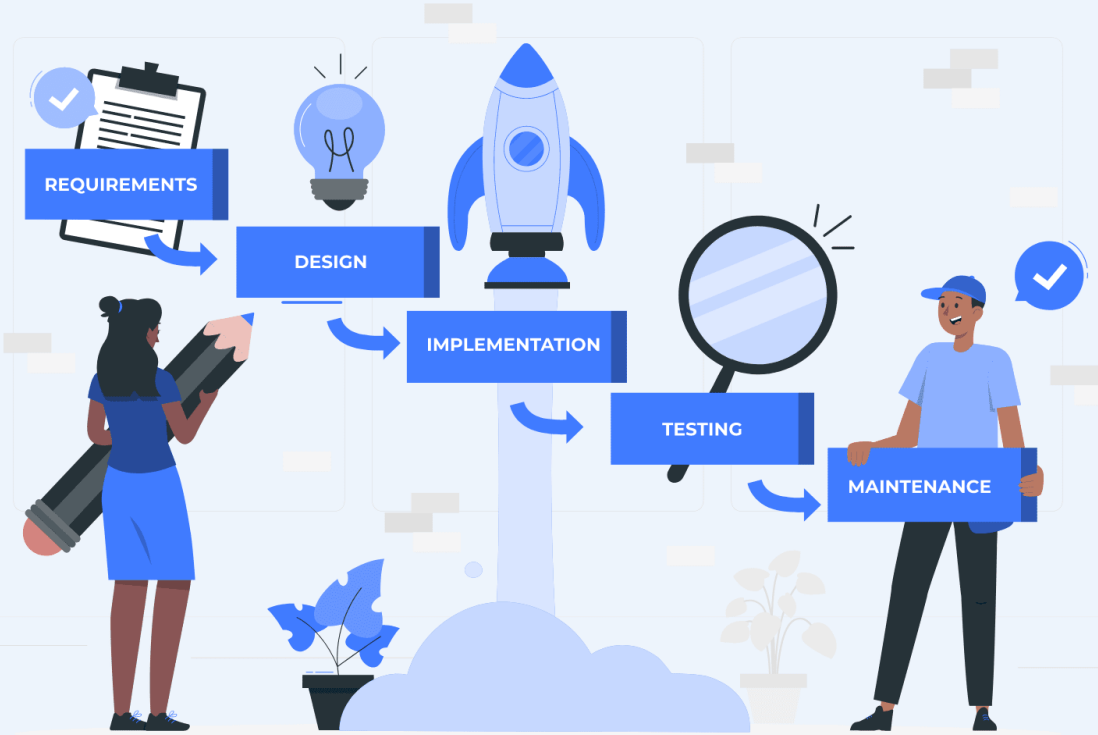
RUDN university

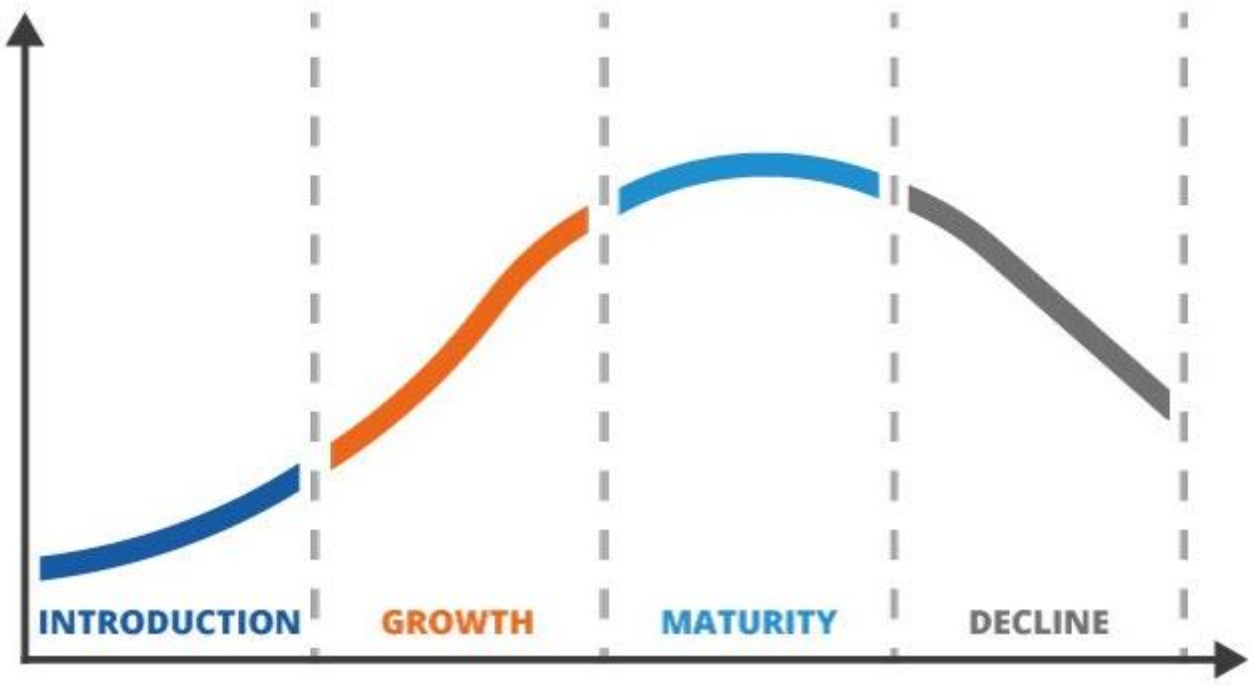
ЖИЗНЕННЫЙ ЦИКЛ **MVP**

АРХИТЕКТУРА, РАЗВИТИЕ, ЭКСПЛУАТАЦИЯ



Life Cycle

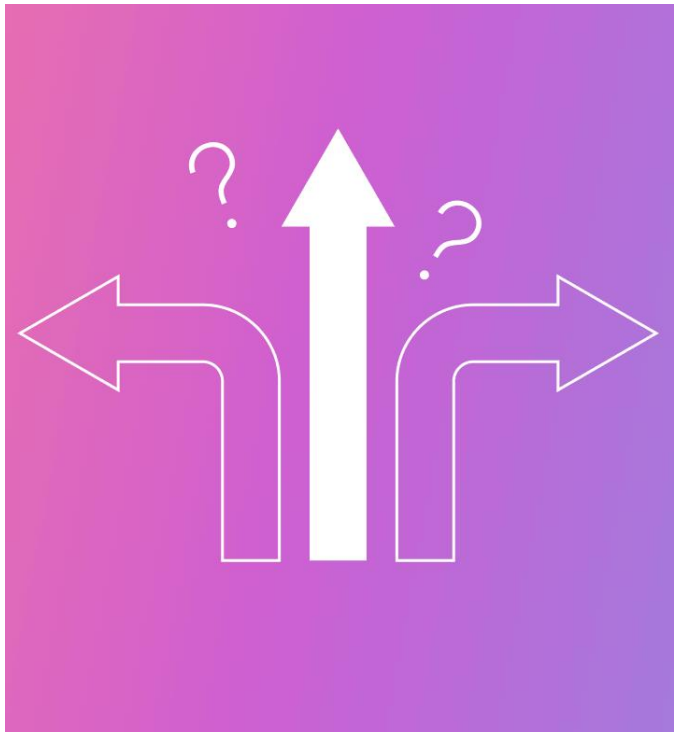
ЖИЗНЕННЫЙ ЦИКЛ МАЛЫХ СЕРВИСОВ



Жизненный цикл сервиса — это совокупность этапов его существования от зарождения идеи до вывода из эксплуатации.

Малый сервис подчиняется тем же законам эволюции, что и крупные системы, даже при небольшом масштабе.

ФОРМУЛИРОВКА ЗАДАЧИ И ТРЕБОВАНИЯ



Формулирование требований — это переход от расплывчатой идеи к четко описанному поведению системы и ее ограничений.

Требования к сервису — это формальное описание того, что система должна делать и в каких условиях.

