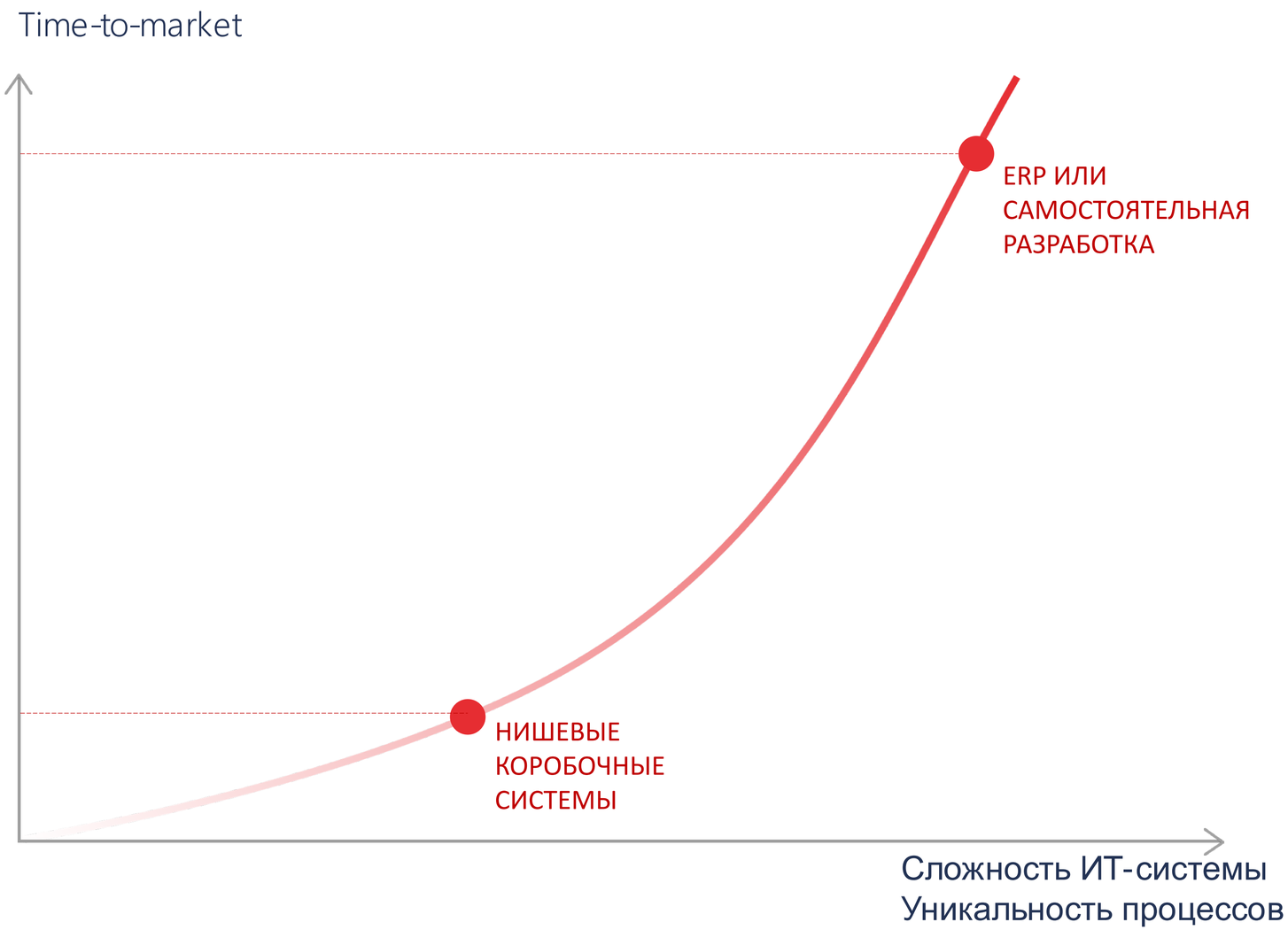
ELMA365

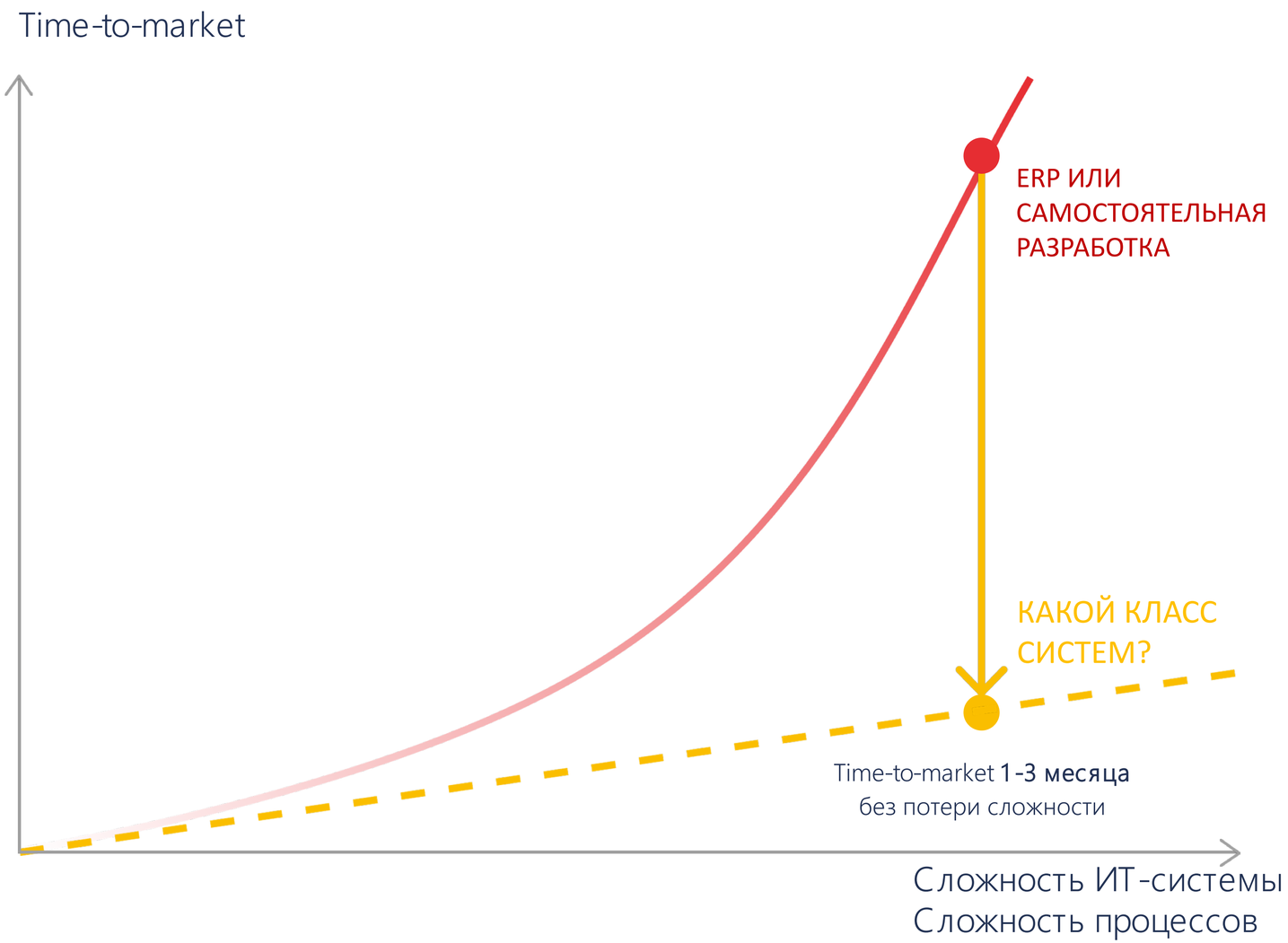
Ключевые подходы и инструменты автоматизации бизнес-процессов

