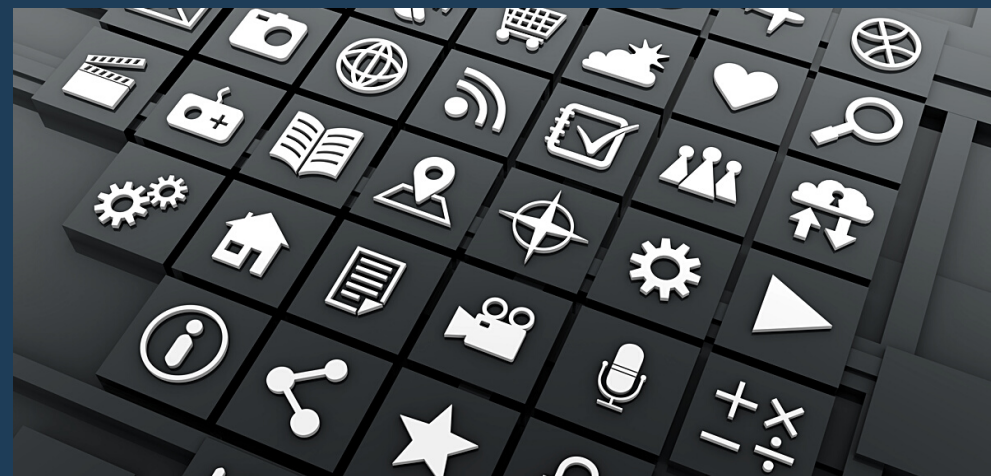
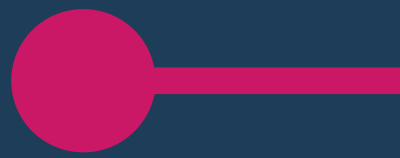


MARY KAY



LOW CODE ПЛЮСЫ И МИНУСЫ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ

СРЕДСТВА АВТОМАТИЗАЦИИ



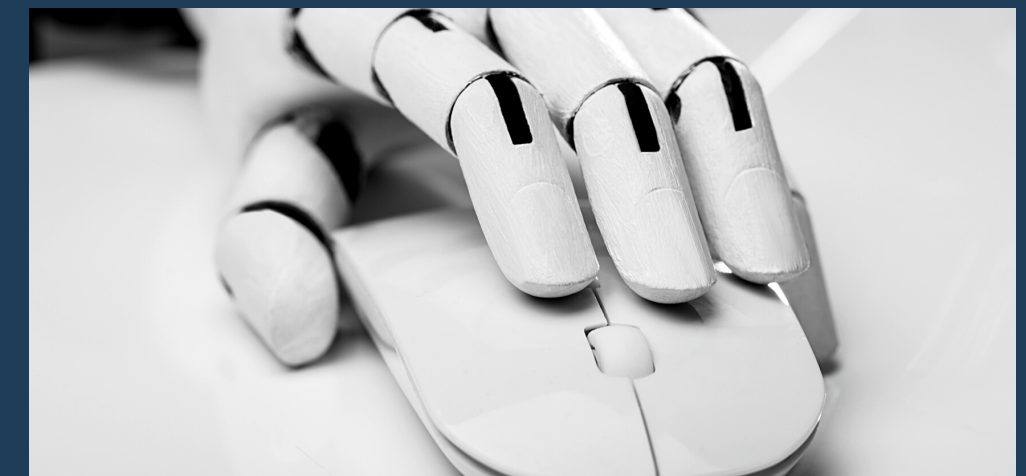
ПРИЛОЖЕНИЯ

Доработка и настройка функционала корпоративных систем



СЕРВИСЫ

Разработка специализированных сервисов



РОБОТЫ

Создание программных роботов для рутинных операций

Факторы успешной автоматизации



ПОЧЕМУ МЫ ЛЮБИМ LOW CODE

СКОРОСТЬ

- Быстрая реализация стандартных решений
- Быстрое внесение изменений

ТЕХНОЛОГИЧНОСТЬ

- Большой набор готовых "блоков"
- Нативная интеграция с популярными системами
- Регулярное обновление и расширение функционала

МАСШТАБИРУЕМОСТЬ

- Заложено в платформу масштабирование
- Репликация однотипных процессов на все рынки региона

КАДРЫ

- Ниже требования к квалификации для разработки и поддержки
- Меньшее время обучения и ниже порог вхождения
- Большой фокус на бизнес процесс