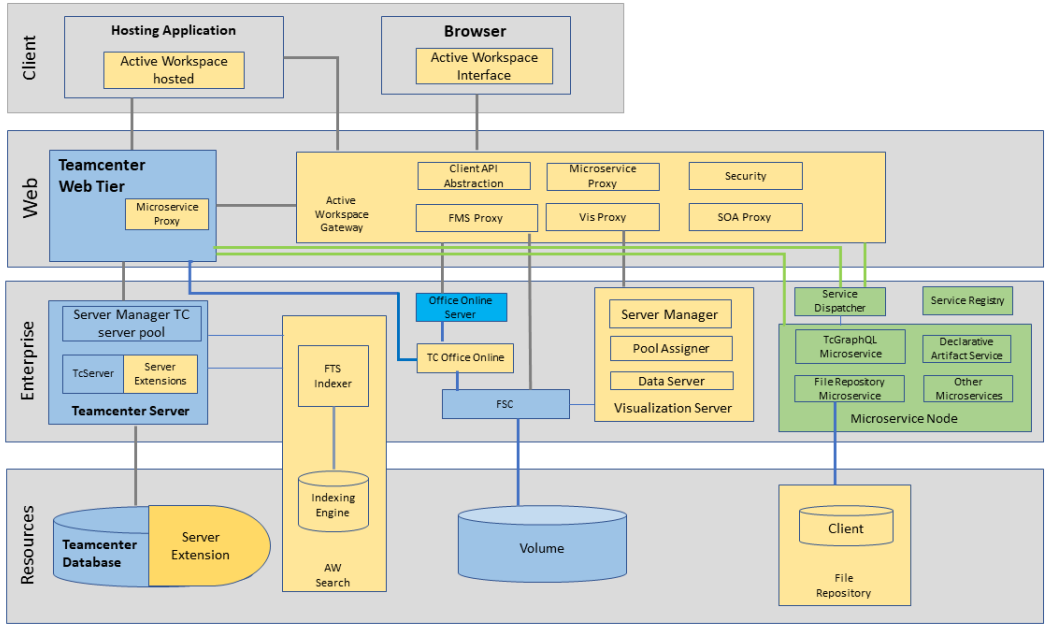
Функциональные требования задают функции и услуги, которые система обязана предоставлять пользователям и другим системам, а также условия, при которых эти функции выполняются.

*Описывают **что** система должна делать.*

Нефункциональные требования определяют ограничения и характеристики качества системы, такие как производительность, надёжность, безопасность, удобство сопровождения, масштабируемость, совместимость и т.п., не описывая конкретные функции напрямую.