elma365.com

Проблематика

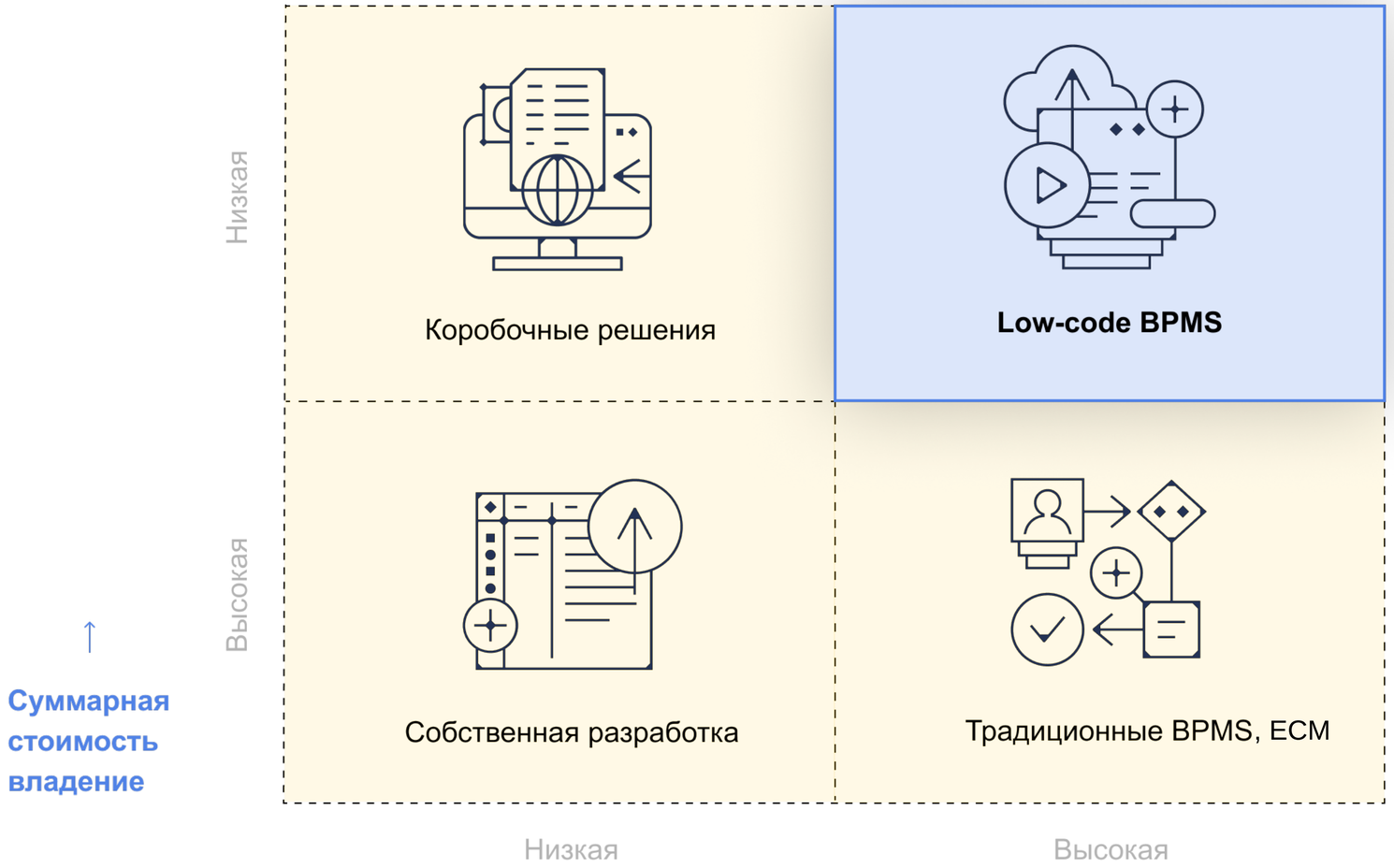


Проблематика



Сегодня изменились требования бизнеса – корпоративные системы должны удовлетворять потребности бизнеса в режиме реального времени. Параметр Time-to-Market стал решающим, внедрение изменений в системе должно происходить не за год, как раньше, а за 1-2 месяца.

