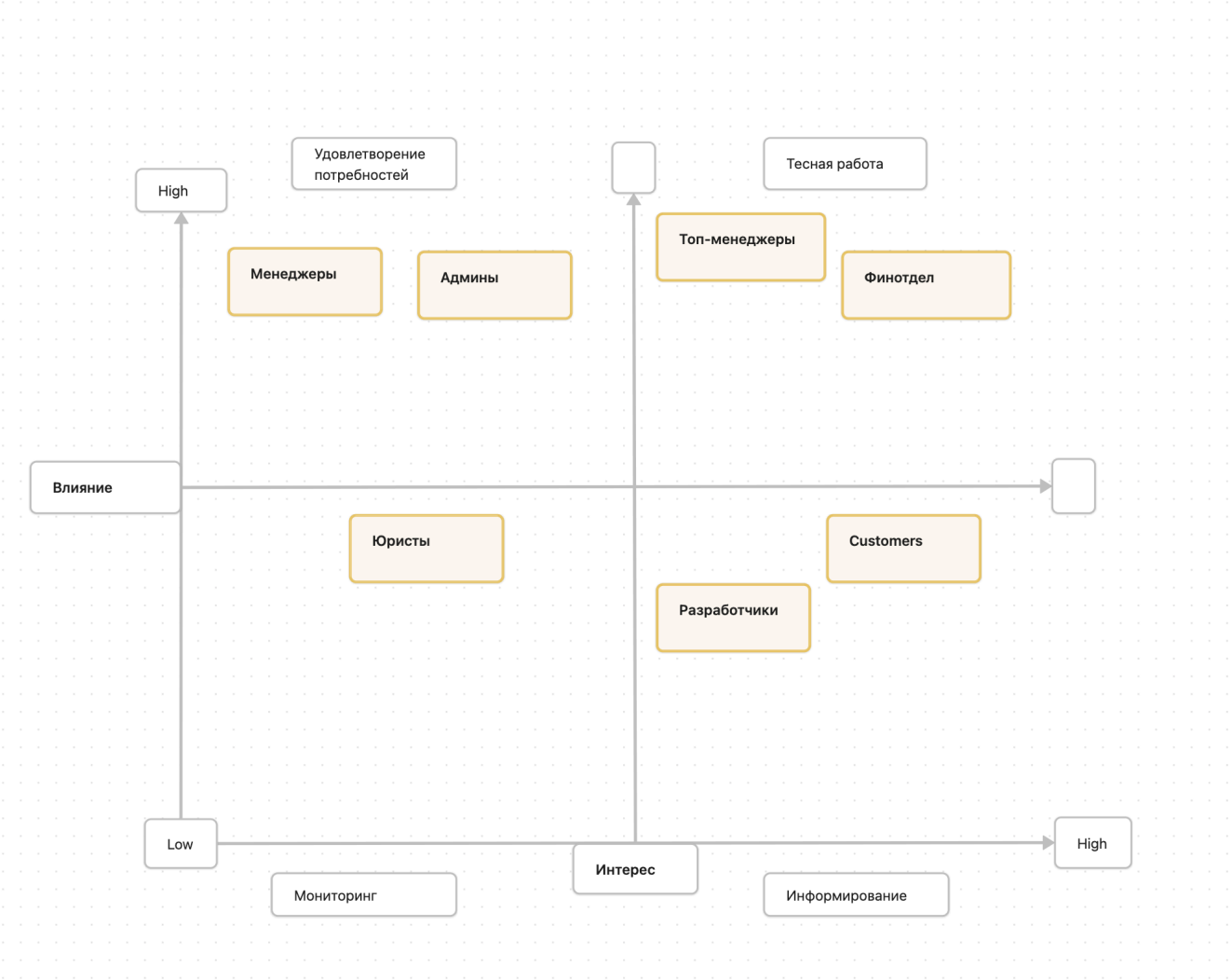
AC - Домашнее задание №3

Группы стейкхолдеров и их консёрны



Галочки - что планируется учитывать

Тесная работа

Топ-менеджмент

- [x] Agility, Deployability, Testability - Low TTM
- [x] повышенные Agility, Deployability, Testability для Candidates Testing (scoring на языке топов)