*Описывают **как** система должна выполнять свои функции.*

АРХИТЕКТУРА И ВЫБОР ТЕХНОЛОГИЙ

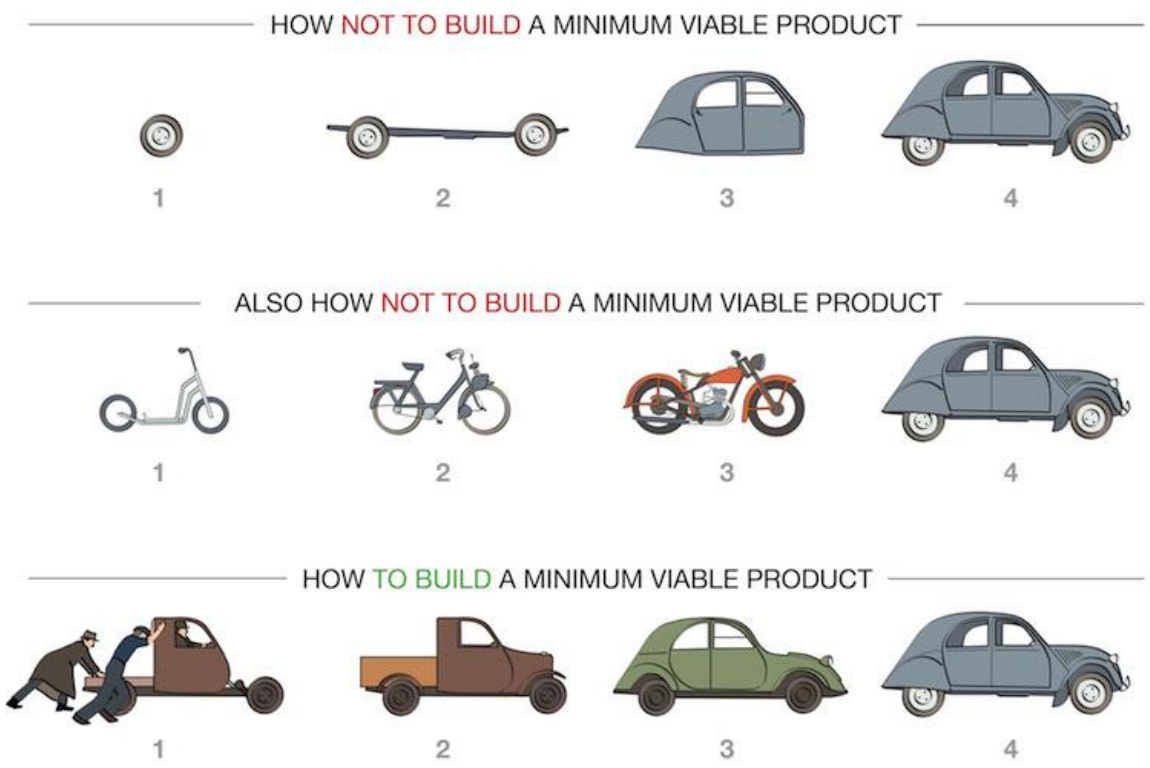


Архитектура малого сервиса — это разбиение на слои (интерфейс, бизнес-логика, данные, инфраструктура) и определение связей между ними.

От архитектуры зависит, будут ли изменения локализованными или потребуют переписывания значительной части системы.

Выбор технологий (языка, фреймворков, СУБД, способов конфигурации и развертывания) должен подчиняться архитектурным решениям и учитывать прогнозируемое развитие сервиса.

МИНИМАЛЬНО ЖИЗНЕСПОСОБНАЯ ВЕРСИЯ (**MVP**)

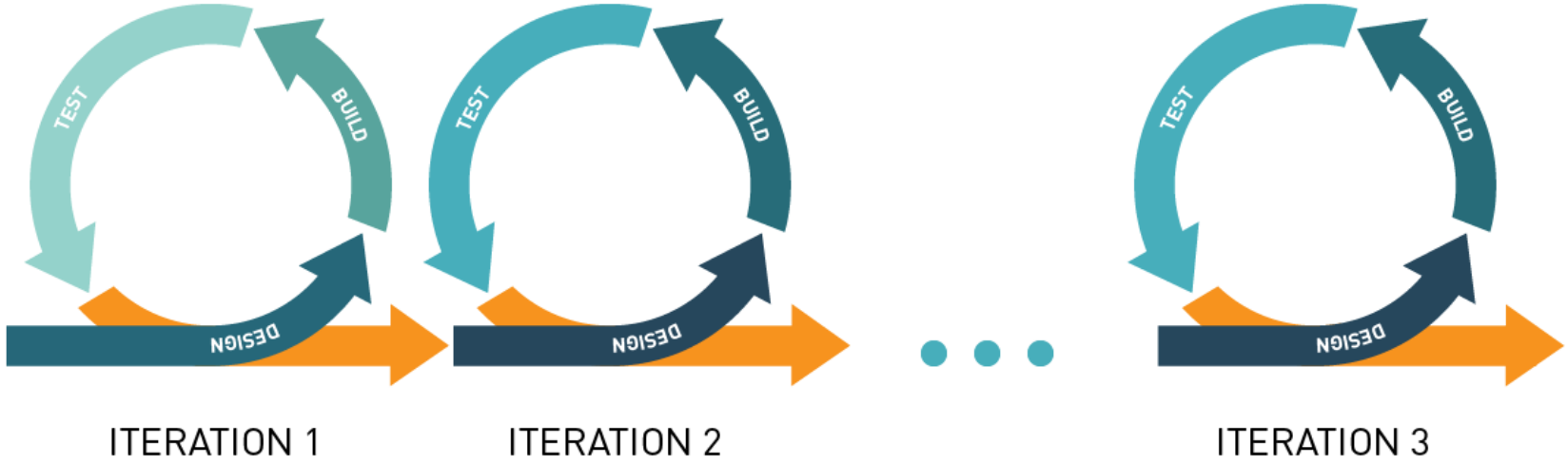


Минимально жизнеспособная версия — это первая реализация сервиса, которая уже решает полезный для пользователя сценарий, но содержит только необходимый минимум функций.