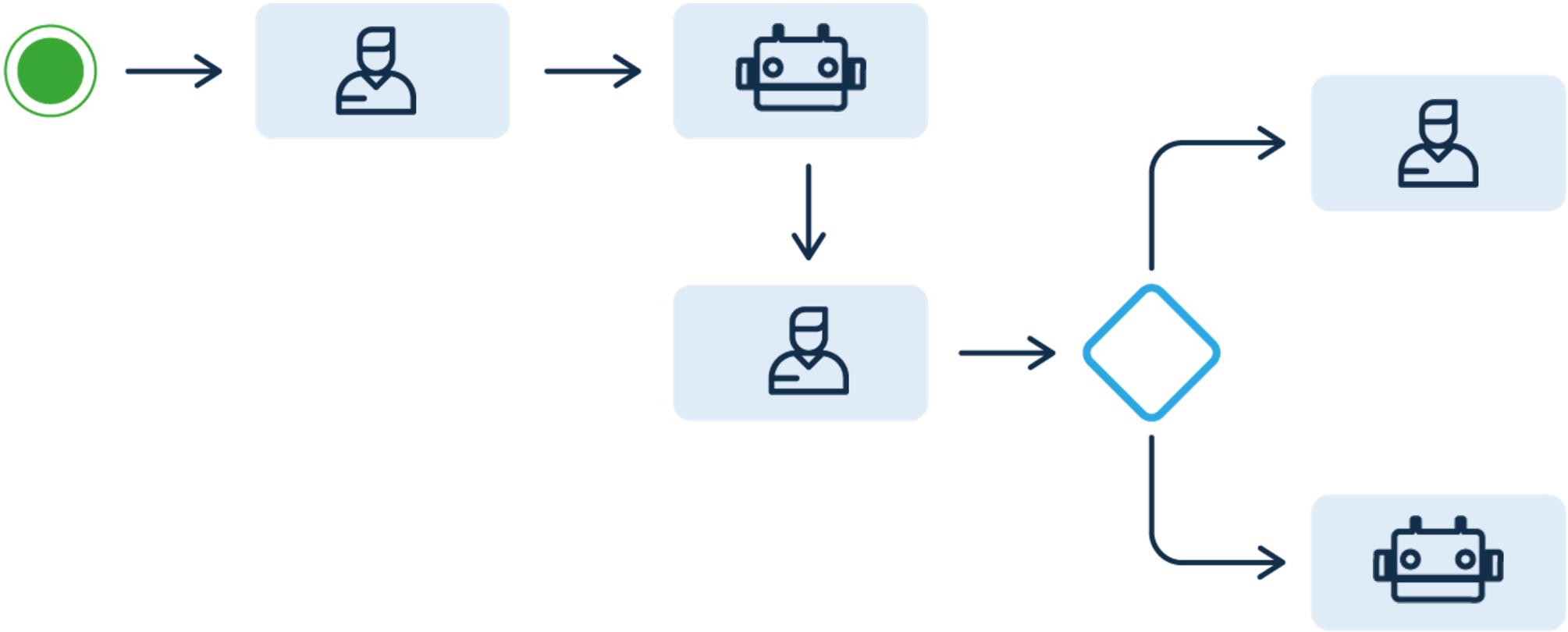
Подходы к автоматизации



Подходы к автоматизации

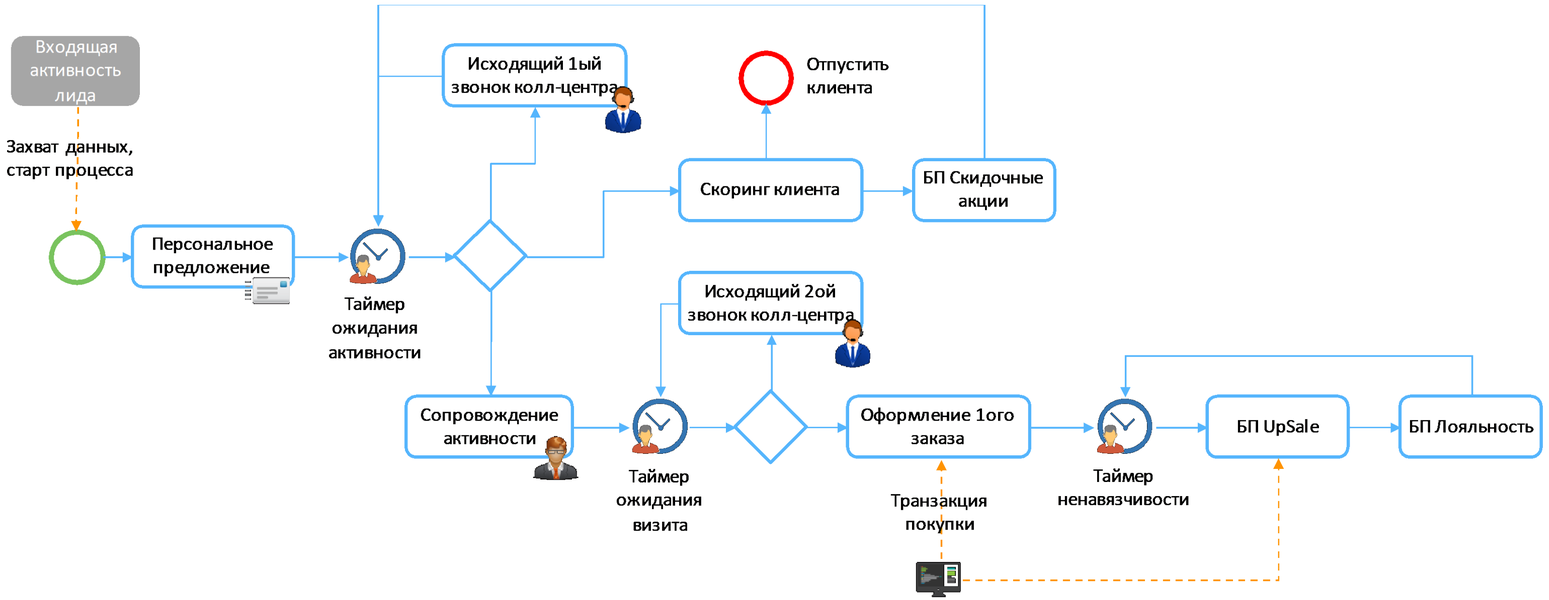
Несколько слов про RPA

Идея RPA



В СТАБИЛЬНОМ ПРОЦЕССЕ ЗАМЕНЯЕМ СОТРУДНИКОВ РОБОТАМИ

Классификация процессных задач



Инициация процесса. Формализованное действие запускающее бизнес-процесс.

