













Получите свой \*\*
Минимальный жизнеспособный продукт \*\*, чтобы проверить свою концепцию.







#### [MVP]

\*\* периметр \*\* вашего MVP должен быть \*\* уменьшен \*\*, позволяя вам продавать ваш продукт.

Сделайте ставку на \*\* Early Adopter \*\* и получите максимум \*\* обратную связь \*\*.

Ваш MVP развертывается и может использоваться в \*\* производстве \*\*.











\*\* Fail \*\* fast \*\* \*\* быстро.





## [ СБОЙ-FAST ]

\*\* Быстро испытать \*\* решение (несколько недель), собрать \*\* обратную связь \*\* от своих пользователей и узнать о своих ошибках.

Не бойтесь все менять \*\*.

Не забывайте, \*\* вы потерпите неудачу! \*\*











\*\* K \*\* eep \*\* I \*\* t \*\* S \*\* imple и \*\* S \*\* tupid.

 Почему сложнее, когда это может быть просто? \*





## [ПОЦЕЛУЙ]

\*\* Избегайте чрезмерной инженерии \*\*, если достаточно «бумажной» модели или формы Google для проверки вашей концепции, не ходите дальше.

Оставайтесь просто! Технически и функционально.









## ПОРА ТОРГОВАТЬ [ ЭФФЕКТИВНОСТЬ]

Указать меньше, \*\* расширить подробнее \*\*.





#### [ ЭФФЕКТИВНОСТЬ ]

Ограничьте свои спецификации до нужных предметов, \* сосредоточьтесь на «чем» \*\*, а не на «как».

Продукт должен быть максимально самодокументированным \*\*.

Документация должна быть версирована так же, как и код.











Систематически изучать решения SaaS \*\*.







#### [SAAS]

\*\* Решения SaaS \*\* являются устойчивыми и экономически эффективными \*\*.

В некоторых случаях \*\* SaaS \*\* может \*\* ускорить \*\* реализацию \*\* MVP \*\*.

Подумайте об экономическом видении \*\* в отношении альтернатив с точки зрения \*\* общей стоимости \*\* (\*\* TCO \*\*: \*\* T \*\* otal \*\* C \*\* ost of \*\* O \*\* wnership ) и не только с точки зрения стоимости ли







\*\* Основной бизнес \*\* не должен быть помехой для строительства новых услуг и приложений.





## [ СЕРДЦЕ БИЗНЕСА ]

Темпы эволюции и доставки основного бизнеса должны быть \*\* совместимы с гибкостью \*\* услуг, которые его потребляют.

Основной бизнес должен \*\* предоставлять услуги \*\*.

Основной бизнес должен принять принцип \*\* Event-Driven \*\*, он сообщает о действиях руководства в форме событий.









\*\* Развертывание в производстве \*\* является не-событием.







# [ НЕПРЕРЫВНОЕ РАЗВЕРТЫВАНИЕ ]

Воспользуйтесь \*\* непрерывным развертыванием \*\*, чтобы адаптировать \*\* \*\* производство \*\* к \*\* бизнес-требованиям \*\*, а не наоборот.

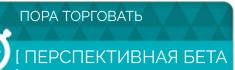
\*\* Развертывания \*\* в разных средах, вплоть до \*\* производства \*\*, должны быть \*\* автоматизированы \*\* и \*\* частыми \*\*.











Постоянный бета-метод позволяет привлекать ваших пользователей к процессу разработки.







#### [ ПЕРСПЕКТИВНАЯ БЕТА ]

Не стесняйтесь использовать принцип вечной беты, в котором \*\* пользователи участвуют в разработке \*\*.

Термин «вечная бета» относится к приложению, разработанному точно в срок, \*\* постоянно развивается \*\*, а не к неполному продукту.

















#### ПОЛЬЗОВАТЕЛЬСКИЙ ОПЫТ

#### [ВОСПРИЯТИЕ]



Опыт, воспринимаемый пользователем, является основополагающим.

\* эргономика \*\* не подлежит обсуждению.







#### [ВОСПРИЯТИЕ]

Не пренебрегайте работой \*\* дизайнеров UX \*\*, это имеет фундаментальное значение при разработке приложения.

Интеграция \*\* отзывов \*\* ваших пользователей, это важно.









#### ПОЛЬЗОВАТЕЛЬСКИЙ ОПЫТ

## [ ПРЕДСТАВЛЕНИЕ ]

Используйте \*\* мощные интерфейсы \*\* для внутреннего и внешнего использования.







