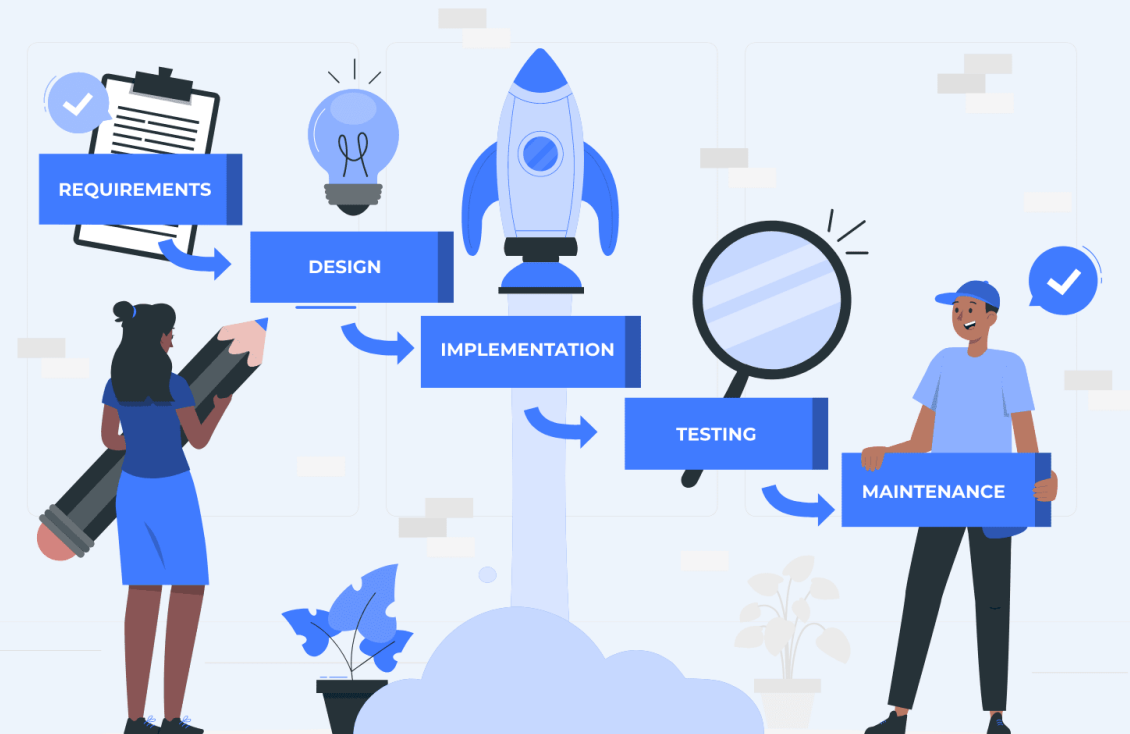


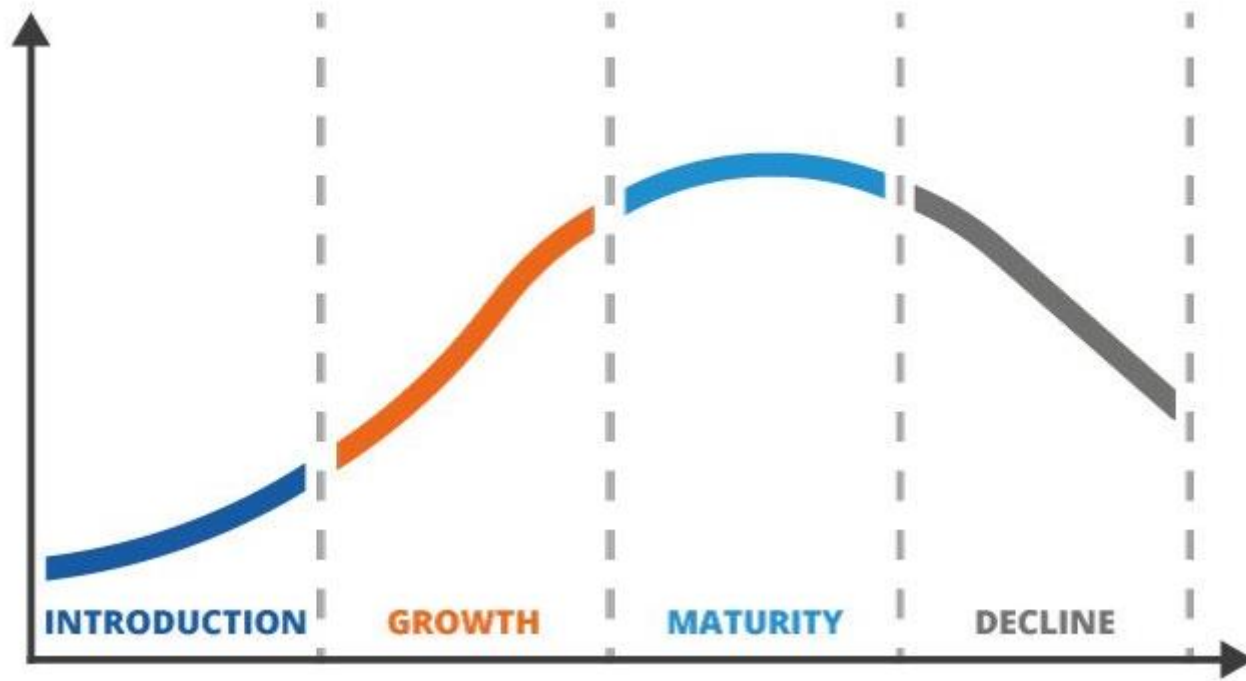
# ЖИЗНЕННЫЙ ЦИКЛ MVP

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.