Оцифровка данных. Перевод данных из бумаги, аудио- и других каналов в цифровой. Получение текстовых данных.

Формализация данных. Перевод данных из произвольного текстового вида в структурированный вид. Заполнение форм.

Обогащение данных структурированное. Поиск дополнительных структурированных данных и их фиксация в структурированном виде.

Перемещение данных между сотрудниками. Передача данных по произвольным каналам между сотрудниками.

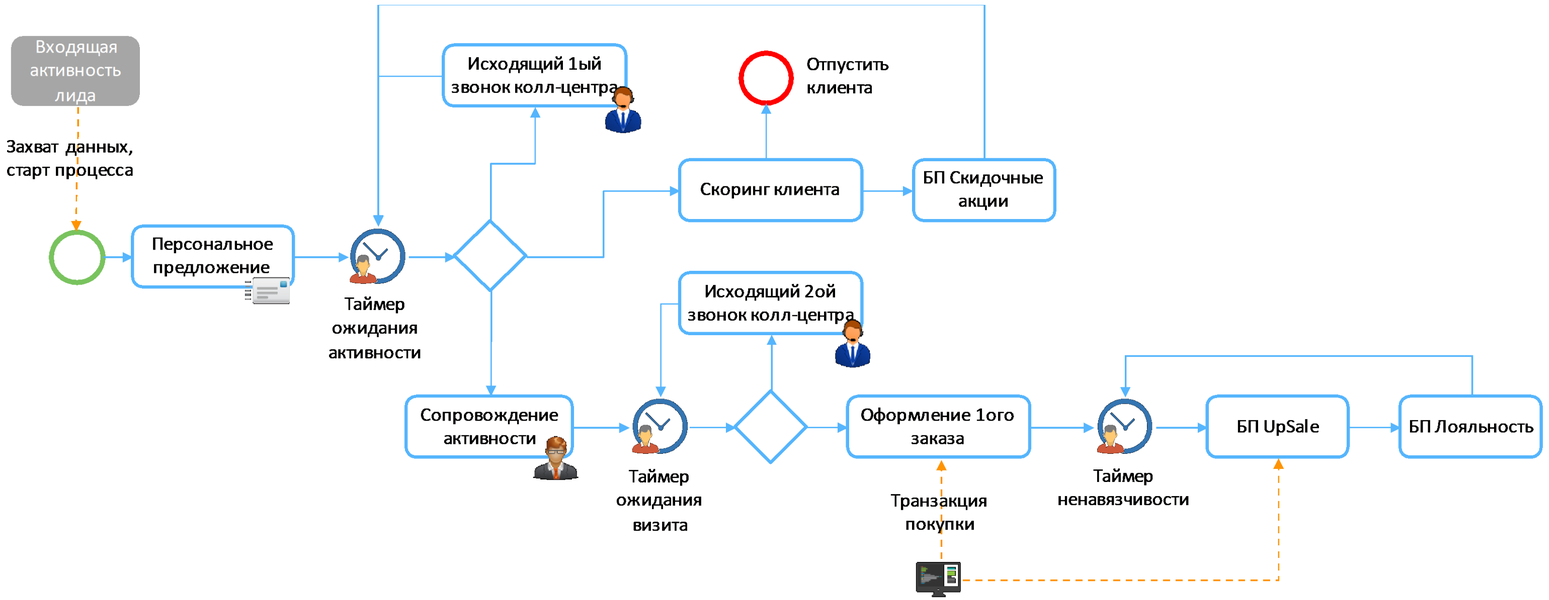
Перемещение данных между ИТ-системами. Ручной ввод данных в ИТ-системы компании.

Перемещение данных между контрагентами. Отправка внешним контрагентам данных (документов) по произвольным каналам.

Создание артефактов (упаковка данных). Формирование документов по шаблонам и в произвольной форме.

Принятие решений по правилам. Проверка исполнения условий, механическое принятие решений на основании формальных правил.

Классификация процессных задач



Актуализация данных. Обновление данных по правилам, например, подтверждение.

Согласование в принятии решений. Контрольная функция, осуществляется в принятии решений, по сути, финальное утверждение.