ЛОЖКА ДЁГТЯ

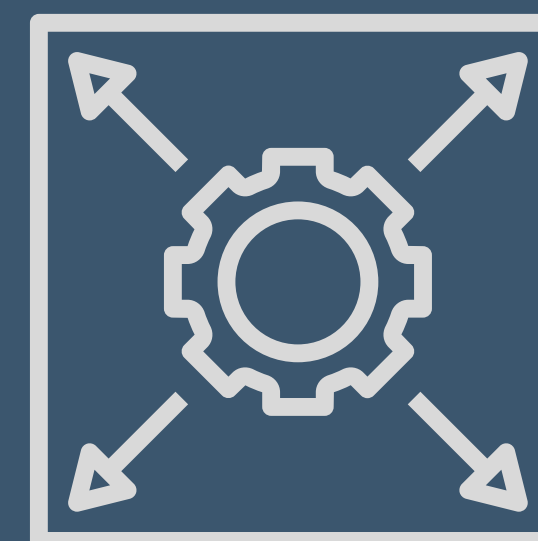
**ЗАВИСИМОСТЬ
ОТ ПЛАТФОРМЫ**



**СТОИМОСТЬ
ЛИЦЕНЗИЙ**



**ОГРАНИЧЕНИЯ
КАСТОМИЗАЦИИ**



Трансформация компетенций



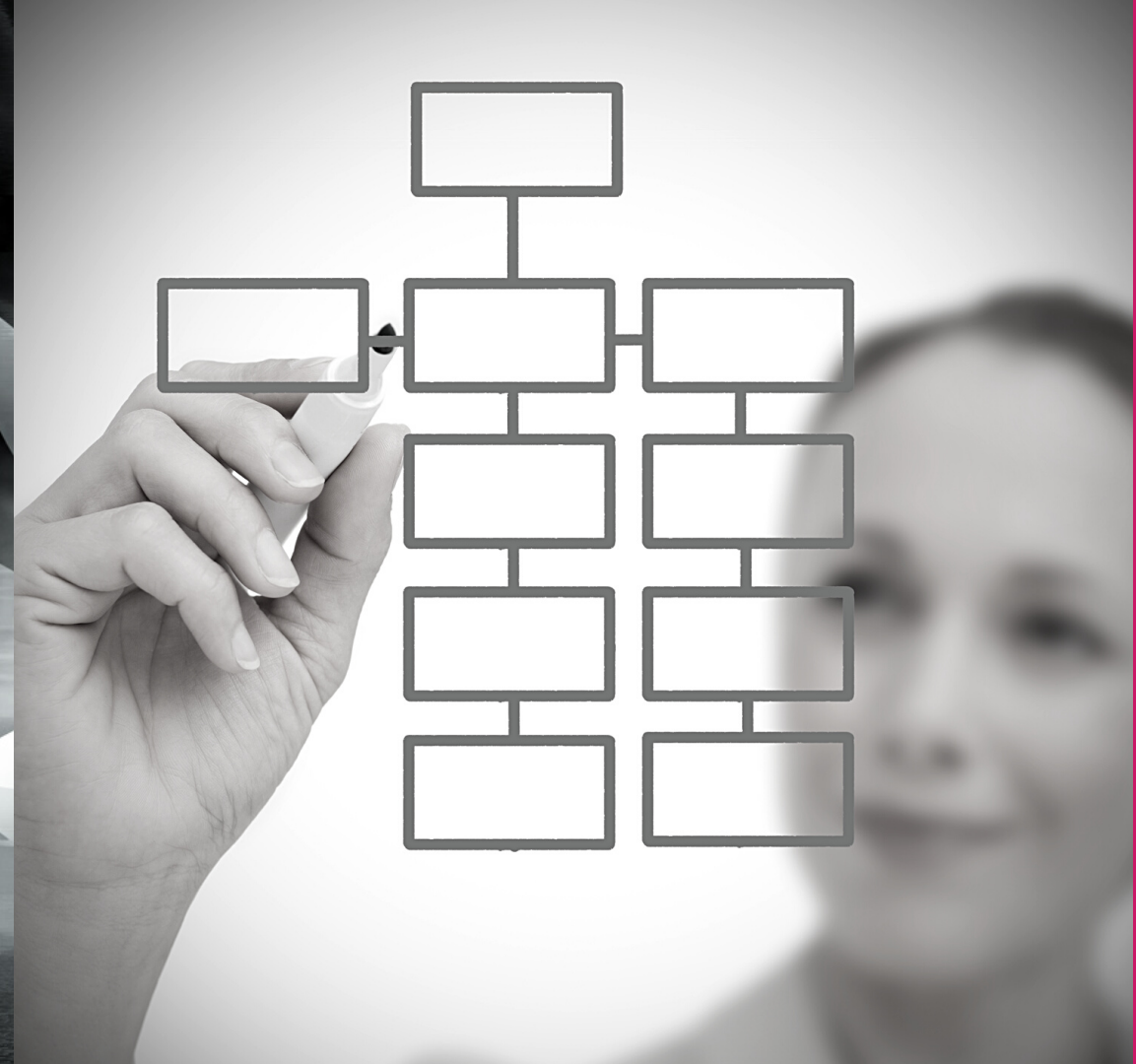
НОВЫЕ ВОЗМОЖНОСТИ И РИСКИ





```
public:  
    virtual int dummy( int, int ) {}  
private:  
    virtual int add_( int a, int b ) { return a + b }  
    virtual int sub_( int a, int b ) { return a - b }  
    virtual int mul_( int a, int b ) { return a * b }  
    virtual int div_( int a, int b ) { return a / b }  
    virtual int mod_( int a, int b ) { return a % b }  
    virtual int and_( int a, int b ) { return a & b }  
    virtual int or_( int a, int b ) { return a | b }  
    virtual int xor_( int a, int b ) { return a ^ b }  
  
int main( int argc, char* argv[] ) {  
    typedef int (Cpu::*Memfun)( int, int );  
  
    union {  
        Memfun fn:  

```



Программисты не нужны?

(*спойлер* всё ещё нужны)



СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