MVP тестирует жизнеспособность архитектурных решений: проверяется, что ключевая цепочка «запрос → обработка → данные → ответ» работает корректно.

На этом этапе важно сосредоточиться на правильности поведения и чистоте слоёв, а не на полноте функционала.

ИТЕРАТИВНОЕ РАЗВИТИЕ



Итеративное развитие — это последовательность циклов изменений, в каждом из которых уточняются требования, модифицируется код, схема данных и конфигурация, а результат проверяется тестами и в эксплуатации.

Управление изменениями включает анализ влияния каждого изменения, аккуратную работу с миграциями, конфигурацией и feature-toggles.

Без контроля изменений малый сервис быстро превращается в плохо управляемый набор случайных решений.

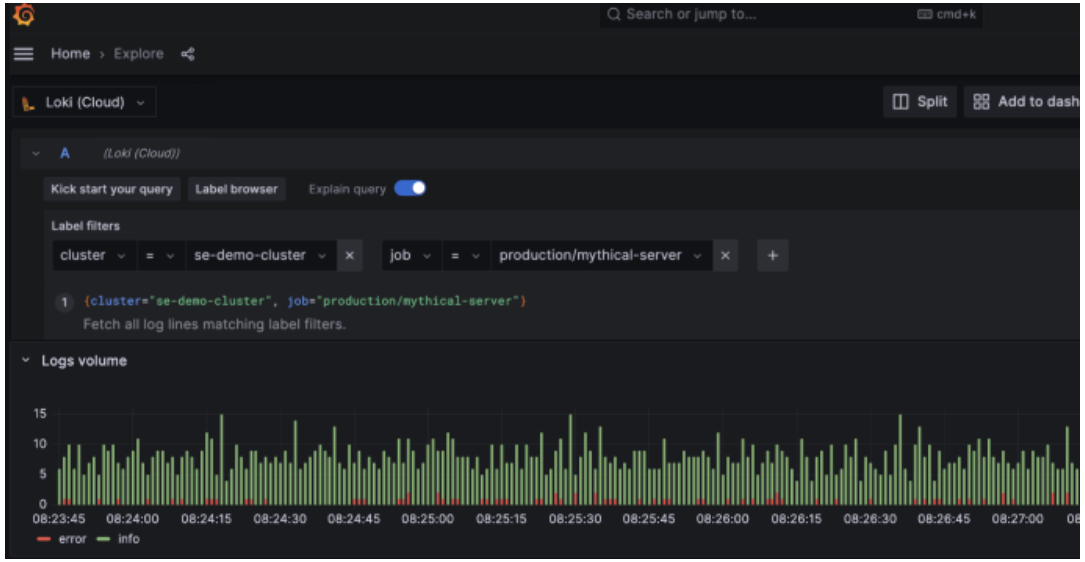
КОНТРОЛЬ КАЧЕСТВА

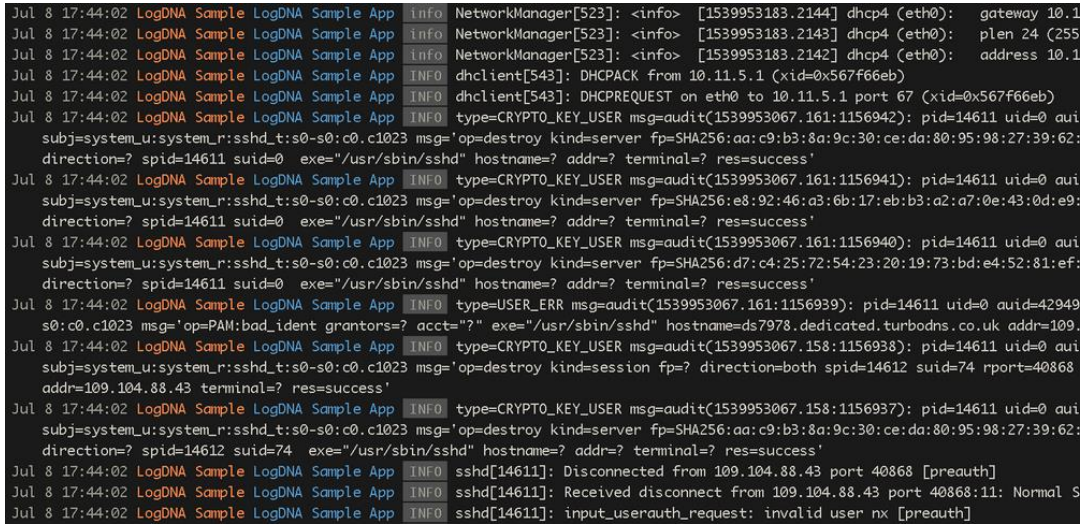
Тестирование в жизненном цикле малого сервиса обеспечивает устойчивость к изменениям и предотвращает регрессии.