Согласование в целях контроля качества. Контрольная функция над состоянием процесса с целью повышения качества транзакции.

Обогащение данных экспертное. Создание новых данных на основе компетенций исполнителя.

Принятие решений экспертное. Сложно-формализуемое принятие решений на основе опыта и/или компетенций ЛПР.

Принятие решений на основе контекста. Сложно-формализуемое принятие решений ЛПР на основе данных о состоянии др. процессов.

Исполнение уникальной операции. Исполнение сотрудником какой-либо уникальной операции.

Классификация процессных задач

Создание ценности в процессе	Обогащение данных экспертное	Принятие решений на основе компетенций	Принятие решений на основе контекста	Исполнение уникальной операции	
Обеспечение качества процесса	Актуализация данных	Согласование в принятии решений	Согласование в целях контроля качества		
Стандартизация процесса	Оцифровка данных	Формализация данных	Обогащение данных структурированное	Создание артефактов (упаковка данных)	Принятие решений по правилам
Перевод процесса в исполняемый вид	Инициация процесса	Перемещение данных между сотрудниками	Перемещение данных между ИТ-системами	Перемещение данных между контрагентами	

Применение роботов для разных типов задач

Скрипт.
Автоматизация в ИТ-системе

Сотрудник

Принятие решений по правилам

Перемещение данных между контрагентами

Согласование в принятии решений

Согласование в целях контроля качества

Создание артефактов (упаковка данных)

