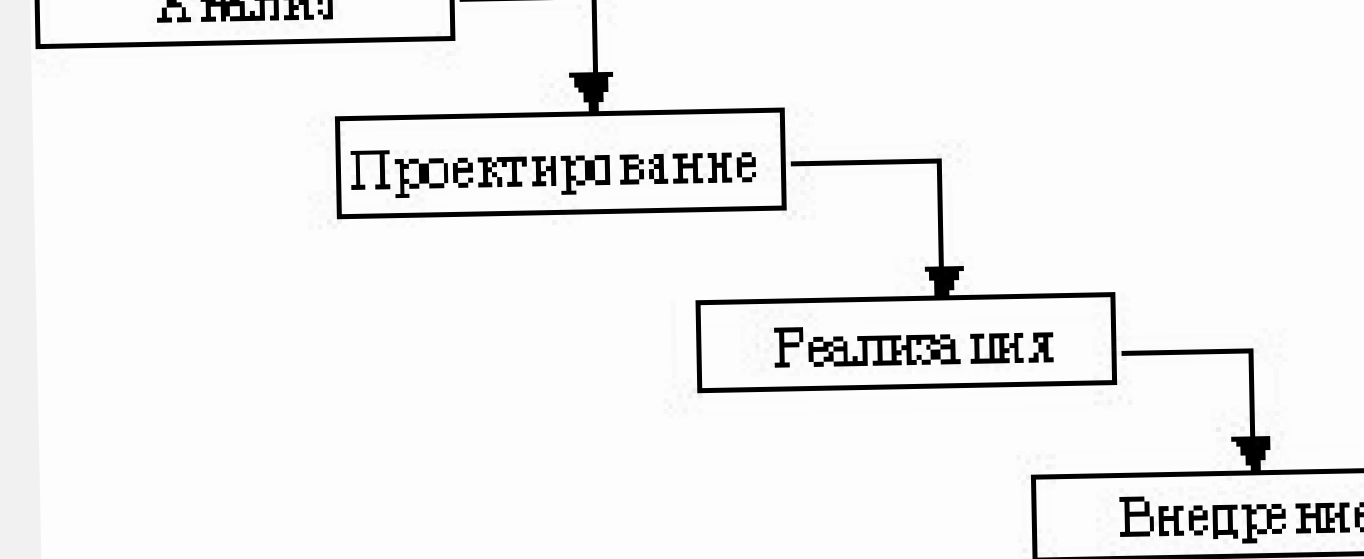


# Этапы жизненного цикла информационных систем

Эксплуатация, сопровождение и модернизация

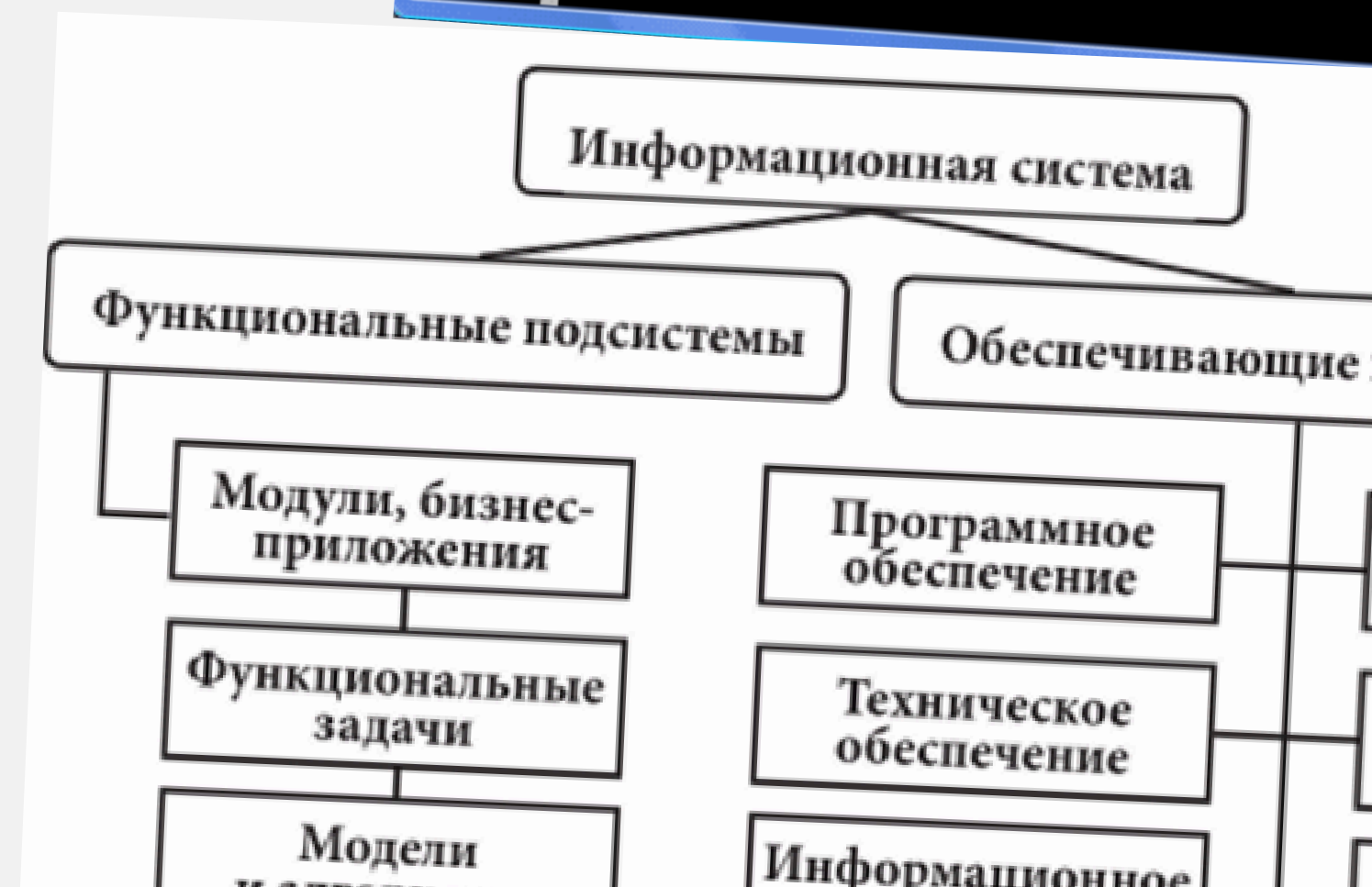
*На этом этапе система уже запущена, ей пользуются люди, появляются ошибки, новые требования и обновления.*



```
mysql> SELECT * FROM users;
```

id_user	name	email	password
1	sergey	sergey@mail.ru	1111
2	valera	valera@mail.ru	2222
3	katy	katy@gmail.ru	3333
4	sveta	sveta@rambler.ru	4444
5	oleg	oleg@yandex.ru	5555

5 rows in set (0.10 sec)  
mysql>



# Что происходит после запуска системы 🏰

- Система начинает работать с реальными пользователями и данными.
- Обнаруживаются ошибки, которых не было видно на тестах.
- Обновляются серверы, операционные системы, базы данных.
- Требования меняются: нужно добавить отчёт, форму, интеграцию.



# Эксплуатация, сопровождение, модернизация 🚂

**Эксплуатация** – система работает в обычном режиме и выполняет свои функции.

**Сопровождение** – исправляются ошибки, вносятся мелкие изменения, система подстраивается под новое окружение.

**Модернизация** – более крупные изменения: переход на новую платформу, серьёзное обновление функциональности или архитектуры.



# Зачем тестирование после внедрения🐎

- Любое изменение может что-то сломать.
- Тестирование проверяет, что новая функция работает как нужно.
- Регрессионные тесты проверяют, что старые функции не сломались.
- Нагрузочные тесты показывают, выдержит ли система рост пользователей.
- Тесты безопасности помогают не открыть новую «дыру» при обновлении.





# Виды тестирования ♚

Модульное тестирование — проверяем маленький кусок программы (одну функцию).

Интеграционное — смотрим, как несколько модулей работают вместе.