Финотдел

- [x] Securability, Consistency для Customers Accounting, Workers Accounting

Разработчики

Исходя из требований [US-300] - [US-303] становится понятно, что это один из ключевых стейкхолдеров (аналог котов-датасаентистов из HCB)

- [x] повышенные Agility, Deployability, Testability для Task Matching ([US-300])

- [ ] Maintainability на все модули системы

Удовлетворение потребностей

Менеджеры

- [x] Scalability для Task Tracking - (10 заказов/мин теперь) **Админы**
- [ ] Availability на все модули системы
- [ ] Maintainability

Информирование

Клиенты

- [x] Availability, Performance на Task Tracking
- [ ] *Usability, Consistency, Simplicity - про это явно ничего не сказано, возможно это скрытые требования (вылезут по мере эксплуатации системы)*

Мониторинг

Юристы

- [x] Securability - на модули, затрагивающие финансы (CatFinCompliance) и пользовательские данные (соответствие правовым нормам) - Customers Accounting, Workers Accounting, Task Matching (личные данные воркеров)

Кого могли забыть

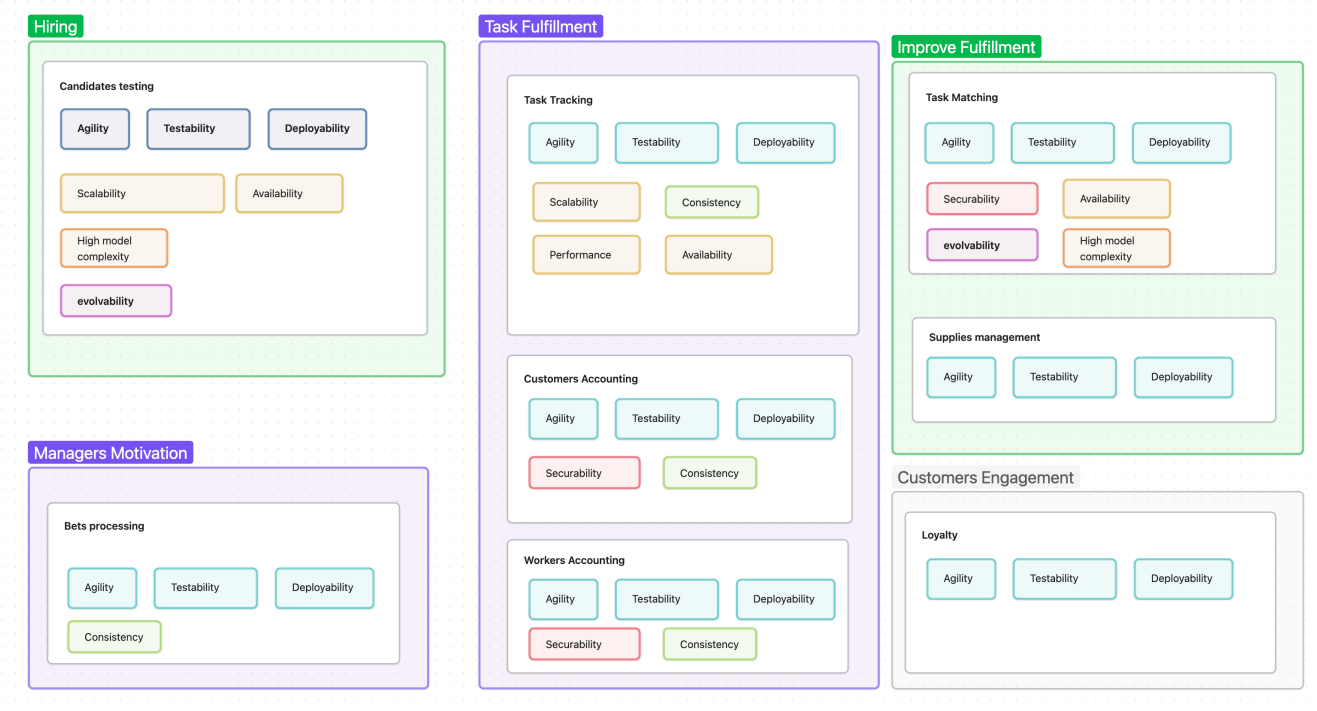
(в скобках квадрант на чарте)