### [ПРЕДСТАВЛЕНИЕ]

Интерфейсы ориентированы на \*\* эффективность \*\*.

\*\* производительность \*\*
интерфейса экономит время \*\*,
увеличивает удовлетворенность
пользователей \*\* и,
следовательно, \*\* сохраняет их
разочарование \*\*.









### [ ПЕРВАЯ МОБИЛЬНАЯ ]

Примите стратегию \*\* Mobile First \*\*.





#### ГПЕРВАЯ МОБИЛЬНАЯ І

Мобильные устройства являются
\*\* самой важной \*\* частью \*\* рынка
\*\*.

Мышление, мобильное, думает о \*\* существенном \*\*.

\*\* Ревизионный дизайн \*\* является нормой, он является источником сбережений (\*\* MVP \*\*).









### [ OMNI-КАНАЛ ]

Адаптация к использованию, \*\* omniканал \*\* является нормой.







### LOMNI-КАНАЛ 1

Многоканальный подход предоставляет пользователю \*\* объединенный опыт \*\* (пример: Netflix).

Различные \*\* каналы \*\* \*\* синхронизированы \*\* и \*\* когерентные \*\* (в отличие от пакетных процессов).

Все участники (клиенты, консультанты) получают доступ к одной и той же информации.









## [SELF-DATA]

\*\* \*\* пользователи \*\* являются \*\* владельцами \*\* своих данных и их курса.







#### SELF-DATA 1

Оставьте \*\* индивидуумов \*\*, в любое время \*\*, \*\* \*\* на \*\* своих \*\* личных \*\* данных.

Установите \*\* уверенность \*\*, позволяя отслеживать и контролировать пользователей в режиме реального времени.

\*\* подсистемы \*\* должны отвечать тем же требованиям.











Клиентские отношения должны быть унифицированы и контекстуализированы с помощью гибкого, унифицированного и управляемого событиями

CRM / SFA. \*



#### I CRM / SFA 1

Выбирайте \*\* CRM , который управляет как отношениями с клиентами, так и лидерами продаж ( SFA \*\*: \*\* S \*\* ales \*\* F \*\* orce \*\* A \*\* utomation ).

\*\* CRM \*\* должен быть \*\* открыт \*\* для новых возможностей.

\*\* CRM \*\* производит \*\* события \*\*, соответствующие действиям руководства, вписывающимся в логику \*\* платформы Event-Driven









# [БОЛЬШИЕ ДАННЫЕ]

Платформа Big Data
позволяет централизовать

\*\* и обрабатывать
пользовательские данные,
чтобы наилучшим
образом обслуживать \*\*
путешествие \*\*.





## [ БОЛЬШИЕ ДАННЫЕ ]

Централизовать \*\* Группу Maif \*\*, \*\* Партнер \*\* и \*\* Данные поставщика \*\* в логике \*\* пути \*\*.

«Подготовка данных» и обработка могут \*\* консолидировать \*\* данные.

\*\* Команды «Большие данные» сотрудничают \*\* с функциональными группами для обеспечения \*\* данных \*\* управления.









### [ DESKTOP ]

Рабочая станция адаптирована и адаптируется к \*\* использует \*\* и \*\* современные каналы \*\*.







#### [DESKTOP]

Примите \*\* федерацию идентичности \*\* для унифицированного опыта.

\*\* \*\* \*\* \*\* позволяет \*\* предлагать \*\* обзор \*\*, он не заменяет приложения.

Рабочая станция должна быть \*\* мобильной \*\*, \*\* многоканальной \*\* и \*\* стандартной \*\*, чтобы разрешить открытие в \*\* расширенном предприятии \*\*.









## [ АВТОРЫ ]

Не забывайте, что ваши партнеры \*\* используют современные приложения дома в UX.







#### [АВТОРЫ]

Лечите \*\* всех своих пользователей как «клиентов» \*\*: пользователи Интернета, менеджеры, операторы, разработчики и т. Д.

Не стоит недооценивать \*\* усилие UX \*\* для внедрения приложений управления внутренним использованием.









# [ВСЕ ИЗМЕРЕНИЕ]



Все, что можно измерить, должно быть.

\*\* Без меры все это только мнение \*\*.