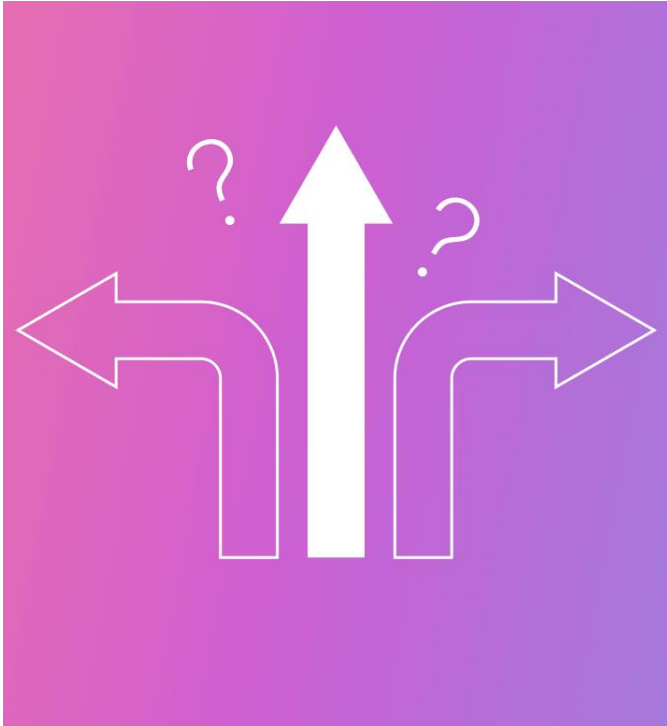
**Требования к сервису** — это формальное описание того, что система должна делать и в каких условиях.