- **Коты-снабжецы** (УП) - могут что-то знать про закупку расходников, могут быть "боли" в процессе снаряжения конкретных заказов расходниками, "подводные камни" во взаимодействии с пекарней - на текущей момент я не смог выявить особые требования к характеристикам Supplies Management - эти стейкхолдеры могли бы навести на них
- **Коты-контроллёры** (И), занимающиеся исследованиями отмененных заявок. Это те же менеджеры, что участвуют в тотализаторе и контролирующие ход выполнения заявки?
- **Коты-рекрутёры** (УП) - нанимающие воркеров - это всё те же менеджеры что и выше? Как-то много обязанностей для одной рабочей единицы. На первых порах наверное это может сработать. Но в дальнейшем скорее всего разделятся по трём направлениям минимум (найм/скоринг, контроль качества, менеджмент заявок). С другой стороны, их консёрны скорее всего лежали бы в области Candidates Testing, которая уже в пристальном внимании топ-менеджмента, а у них более высокий приоритет

Ограничения по проекту

- о Тотализаторе должны знать **только** менеджеры (топы и разрабы не знают тоже)
- Хранение данных в соответствии с CatFinCompliance, а также требования по мониторингу там же
- Все выплаты только через Golden Gate
- Преимущественно in-house разработка (исключение: интеграции с Golden Gate и с Пекарней Печенек)

- Бесплатная инфраструктура -> можно позволить себе более ресурсоемкие решения (распределённые системы)
- [x] Scalability для Candidates Testing - угроза DDoS от конкурентов
- [x] Availability для Task Matching - система гарантировано выделяет воркера под задачу при обращении ([US-302])
- [x] Consistency для Task Tracking, Customer Accounting, Workers Accounting - показывать актуальные списки заявок и их статусы, выставленных счетов и выплат



(Крупнее см. файл MCF_ComponentsProperties.png)

Выбор архитектурного стиля

Распределённый стиль для всей системы в целом - либо микросервисы, либо service-based

Отдельные сервисы:

Task Matching

- Pipeline - требование по набору последовательных шагов в [US-300], map-reduce like
- сложная модель
- повышенные требования по доступности (Task Tracking не будет работать без нее)
- но повышенные требования к конфидециальности данных (правовые нормы), а также evolvability мешают объединению с Task Tracking

Candidates Testing

- Microkernel - скоринг планируется применять к различных системам в будущем (вводные от топ-менеджмента), следовательно несмотря на общий алгоритм ожидаются частности, которые можно реализовать в отдельных плагинах (под каждую систему)
- повышенные требования к масштабируемости системы (угроза DDoS, да и в целом большой поток заявок на устройство на работу)

Accounting (both Customers and Workers)

- идентичные характеристики
- в частности требования к безопасности и сохранности данных
- работа с деньгами сосредоточена в одном сервисе
- Layred Monolith - есть и презентационная функциональность (показывать списки счетов) и бизнес-логика + работа с хранилищем данных

Task Tracking

- Layred Monolith - есть и презентационная функциональность (показывать списки тасок) и бизнес-логика + работа с хранилищем данных

Supplies Management + Loyalty (интеграция с печеньками)

- схожие характеристики (нет вводных кроме общих)
- вся лояльность пока заключается в интеграции с Печеньками и они нужны только при работе кота-снабженца - кажется разумным разместить это все в одном сервисе
- Modular Monolith

Взаимодействие Task Tracking - Supplies Management

- напрашивается event-driven стиль из-за пошагового характера рабочего процесса

Bets processing

- Layred Monolith

Выбор БД

- **Accounting** - отдельная РСУБД, CatFinCompliance, consistency
- **Task Matching** - в требованиях четко не прописан характер данных, но складывается, что это всё-таки структурированные данные (характеристики эталонных и регулярных образцов + история прошлых заказов - даже если не прямая репликация из Task Tracking, то точно ETL подгоняющий результаты под нужную форму), возможно имеет смысл рассмотреть документоориентированную базу данных. Не у каждого из образцов есть все характеристики, в процессе развития системы могут выявляться новые, а старые упраздняться - в РСУБД пришлось бы мутить миграции схемы и а то и данных.
- **Task Tracking** - отдельная РСУБД (сделать разделяемую с Supplies Management мешают повышенные требования к ТТ + единственное пересечение в данных - таски, это можно так или иначе экспортировать в SM)
- **Supplies Management** - отдельная БД, жестких ограничений тут не вижу, может подойти как РСУБД, так и документоориентированная. Сам склоняюсь больше к РСУБД. Нет такой вариативности в данных как в ТМ, схема будет редко меняться. Данные формализованы.
- **Bets Processing** - простая РСУБД, для Key-Value хранилища модель данных сложновата - Bet, Task, Manager
- **Candidates Testing** - самый сложный для меня пункт. С одной стороны формализованый процесс проведения тестирования, наборы тестов и их конфигурация наводят на мысль о РСУБД. С другой стороны, необходимость адаптирования скоринга под разные системы грозит проблемами при жесткой схемы БД, что подталкивает к документоориентированной