#### [ВСЕ ИЗМЕРЕНИЕ]

Подумайте о показателях во время \*\* разработки \*\* приложения. \*\* журналы \*\* должны иметь \*\* бизнес, а также технические \*\* измерения.

Не пренебрегайте характеристиками \*\* \*\*, они фундаментальны.

Функциональная группа обеспечивает \*\* операцию \*\*: она отвечает за использование \*\* приложения \*\*-







## [ Тестирование А / В ]

\*\* Тестирование A / B \*\*

экономит ваше время,

позволяя решить \*\*

обратную связь \*\*.







## [Тестирование А / В ]

Вместо того, чтобы произвольно решать два решения, не стесняйтесь настраивать \*\* A / В тестирование \*\*.

Этот шаблон состоит из представления \*\* двух разных версий \*\* одного и того же приложения и выбора одного из них на основе \*\* объективных мер \*\* активности пользователя.











\* Учитывайте деградацию \*\*, а не прерывание обслуживания в случае сбоя.







#### [ N3HOC ]

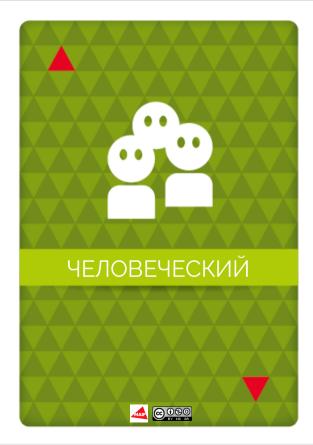
В \*\* сбое \*\* одной из подсистем \*\* ненормальная \*\* версия службы должна \*\* считаться \*\* в первую очередь, а не прерыванием.

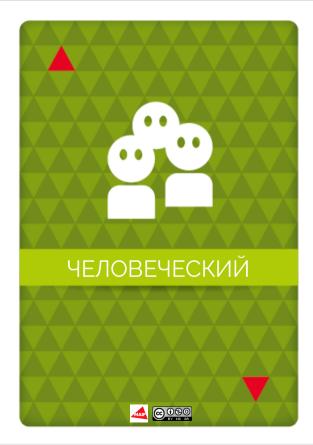
С \*\* Автоматическими выключателями \*\*, \*\* изолируйте пробой \*\*, чтобы \*\* избегать \*\* его \*\* удара \*\* и \*\* распространения \*\* по всей \*\* системе \*\*.



















Команда организована вокруг \*\* продукта \*\* или \*\* услуги \*\*, оказанной.







### ЧЕЛОВЕЧЕСКИЙ

### [ KOMMEHTAPИЙ FEATURE I

Команды - \*\* Feature Feature \*\*, организованные вокруг согласованного функционального набора и состоящие из всех \*\* навыков \*\*, необходимых для этого набора.

Hапример: Business Expert + Web Developer + Java Developer + Architect + DBA + Operational.

\*\* Ответственность \*\* является \*\* коллективной \*\*, функциональная группа облад тараномочиями,









# [ КОМАНДА 2-ПИЦЦЫ ]

Ограничьте \*\* размер функциональных групп \*\* (от 5 до 12 человек).





## [КОМАНДА 2-ПИЦЦЫ]

Ограничьте размер функциональной группы: \*\* от 5 до 12 человек \*\*.

Ниже 5 она слишком чувствительна к внешним событиям и испытывает недостаток в творчестве. Выше 12, он теряет производительность.

Термин \*\* «2-Pizza Team» \*\*
указывает, что размер Feature
Теат не должен превышать
количество людей, которым
можно корми









Ставка на универсальных людей, которые \*\* умеют делать \*\* и которые \*\* любят делать \*\*.





## [ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ARTISAN ]

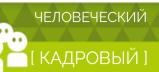
Самой важной является \*\* культура развития \*\*, \*\* масштабируемость \*\* и \*\* адаптивность \*\*.

Рекрутинг \*\* разработчиков программного обеспечения и разработчиков полных стеков \*\*, они приносят реальную добавленную стоимость благодаря их ноу-хау и их общему видению.









Будьте \*\* привлекательны
\*\*, чтобы набрать \*\* лучший
\*\*





## [КАДРОВЫЙ]

Предложить рабочие режимы, адаптированные к сотрудникам: \*\* мобильность \*\*, \*\* домашняя работа \*\*, \*CYOD\* (Choose Your Own Device.

Дайте время для экспериментов и сделайте это \*\* в рабочее время \*\*.