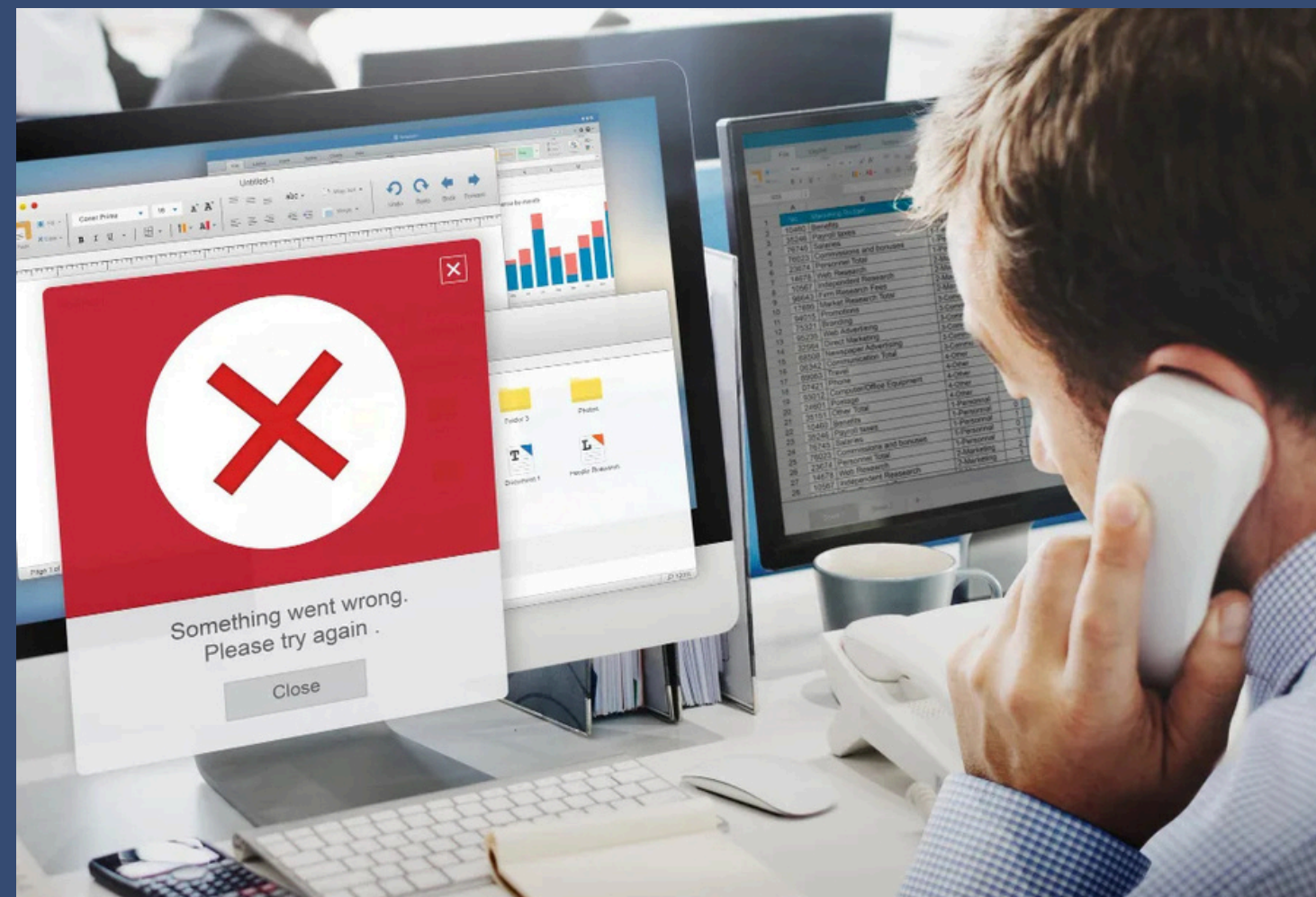
Системное — проверяем всю систему целиком.

Приёмочное — проверяем типичные действия пользователя: «от начала до конца».



# Как ищут ошибки в работающей системе

- Логи (журналы событий) помогают понять, что происходило перед ошибкой.
- Мониторинг показывает графики нагрузки, времени отклика и числа ошибок.
- Стенды сопровождения позволяют безопасно воспроизвести проблему не на «боевой» системе.
- Профилировщики помогают находить «тормоза»: где тратится время и память.