Модульные, интеграционные и контрактные тесты закрепляют ожидаемое поведение кода и интерфейсов, а статический анализ и ревью кода помогают выявлять проблемы до запуска.

Качество системы — это результат не разовой проверки перед релизом, а постоянного применения инженерных практик на всех этапах жизни сервиса.

ЭКСПЛУАТАЦИЯ





Эксплуатация опирается на три ключевых механизма: конфигурацию, логирование и мониторинг.

Конфигурация управляет поведением сервиса без изменения кода, **логирование** фиксирует важные события и ошибки, а **мониторинг** даёт агрегированную картину состояния через метрики.

Вместе они обеспечивают **наблюдаемость** и **управляемость**: позволяют понимать, что происходит с сервисом и как он реагирует на изменения нагрузки и среды.

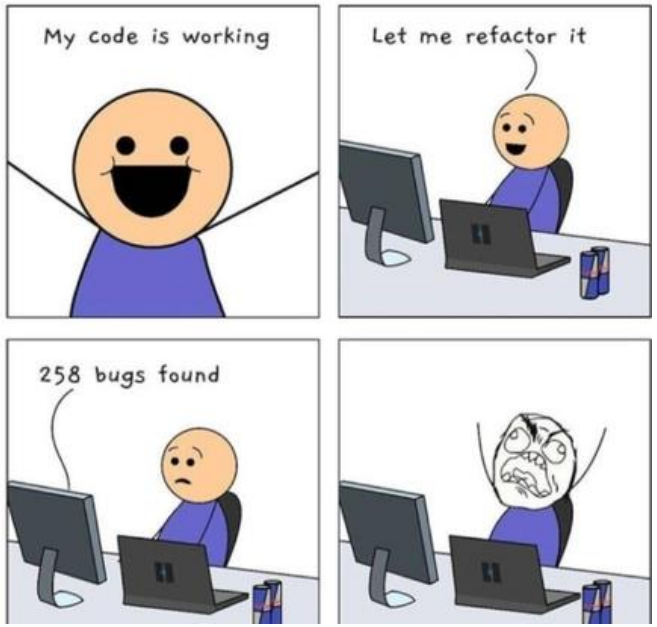
ИНЦИДЕНТЫ

Инцидент — это отклонение работы сервиса от ожидаемой, приводящее к деградации или недоступности функциональности.

Работа с инцидентами включает обнаружение, стабилизацию (быстрое восстановление приемлемого состояния), анализ причин и внедрение корректирующих действий.

Каждый серьёзный инцидент должен приводить не только к исправлению, но и к усилению архитектуры, тестов, логов или конфигурации, чтобы аналогичные случаи не повторялись.

ТЕХНИЧЕСКИЙ ДОЛГ И РЕФАКТОРИНГ



Технический долг — это совокупность решений, которые временно упростили разработку, но усложняют дальнейшие изменения и сопровождение системы.

Рефакторинг — управляемое изменение внутренней структуры кода и данных без изменения внешнего поведения, направленное на погашение этого долга.

Регулярная работа с техническим долгом — необходимое условие того, что даже малый сервис останется поддерживаемым, а не застывшим и опасным для изменений.

ЗАВЕРШЕНИЕ ЖИЗНЕННОГО ЦИКЛА

Завершение жизненного цикла сервиса включает управляемый вывод из эксплуатации: корректное обращение с данными, уведомление потребителей и отключение инфраструктуры.

Технически это сопровождается архивированием кода, конфигураций, схем и, при необходимости, логов, а организационно — фиксацией выводов о принятых решениях и их последствиях.