Организация должна быть
\*\* двигателем сна \*\*

Накануне это часть работы.







#### [EBA]

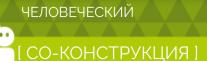
Организация должна быть \*\* дневным движком \*\*, создавая такие системы, как \*\* непрерывное образование \*\* или \*\* бизнес-университеты \*\*.

Не стесняйтесь сочетать их с другими более неформальными \*\* способами, такими как: \*\* Кодирование Dojos \*\*, \*\* Кофейные обеды \*\*, \*\* Внешние \*\* Конференции.









Преодолейте барьеры между сделками, сделайте ставку на цели \*\* конвергенции \*\*.





## [ СО-КОНСТРУКЦИЯ ]

Чтобы преодолеть барьеры между сделками, недостаточно объединить людей вокруг общего продукта в общем месте.

\*\* Гибкие подходы \*\* устраняют эти барьеры, чтобы обеспечить \*\* сближение целей \*\*.

Эти методы являются неотъемлемой частью ключей к успеху, организация является гарантом.









\*\* Методы DevOps \*\* позволяют стенам упасть между Build и Run.





## [DevOps]

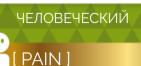
Принять \*\* DevOps \*\* для сближения \*\* Dev \*\* и \*\* Ops \*\* в направлении общей цели: \*\* служить организации \*\*.

\*\* Торги остаются разными \*\*! DevOps не означает, что один и тот же человек выполняет задачи Dev и Ops. \*\* Разработчики \*\* и \*\* Операционные \*\* обязаны \*\* сотрудничать \*\*, чтобы \*\* извлечь выгоду \*\* из \*\* \*\* навыков \*\* и улучшить \*\* эмпатию \*\*.











\*\* Жесткие задания \*\* выполняются \*\* Группой разработчиков \*\*.

> Затем следует автоматизация.







## [PAIN]

В традиционной организации \*\* отсутствие понимания \*\* между командами обычно связано с расстоянием и \*\* отсутствием связи \*\*.

- \*\* Члены функциональной группы \*\* \*\* несут ответственность \*\* и \*\* солидарны \*\* для всех задач.
- \*\* Боли \*\* ключевой фактор в \*\* Непрерывном улучшении \*\*.









Сервисные центры трудно согласовать с \*\* коллективным обязательством \*\*.





### <u>ЧЕЛОВЕЧЕСКИЙ</u>

### I CDS I

Функциональные группы построены на принципах, которые в значительной степени зависят от сотрудничества \*\* и \*\* коллективного участия \*\*.

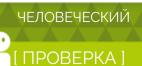
Сервисные центры продвигаются к рационализации и консолидации ИТ по бизнесу, что противоречит этому понятию коллективной ответственности.











Организация имеет \*\* роль валидации \*\*, не будучи догматической.





### [ПРОВЕРКА]

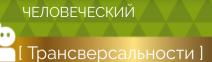
Убедитесь, что организация сохраняет свою \*\* роль проверки \*\* в инструментах и целях использования. В частности, в \*\* инструментах, влияющих на наследие \*\* (пример: управление исходным кодом).

\*\* Предоставить \*\*
Функциональные группы с \*\*
означает \*\* для поддержки их
выбора.

Не будьте \*\* догматичны \*\* и обязательно 🗽 🍘 🎎 Ч









Ожидается, что группы функций \*\* будут общаться \*\* и поделиться своим \*\* опытом \*\* и \*\* навыками \*\*.







## [Трансверсальности]

He создавайте препятствий между \*\* Feature Groups \*\*.

Настройте \*\* организацию \*\* и \*\* ловкость \*\*, необходимую для функциональных групп, чтобы общаться друг с другом и делиться своими навыками и опытом.

Организация трансверсальности в \*\* Spotify \*\* (Tribes, Chapters and Guilds) является красноречивым примером.





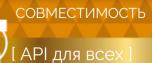














\*\* АРІ для всех видов использования \*\*: внутренние, клиенты и партнеры, публичные.





### [ АРІ для всех ]

Откройте свою организацию для новых пользователей и новых клиентов с \*\* Public API \*\*.

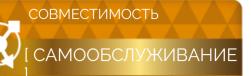
В \*\* коммерческих \*\*
партнерствах \*\*, \*\* клиенты \*\* как
\*\* провайдеры \*\*, АРІ являются
стандартным обменным
форматом.

