переделал домашку первого урока)
Схема ES и модель данных слишком большая чтобы сюда приложить, она будет в репозитории лежать.

Характеристики важные для проекта

Для всего проекта нам важны характеристики agility, testability и deployability, т.к. нам важен Time to Market. (Взял с урока 2.2)
“Для бизнеса критично проверять новые гипотезы по отсеву котов и изменять уже существующие с максимальной скоростью и надёжностью.” - значит для поддомена рекрутинга нам важны Performance, Availability, Reliability,.

1. **Scalability (Масштабируемость)**

   Выдерживать рост заявок (рекрутинга) и заказов (клиентов), а также увеличивающийся объём транзакций.

2. **Reliability / Availability (Надёжность / Доступность)**

   Избегать сбоев в критических процессах, обеспечивать устойчивую работу системы.

3. **Maintainability / Modifiability (Поддерживаемость / Лёгкость внесения изменений)**

   Необходим для core доменов, также низкий каплинг.

4. **Testability (Тестопригодность)**

   Упрощать проверку системы после внесения изменений, важно для частых экспериментов с новым функционалом.

5. **Resilience / Fault-tolerance (Устойчивость к сбоям)**

   Если одна часть выходит из строя, остальные модули продолжают работать корректно.

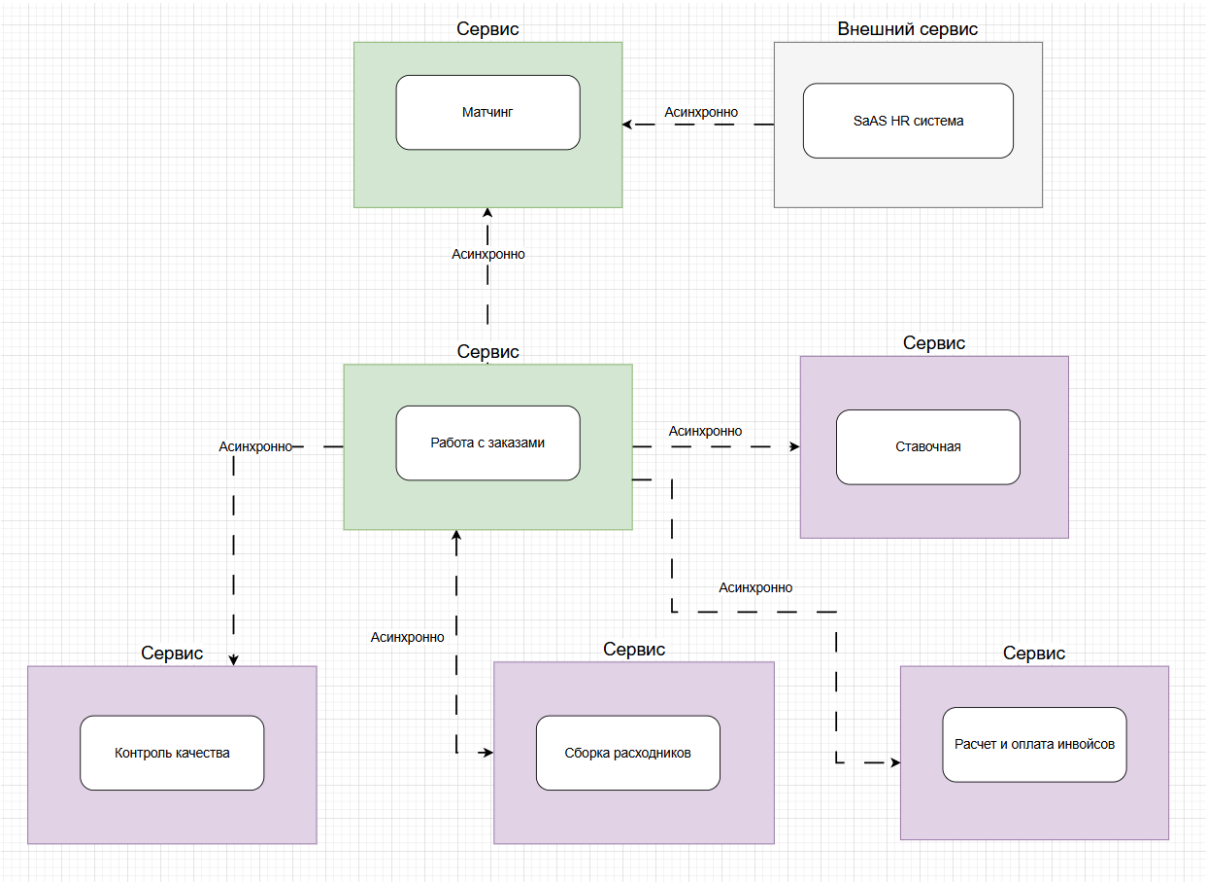
6. **Extensibility / Evolvability (Расширяемость / Эволюционируемость)**

   Возможность адаптироваться к новым платёжным системам (для биллинга).

Тут я не успеваю сделать свою табличку с выбором и сравнением характеристик, поэтому вставлю просто картинку из урока.

	layered	modular monolith	service-based	microservices
agility	★	★★	★★★★	★★★★★
abstraction	★	★	★	★
configurability	★	★	★★	★★★
cost	★★★★★	★★★★★	★★★★	★
deployability	★	★★	★★★★	★★★★
domain part.	★	★★★★★	★★★★★	★★★★★
elasticity	★	★	★★	★★★★★
evolvability	★	★	★★★	★★★★★
fault-tolerance	★	★	★★★★	★★★★
interation	★	★	★★	★★★
interoperability	★	★	★★	★★★
maintainability	★★	★★★	★★	★★★★★
modifiability	★★	★★★	★★	★★★★★
modularity	★	★★	★★★★	★★★★★
performance	★★★	★★★	★★★	★★
reliability	★★★	★★★	★★★★	★★★★
scalability	★	★	★★★	★★★★★
security	★★	★★	★★★	★★★★★
simplicity	★★★★★	★★★★★	★★★	★
testability	★★	★★	★★★★	★★★★
workflow	★	★	★	★

Для реализации я бы выбрал микросервисную архитектуру с асинхронной коммуникацией, судя по требованиям на старте не планируется сильно большая нагрузка (Общая нагрузка на систему не будет превышать 10 заказов в день и 100 клиентов. Воркеров будет около 20 котов.), но это все в теории, и т.к. нет особых требований по стоимости разработки, я выбрал именно микросервисы. Да и гибкость их лучше чем у service-based, что нам важно для низкого TTM.


Сервис
Матчинг
Асинхронно
Внешний сервис
SaAS HR система
Асинхронно
Сервис
Работа с заказами
Асинхронно
Сервис
Ставочная
Асинхронно
Асинхронно
Асинхронно
Сервис
Контроль качества
Сервис
Сборка расходников
Сервис
Расчет и оплата инвойсов