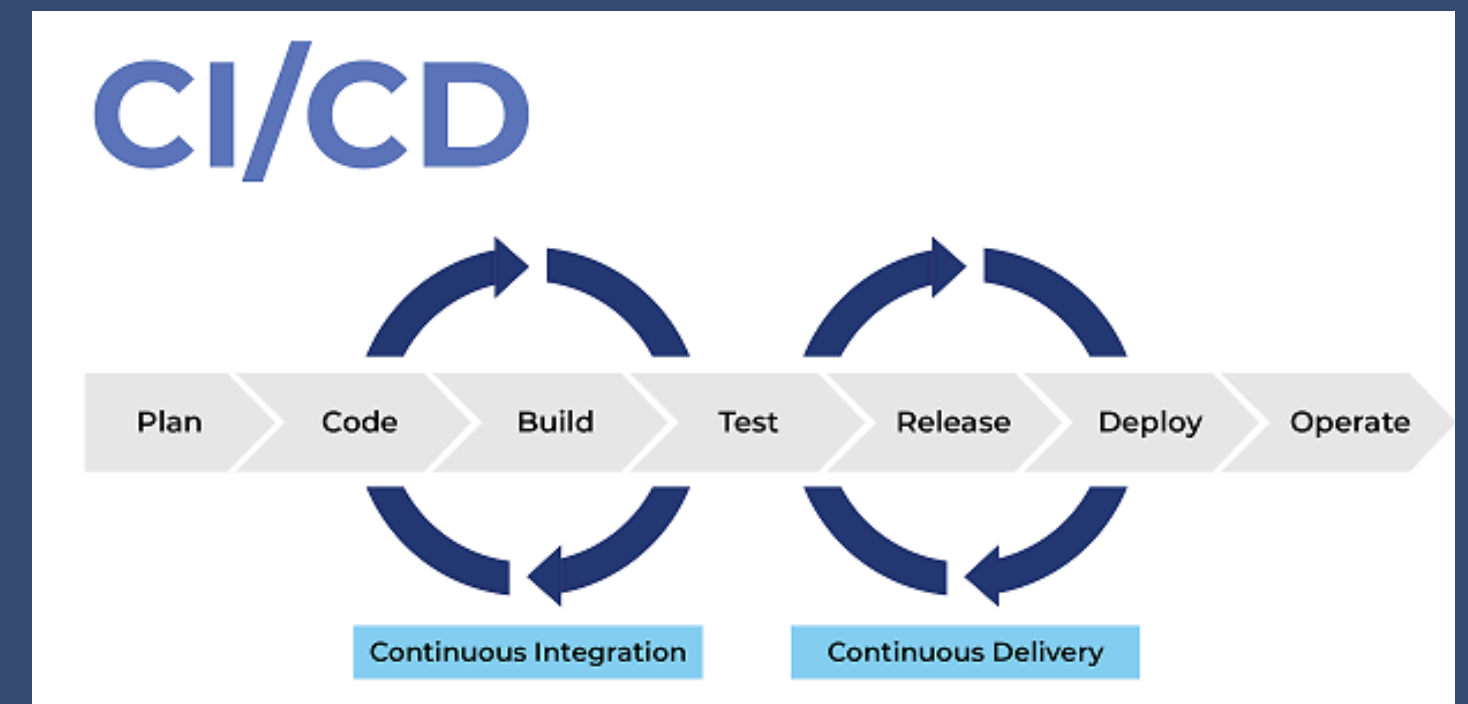
# Как изменения попадают в систему 🏰

Обычно используется цепочка «сборка → проверка → установка».

Код собирается автоматически, запускаются тесты, формируется релиз. Стратегия:

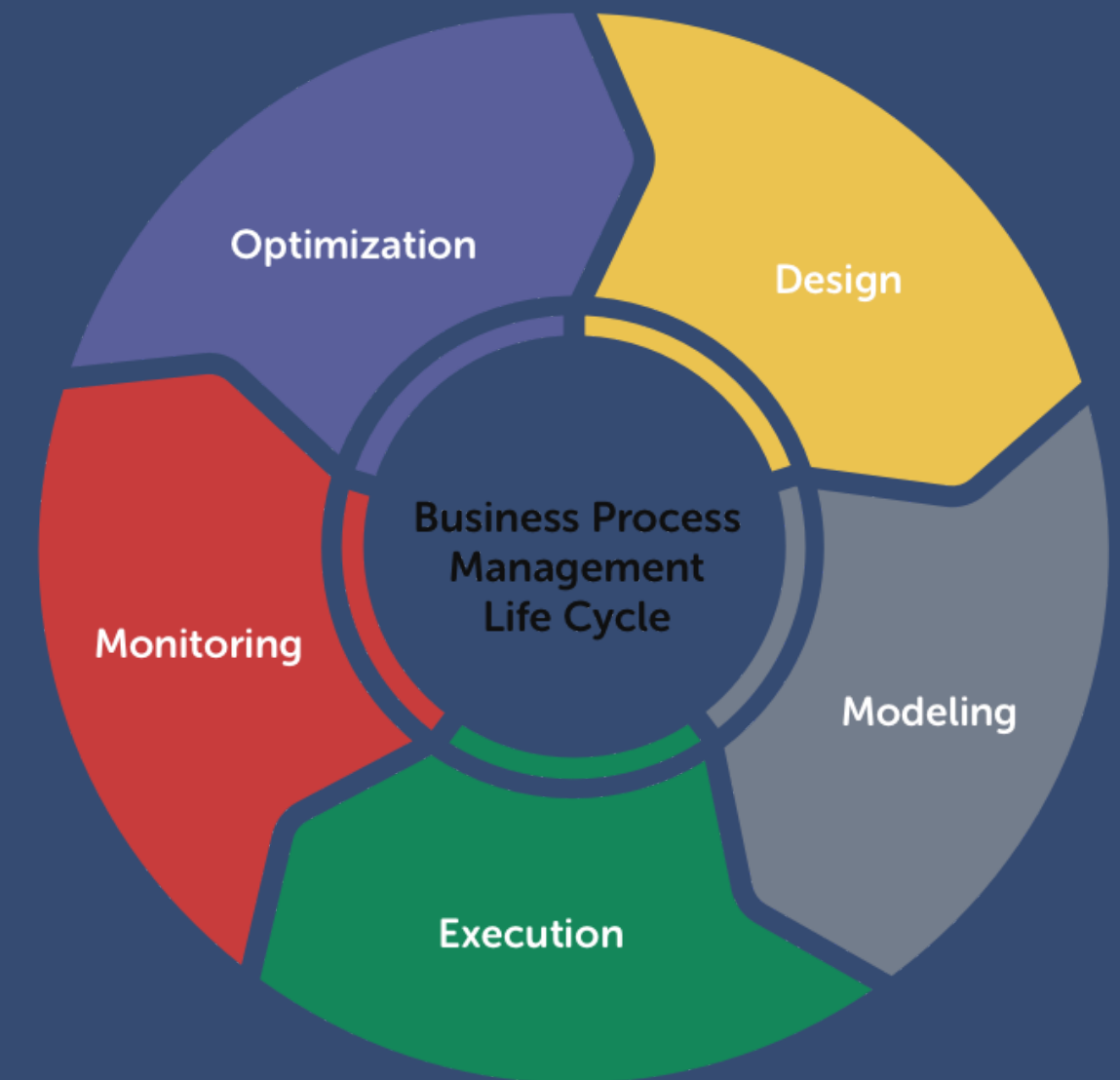
- две среды (blue-green) и переключение трафика;
- сначала небольшая группа пользователей (canary);
- поочерёдное обновление серверов без остановки сервиса.

Цель — установить обновление с минимальным риском для пользователей.



# Тестирование всей системы и данных 🏰

- Проверяются сквозные бизнес-процессы: от ввода данных до отчёта.
- Тестируется работа с внешними системами (API, сторонние сервисы).
- При изменении структуры баз данных проверяются миграции: перенос и преобразование данных.
- Отдельно проверяется резервное копирование и восстановление после отказа.





# Итог

## *Зачем управлять эксплуатацией и модернизацией*

Эксплуатация, сопровождение и модернизация нужны для того, чтобы система:

- не ломалась при изменениях;
- могла работать много лет, а не один-два сезона;
- безопасно обновлялась и не теряла данные;
- постепенно развивалась вместе с требованиями.

Главная идея: жизненный цикл системы не заканчивается после разработки.