\*\* API \*\* также предназначены для использования \*\* для внутреннего использования \*\* организации.











Использование API должно быть \*\* простым \*\* и \*\* быстрым \*\*.





## [ САМООБСЛУЖИВАНИЕ ]

Использование API должно быть как можно более простым. Подумайте о \*\* опыте разработчиков \*\*.

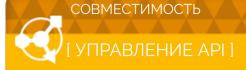
Лучшим решением для проверки адекватности с необходимостью является \*\* быстрое тестирование API \*\*: достаточно нескольких минут!

Платформа должна предлагать \*\* графический интерфейс \*\* для простого тестирования API.









Использование API должно контролироваться \*\* и \*\* контролироваться \*\*.







## [УПРАВЛЕНИЕ АРІ]

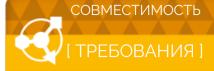
Внедрите решение управления АРІ для управления \*\* квотами \*\*, \*\* дросселированием \*\*, \*\* аутентификацией \*\* и \*\* протоколированием \*\*.

Сбор показателей для управления \*\* мониторингом \*\*, \*\* фильтрацией \*\* и \*\* отчетностью \*\*.









Задайте \*\* требования \*\* для \*\* внешних систем и услуг \*\*, встроенных в платформу.





## [ТРЕБОВАНИЯ]

Требовать \*\* внешние системы \*\* для удовлетворения тех же \*\* требований \*\* как \*\* внутренние системы \*\*.

Внешние системы должны публиковать \*\* события \*\* и допускать \*\* технический \*\* мониторинг.

В случае, когда данные внешних систем должны быть интегрированы, \*\* полная \*\* синхронизация должна быть \*\* возможной \*\* 🚵 🔞 🔞







Архитектура должна считаться \*\* мультиарендатором \*\*.







### СОВМЕСТИМОСТЬ

## [ МНОГОКВАРТИРНЫЕ ДОМА ]

Даже если белая метка не рассматривается в базе, настройте архитектуру с несколькими арендаторами. Ваша \*\* первоначальная \*\* заявка является первой \*\* удержанием \*\*.

Подумайте о \*\* многофункциональном экземпляре \*\* системы с самого начала.







## COBMECTUMOCTS

Системы должны быть конфигурируемы \*\* \*\*.







## [SETTING]

\*\* Языки, валюты, бизнес-правила, профили безопасности \*\* должны быть простыми в установке.

Остерегайтесь \*\* гипер-ротации \*\*, это часто бесполезно и \*\* источник затрат \*\*.

\*\* \*\* установка \*\* должна быть масштабируемой \*\* и быстро при необходимости.













Создавайте гибкие и общие системы, используя \*\* функцию flipping \*\*.





### СОВМЕСТИМОСТЬ

## [ ХАРАКТЕРИСТИКА ПЛАНИРОВАНИЯ ]

\*\* функция flipping \*\* заключается в разработке приложения в виде набора \*\* функций \*\*, которые могут быть \*\* включены \*\* или \*\* отключены \*\* горячие, \*\* production \*\*.

В приложении \*\* multi-tenant \*\* функция flipping позволяет вам настроить \*\* сторонников.

Le функция flipping \*\* simplifie l'A / В тестирование \*\*.











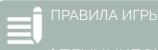




# ПРАВИЛА ИГРЫ [ ТЕХНИЧЕСКИЕ ВЫБОРЫ]







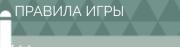
## [ ТЕХНИЧЕСКИЕ ВЫБОРЫ

Функциональная группа должна действовать \*\* ответственно \*\*, чтобы определить варианты, которые влияют на нее исключительно, и варианты, влияющие на организацию.

\*\* выбор \*\*, который \*\* превышает объем \*\* функциональной группы (например, лицензия, нечастый язык программирования), должен быть \*\* подтвержден \*\* организацией или процессом конвергенци







\*\* Правильный инструмент \*\* для \*\* хорошего использования \*\* является источником экономии.





## [ Хорошее использование

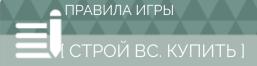
\*\* Плохой инструмент \*\*, наложенный на каждого, является \*\* риском \*\*. \*\* неправильное использование \*\* хорошего инструмента может иметь \*\* очень \*\* ущерб \*\* последствия \*\*. Например, опасные методы Agile опасны.