**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.

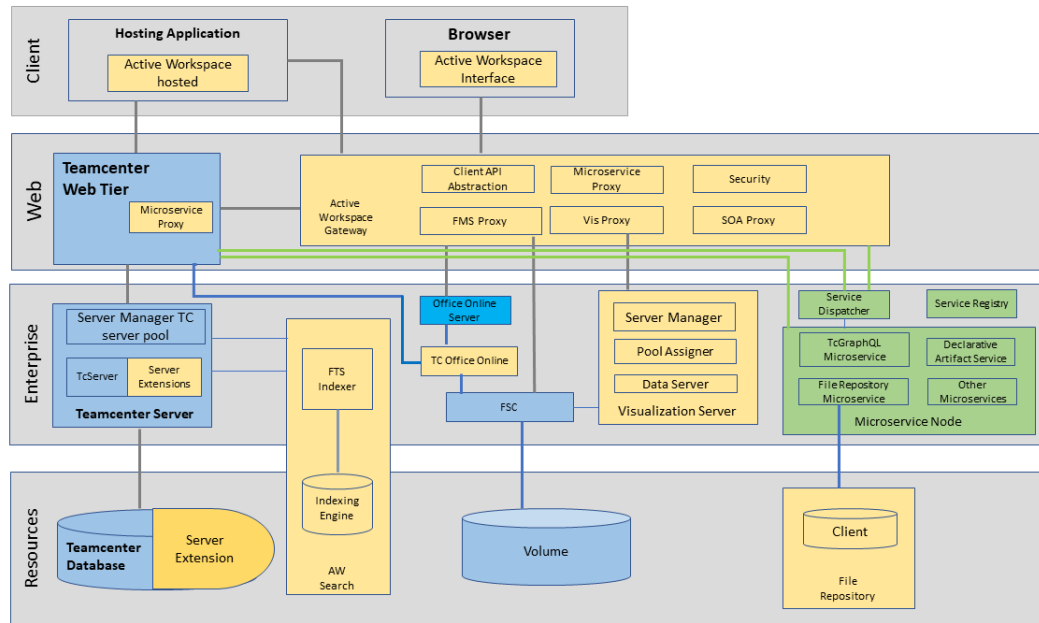


# АРХИТЕКТУРА И ВЫБОР ТЕХНОЛОГИЙ

**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**)

HOW **NOT TO BUILD** A MINIMUM VIABLE PRODUCT



ALSO HOW **NOT TO BUILD** A MINIMUM VIABLE PRODUCT



HOW **TO BUILD** A MINIMUM VIABLE PRODUCT

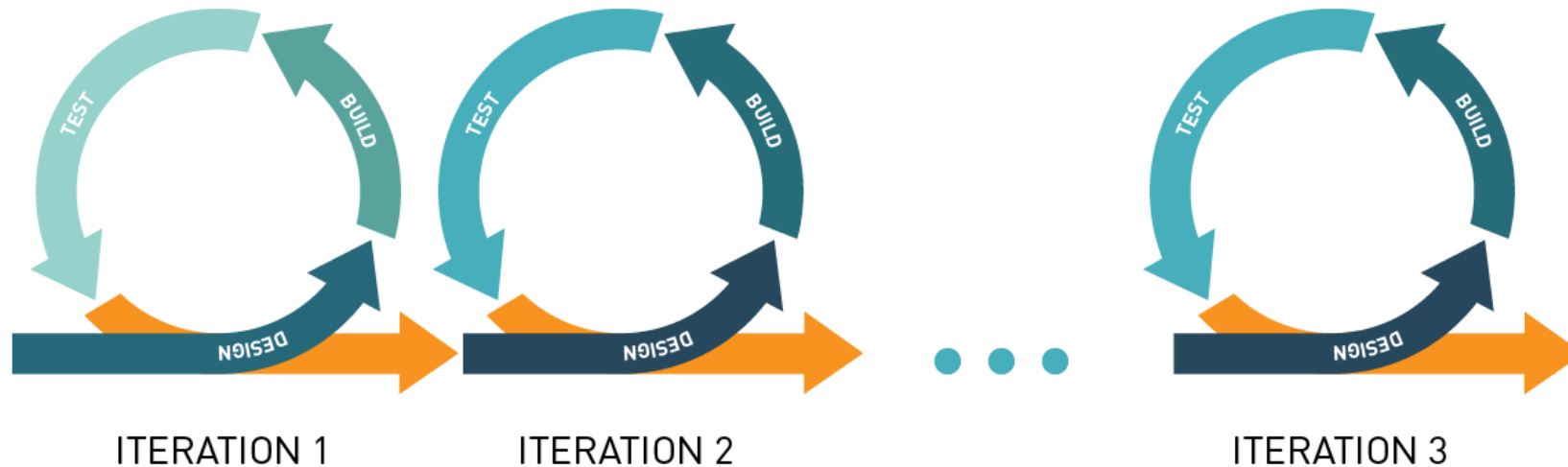


Минимально жизнеспособная версия — это первая реализация сервиса, которая уже решает полезный для пользователя сценарий, но содержит только необходимый минимум функций.

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



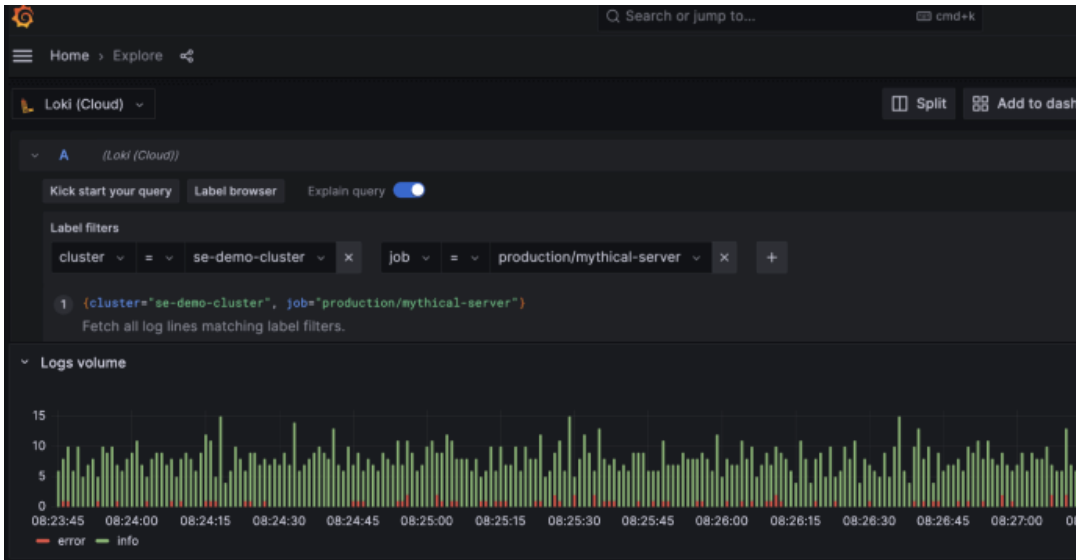
**Тестирование** в жизненном цикле малого сервиса обеспечивает устойчивость к изменениям и предотвращает регрессии.