БД. Высокие требования к производительности склоняют к выбору последней (но лишь чуть, при граммотном тюнинге современных РСУБД можно добиться и от них высокой производительности)

Стиль коммуникаций

- **Candidates Testing -> Task Tracking, Task Matching** - event-driven, асинхронный стриминг новых воркеров. Не требует моментального ответа. Сообщает о появлении воркера в системе
- **Task Tracking -> Task Matching** - event-driven, сообщает о появлении новой заявки. Асинхронный, чтобы не блокировать работу TT пока Matching подбирает воркера. А также чтобы распараллелить процесс назначения.
- **Task Matching -> Task Tracking, Bets Processing** - event-driven, сообщает об успешном присвоении заявке исполнителя
- **Task Tracking -> Supplies Management** - event-driven, сообщает о появлении взятой в работу заявки в системе
- **Supplies Management -> Loyalty** - синхронный вызов (в рамках модульного монолита) и request-response - в рамках интеграции Loyalty с API пекарни
- **Supplies Management -> Task Tracking** - event-driven, сообщает о собранном снаряжении для заявки
- **Task Tracking -> Bets Processing, Accountancy** - event-driven, сообщает о завершении заявки (выполнена или отменена)
- **Accountancy -> Golden Gate** - request-response

Вообще на основе описанных коммуникаций четко вырисовывается то, что у Бена Стопфорда в книге "Проектирование событийно-ориентированных систем" называется моделью событийного сотрудничества

Фитнес-функции

Общее

- code coverage 95% (придерживался у себя на 2 проектах, никогда не подводило),
- основные сценарии желательно покрыты e2e - чтобы комплексно проверить работу нескольких модулей системы
- проверка периодичности релиза не реже раза в месяц - соблюдение общего TTM от топ-менеджмента
- проверка настроенного расписания бэкапов БД (особенно Accountancy и Task Tracking)
- периодическое прохождение процедуры восстановления из бэкапов системы после аварийного падения (консёрн топов о сохранности данных, консёрн админов)
- сканирование исходников на наличие захардкоженных (или переменных окружения) с именами хостов сервисов - сервисы не должны знать друг о друге, только об очередях. Исключение - интеграции с упомянутыми внешними системами

Candidates Testing

- SLI-метрики для сайта с заявками кандидатов (мониторинг как система реагирует на резко возросшую нагрузку)

- проведение нагрузочного тестирования перед релизом - валидировать способность держать DDoS-атаки
- проверка периодичности релиза не реже раза в неделю - соблюдение TTM по скорингу от топ-менеджмента
- согласование на заведение новых очередей\топиков
- сверка с schema-registry (где только события, описанные в разделе "Стиль коммуникации" + добавляемые по предыдущему пункту)

Task Matching

- проверка, что структура сервиса соответствует идее pipelin'a - поддержание возможности свободного комбинирования шагов по подбору воркера на заявку
- ручная проверка отсутствия жесткой схемы в БД - если возникнет желание использовать РСУБД,

Task Tracking

- проведение нагрузочного тестирования перед релизом - убедиться, что соблюдается условие по 10 заявок в мин (консёрн менеджеров)

Accountancy

- проверка изолированности БД Аккаунтинга от БД Таск Трекинга - у разработчиков может возникнуть искушение реализовать service-based подход вместо event-driven и держать счета в той же БД, что и таски
- проверка авторизованного доступа к сайту Аккаунтинга (только воркеры, клиенты и сотрудники финотдела)

ADR

- TBD - не хватило ни времени, ни сил. Позже опишу