\*\* Инструменты \*\* должны быть поставлены под сомнение \*\*.

\*\* Excel \*\* часто является рациональны выболаюм \*\*, но это







Привилегия \*\* Сборка \*\* для основного бизнеса.

Рассмотрим \*\* Купить \*\* для остальных, в каждом конкретном случае.



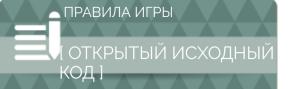
## [СТРОЙ ВС. КУПИТЬ]

Чем больше инструмент использует \*\* функцию, отличающую \*\* функцию для организации, тем больше она должна быть построена \*\*. Основной бизнес должен позволять \*\* специфика \*\* и \*\* быстро и часто адаптироваться Некоторые \*\* программные пакеты \*\* \*\* иногда адаптируются <sup>\*\*</sup> к этой потребности.

Для \*\* остальное \*\*: SaaS, Open Source, Build или Owner должны быть изучены каждом случае







\*\* Максимально используйте Open Source

Альтернативные варианты должны поддерживаться.





## [ ОТКРЫТЫЙ ИСХОДНЫЙ КОД ]

\*\* Запатентованные решения \*\* являются \*\* риском \*\* для организации, которая должна иметь возможность возобновить обслуживание, если это необходимо.

Существует несколько проприетарных инструментов, которые не имеют \*\* альтернатив с открытым исходным кодом \*\*.

Организация \*\* извлекает выгоду 
\*\* из \*\* Сообшатваерткрытым 
\*\*









Разработка \*\* автономных \*\* и \*\* слабо связанных \*\* услуг.





## [ MICRO-УСЛУГИ ]

\*\* слабая связь \*\* должна быть нормой.

Каждая микросервис имеет \*\* четко определенный интерфейс \*\*

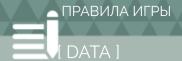
Этот \*\* интерфейс \*\* определяет \*\* ссылку \*\* между \*\* микросервисами \*\*.

\*\* Domain Driven Design \*\*
позволяет, особенно с \*\*
Ограниченными контекстами \*\*,
предвидеть эту проблему.









У каждой службы есть своя система хранения данных \*\*.



# ПРАВИЛА ИГРЫ [ DATA ]

\*\* \*\* Хранилище данных \*\* должно быть \*\* соединено \*\* только с \*\* одним микросервисом \*\*.

\*\* Доступ к данным \*\* с одного микросервиса на другой осуществляется \*\* исключительно через его интерфейс \*\*.

Этот дизайн подразумевает \*\* согласованность во времени \*\* по всей платформе. Он должен быть воспринят на всех уровнях \*\*, включая UX.









Каждое микросервис должно иметь разумный функциональный периметр, который \*\*
«вписывается в голову» \*\*



### [ ОБЛАСТЬ ПРИМЕНЕНИЯ

Микросервис предлагает \*\* разумное количество функций \*\*.

\*\* Не стесняйтесь сокращать \*\* микросервис, когда он начинает расти.

Услуга разумного размера позволяет \*\* спокойно рассмотреть \*\* переписывание \*\*, если возникнет такая необходимость.









\*\* Reactive Manifesto \*\* открывает путь к дизайну реактивных архитектур.



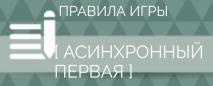
### [ отзывчивый ]

\*\* Отзывчивое \*\*
программирование фокусируется
на потоке данных и
распространении изменений. Он
основан на схеме «\*\*
Наблюдатель », противоречащей
подходу « Итератор \*\*», более
традиционному.

Реактивный манифест устанавливает основные оси: \*\* доступность \*\* и скорость, \*\* устойчивость к ошибкам, \*\* гибкость \*\*, \*\* эластичность \*\* и \*\* ориентация \*\*











### [ АСИНХРОННЫЙ ПЕРВАЯ ]

Сначала обмен между приложениями должен быть \*\* асинхронным \*\*.

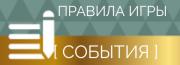
Асинхронные обмены естественно \*\* позволяют \*\* слабую \*\* связь, \*\* изоляцию \*\* и \*\* управление потоком \*\* (\*\* обратное давление \*\*).

\*\* синхронная связь \*\* следует рассматривать только \*\*, когда действие требует этого \*\*.











Информационная система должна быть ориентирована на \*\* события \*\*.