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



```
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2144] dhcp4 (eth0): gateway 10.1
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2143] dhcp4 (eth0): plen 24 (255
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO NetworkManager[523]: <info> [1539953183.2142] dhcp4 (eth0): address 10.1
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO dhclient[543]: DHCPACK from 10.11.5.1 (xid=0x567f66eb)
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO dhclient[543]: DHCPREQUEST on eth0 to 10.11.5.1 port 67 (xid=0x567f66eb)
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156942): pid=14611 uid=0 au
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=destroy kind=server fp=SHA256:aa:c9:b3:8a:9c:30:ce:da:80:95:98:27:39:62:
direction=? spid=14611 suid=0 exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156941): pid=14611 uid=0 au
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=destroy kind=server fp=SHA256:d7:c4:25:72:54:23:20:19:73:bd:e4:52:81:ef:
direction=? spid=14611 suid=0 exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.161:1156940): pid=14611 uid=0 au
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=destroy kind=server fp=SHA256:d7:c4:25:72:54:23:20:19:73:bd:e4:52:81:ef:
direction=? spid=14611 suid=0 exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=USER_ERR msg=audit(1539953067.161:1156939): pid=14611 uid=0 auid=42949
s0:c0.c1023 msg='op=PAM:bad_ident grantors=? acct=?" exe="/usr/sbin/sshd" hostname=ds7978.dedicated.turbodns.co.uk addr=109.
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.158:1156938): pid=14611 uid=0 au
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=destroy kind=session fp=? direction=both spid=14612 suid=74 rport=40868
addr=109.104.88.43 terminal=? res=success'
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO type=CRYPTO_KEY_USER msg=audit(1539953067.158:1156937): pid=14611 uid=0 au
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=destroy kind=server fp=SHA256:aa:c9:b3:8a:9c:30:ce:da:80:95:98:27:39:62:
direction=? spid=14612 suid=74 exe="/usr/sbin/sshd" hostname=? addr=? terminal=? res=success'
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO sshd[14611]: Disconnected from 109.104.88.43 port 40868 [preauth]
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO sshd[14611]: Received disconnect from 109.104.88.43 port 40868:11: Normal S
Jul 8 17:44:02 LogDNA Sample LogDNA Sample App INFO sshd[14611]: input_userauth_request: invalid user nx [preauth]
```

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.

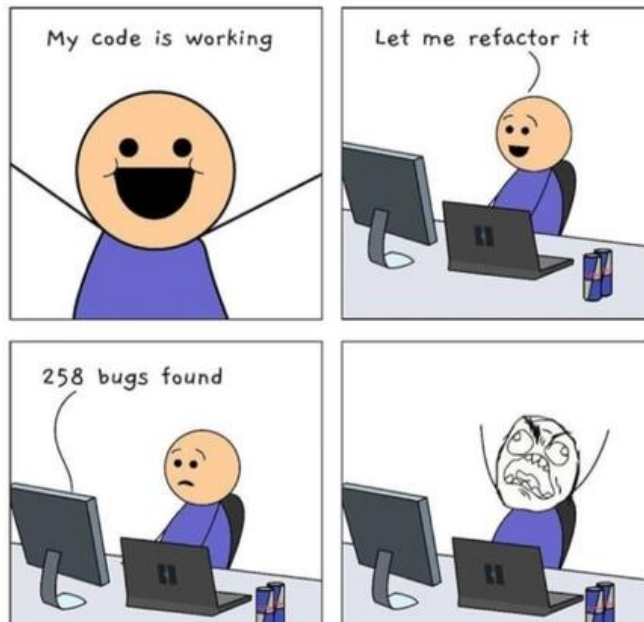
# ТЕХНИЧЕСКИЙ ДОЛГ И РЕФАКТОРИНГ



**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.