Ориентация \*\* \*\* позволяет поддерживать реализацию таких подходов, как \*\* С \*\* ommand \*\* Q \*\* uery \*\* R \*\* Ответственность \*\* S \*\* egregation (\*\* CQRS \*\* ) и \*\* Event Sourcing \*\*.











Привилегия \*\* простого, надежного и мощного \*\* брокер-сообщения для «умной трубы».

## **=**ПРАВИЛА ИГРЬ

### [БРОКЕР СООБЩЕНИЯ]

\*\* ESB \*\* показал \*\* пределы \*\*: \*\*
масштабируемое обслуживание
\*\* является критическим \*\*, как с \*\*
технической \*\*, так и с \*\*
организационной \*\* точки зрения.

\*\* \*\* Брокер \*\* сообщения, такие как \*\* Kafka \*\* предлагают простые \*\*, \*\* прочные \*\* и \*\* упругие \*\* решения.

\*\* Умные конечные точки \*\* и \*\*
Простые трубы \*\* - это
архитектура, которая работает в
масштабе: эт







\*\* Полная синхронизация

\*\* системы должна

учитываться, как только

она \*\* разработана \*\*.

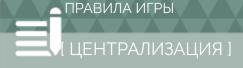


Если \*\* синхронизация \*\* между двумя системами обеспечивается \*\* потоком событий \*\*, общая \*\* пересинхронизация \*\* этих систем должна быть запланирована \*\* во время разработки \*\*.

Автоматический \*\* \*\* \*\* синхронный аудит \*\* (пример: с помощью выборок) позволяет \*\* измерять \*\* и \*\* обнаруживать \*\* любые возможные \*\* ошибки синхронизации \*\*.







\*\* Конфигурация \*\* услуг \*\* централизована \*\*, их \*\* открытие \*\* обеспечивается \*\* справочником \*\*.





### [ ЦЕНТРАЛИЗАЦИЯ ]

\*\* \*\* \*\* \*\* \*\* \*\* \*\* централизованная \*

\*\* для всех \*\* окружения \*\*.

Центральный справочник \*\* обеспечивает \*\* динамическое открытие \*\* \*\* микро-услуг \*\*.

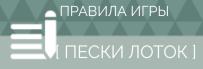
\*\* \*\* глобальная \*\*

масштабируемость \*\* зависит от этого \*\* каталога \*\*.









Группы функций предоставляют \*\* изолированную среду \*\*





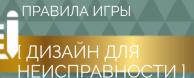
#### [ ПЕСКИ ЛОТОК ]

Группы функций поддерживают
среду ** sandbox ** (текущая
версия и предстоящий выпуск),
чтобы другие ** команды
расширялись **.
В ** некоторых не номинальных **
случаях ** ** ** ** ** ** ** ** ** **











\* Ваша система сработает! \*\*

Создайте его так, чтобы он был терпимым.







### [ ДИЗАЙН ДЛЯ НЕИСПРАВНОСТИ ]

Ваша \*\* система не сработает , это неизбежно. Он должен быть разработан для этого ( Design For Failure \*\*).

Предсказывать \*\* избыточность \*\* на всех уровнях: \*\* аппаратное обеспечение \*\* (сеть, диск и т. Д.), \*\* приложения \*\* (несколько экземпляров приложений), \*\* географические \*\* зоны, \*\* провайдеры \*\* (пример: AWS + OVH).









Предоставьте \*\* набор инструментов \*\*, не налагайте строгие рамки.



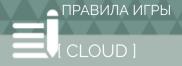
\*\* Внимание к техническим компонентам домов и поперечным \*\*! Они являются ограничительными, дорогими и сложными в обслуживании.

\*\* ускорители \*\*, \*\* набор инструментов \*\*, \*\* технические стеки \*\* могут быть объединены \*\*, \*\* бесплатно \*\* Команды функций, избегая догматического подхода.









Общественный, частный или гибридный, \*\* облако \*\* (\*\* laaS \*\* или \*\* PaaS \*\*) является стандартом для производства.





\*\* Услуги PaaS \*\* \*\* предпочтительны \*\*, \*\* простые \*\* и быстро масштабируются.

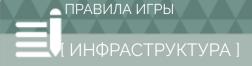
\*\* Услуги laaS \*\* позволяют решать случаи, требующие большей \*\* гибкости \*\*, но требуют более оперативной работы.

Частное облако не является традиционной средой виртуализации, она полагается на \*\* товарное оборудование \*\*.









Функциональные группы не управляют инфраструктурой, она \*\* предоставляется и поддерживается организацией \*\*.





### [ИНФРАСТРУКТУРА]

Проблемы с инфраструктурой не входят в \*\* Функциональные группы \*\*. Инфраструктура должна быть \*\* предоставлена \*\* и \*\* поддерживается \*\* \*\* кроссфункциональной \*\* услугой.









\*\* Контейнеры \*\*
обеспечивают гибкость,
необходимую для
гетерогенной оснастки.

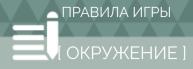


Контейнеры обеспечивают \*\* гибкость \*\*, необходимую функциональным группам для включения \*\* гетерогенной оснастки \*\* в \*\* однородном контексте \*\*.









Использование \*\*
контейнеров \*\* позволяет
преодолеть проблемы \*\*
технических условий \*\*.



### [ОКРУЖЕНИЕ]

Контейнеры (пример: \*\* Docker \*\*) позволяют \*\* освобождаться \*\* от различий в окружающей среде.

Процесс \*\* развертывания \*\* должен быть \*\* агностиком \*\* для среды.

\*\* Некоторые компоненты \*\*, такие как базы данных, не должны развертываться в контейнерах. Их развертывание по-прежнему автоматизировано.









Меры должны быть \*\* централизованы \*\* и \*\* доступны \*\* всем.



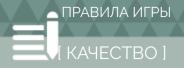
### [ метрическая

Доступ к \*\* метрикам не подразумевает доступ к данным о единице \*\*, он должен контролироваться для обеспечения конфиденциальности.

\*\* Все среды 🔭 🚗 😝 уты.







\*\* качество программного обеспечения \*\* является \*\* ключевым фактором \*\*.

# ПРАВИЛА ИГРЫ [ КАЧЕСТВО]

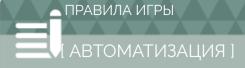
\*\* обзоры кода \*\* являются \*\* систематическими \*\*. Они проводятся членами функциональной группы или другими членами организации в рамках \*\* Постоянного совершенствования \*\*.

Это \*\* вы не проверяетесь, а ваш код \*\*: «Вы не ваш код!».

\*\* qualimetry \*\* может быть частично автоматизирован, но ничто не сравнится с \*\* «новым глазом» \*\*.







\*\* автоматическое тестирование \*\* является необратимым предварительным условием для непрерывного развертывания.



## **=**ПРАВИЛА ИГРЬ

### [ АВТОМАТИЗАЦИЯ

Автоматизированное \*\* тестирование \*\* гарантирует \*\* качество \*\* продукта \*\* со временем \*\*.

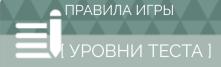
Это \*\* предварительное условие \*\* для непрерывного развертывания, оно позволяет \*\* \*\* изменения \*\* и \*\* частое развертывание \*\*.

\*\* Производственный свиток \*\* становится \*\* анекдотическим \*\* событием!









\*\* Тесты на всех уровнях

\*\*: единица, интеграция,
функциональность,
устойчивость,
производительность.





#### [ УРОВНИ ТЕСТА ]

\*\* \*\* тесты \*\* подходят для \*\* разработки \*\*.

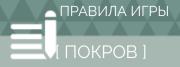
\*\* выступление \*\* тест измерение работоспособность \*\* со временем \*\*.

\*\* Устойчивость \*\* тесты помогают прогнозировать \*\* сбои \*\*.









\*\* Обложка \*\* является основным объективным показателем качества теста.





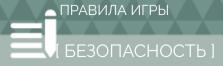
\*\* Кодирование \*\* в тестах является \*\* хорошим \*\* метрикой качества кода.

Это \*\* необходимое условие \*\*, но \*\* недостаточно \*\*, охват \*\* плохой тест-теста может быть высоким, не гарантируя хорошего качества кода.









\*\* security \*\* - это \*\* процесс \*\*, он не должен рассматриваться в ответ на проблемы.



### [БЕЗОПАСНОСТЬ]

\*\* эксперты по безопасности \*\* могут быть \*\* интегрированы \*\* непосредственно в группы функций \*\* при необходимости \*\*.

\*\* эксперты по безопасности \*\* доступны в организации для \*\* аудита \*\*, \*\* осведомленности \*\* и \*\* вперед \*\*.