Принятие решений на основе контекста

Исполнение уникальной операции

Перемещение данных между ИТ-системами

Перемещение данных между сотрудниками

Инициация процесса

Оцифровка данных

Формализация данных

Актуализация данных

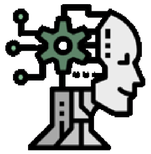
Обогащение данных структурированное

Обогащение данных экспертное

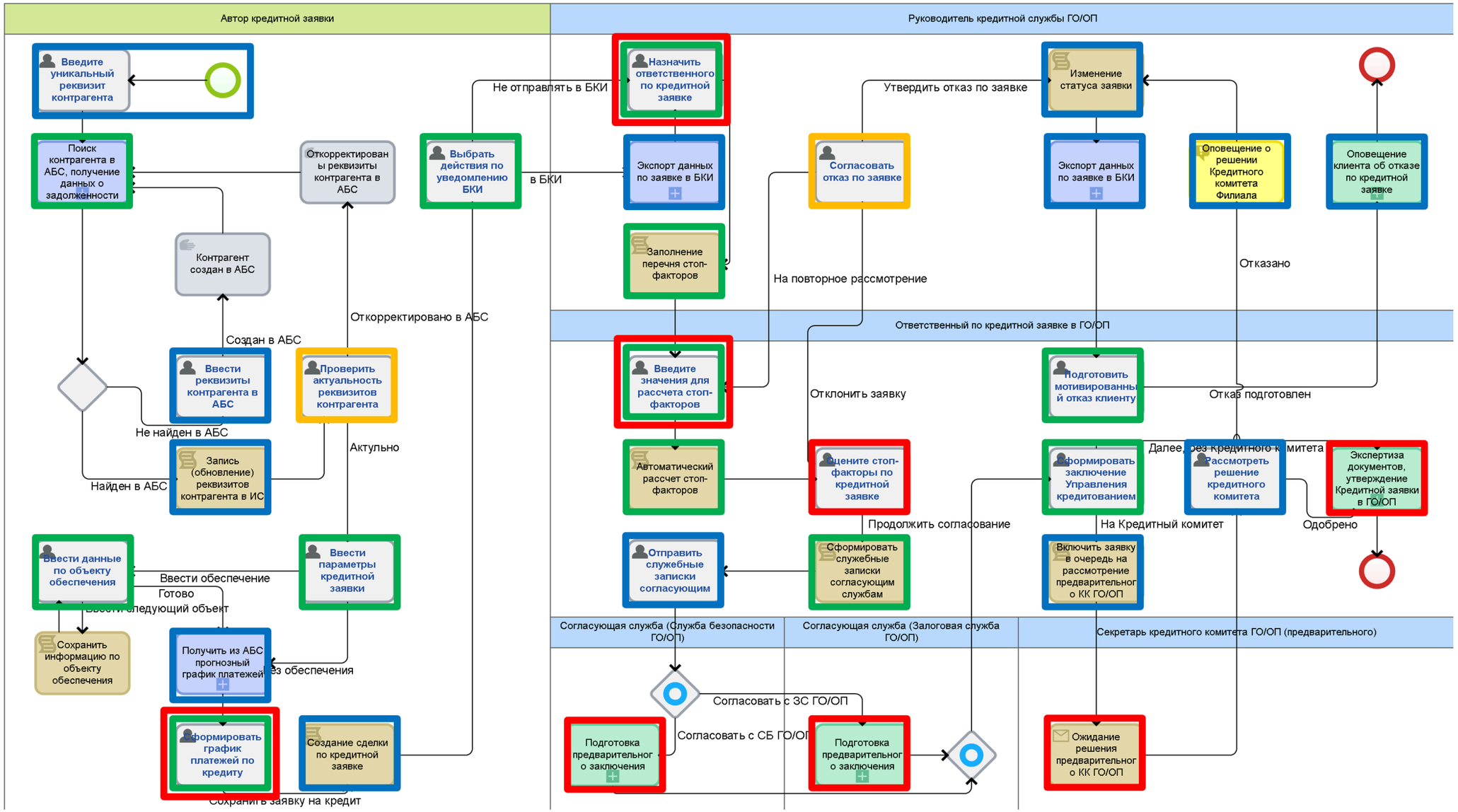
Принятие решений на основе компетенций

RPA

Нейронная сеть или умный чат-бот

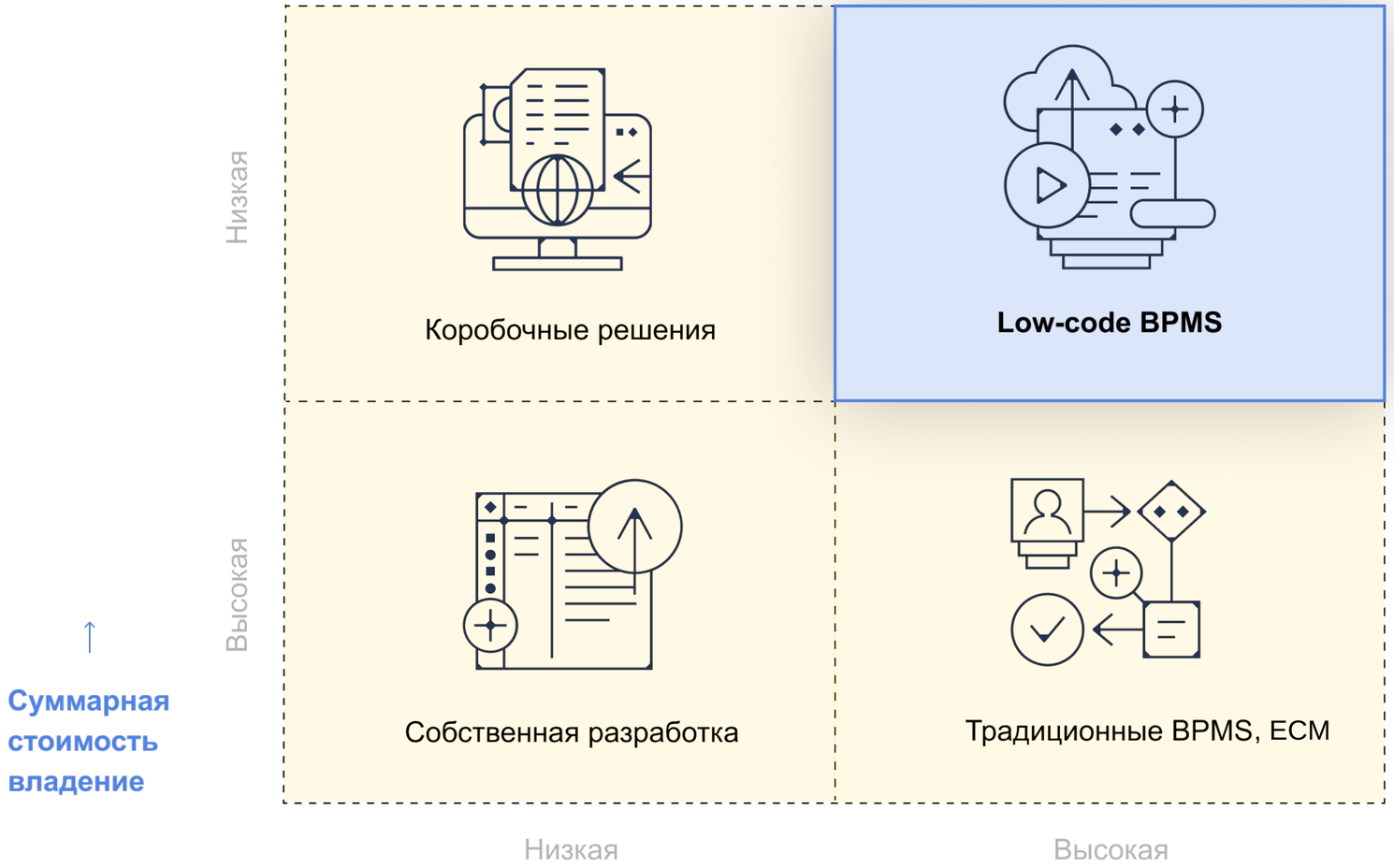


Анализ процесса в разрезе типов задач



Создание сложных информационных систем

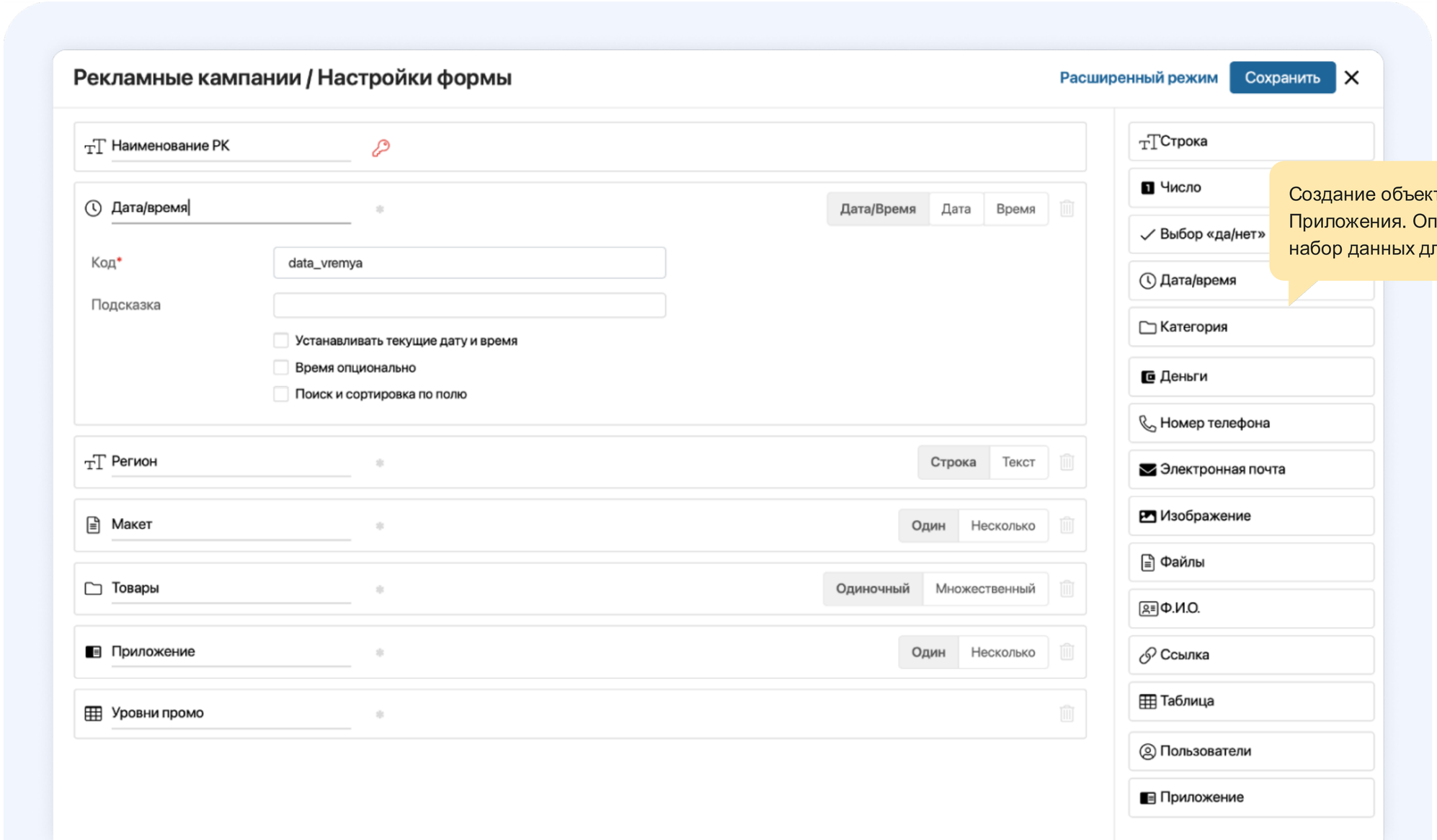
Подходы к автоматизации



Идея Low-code

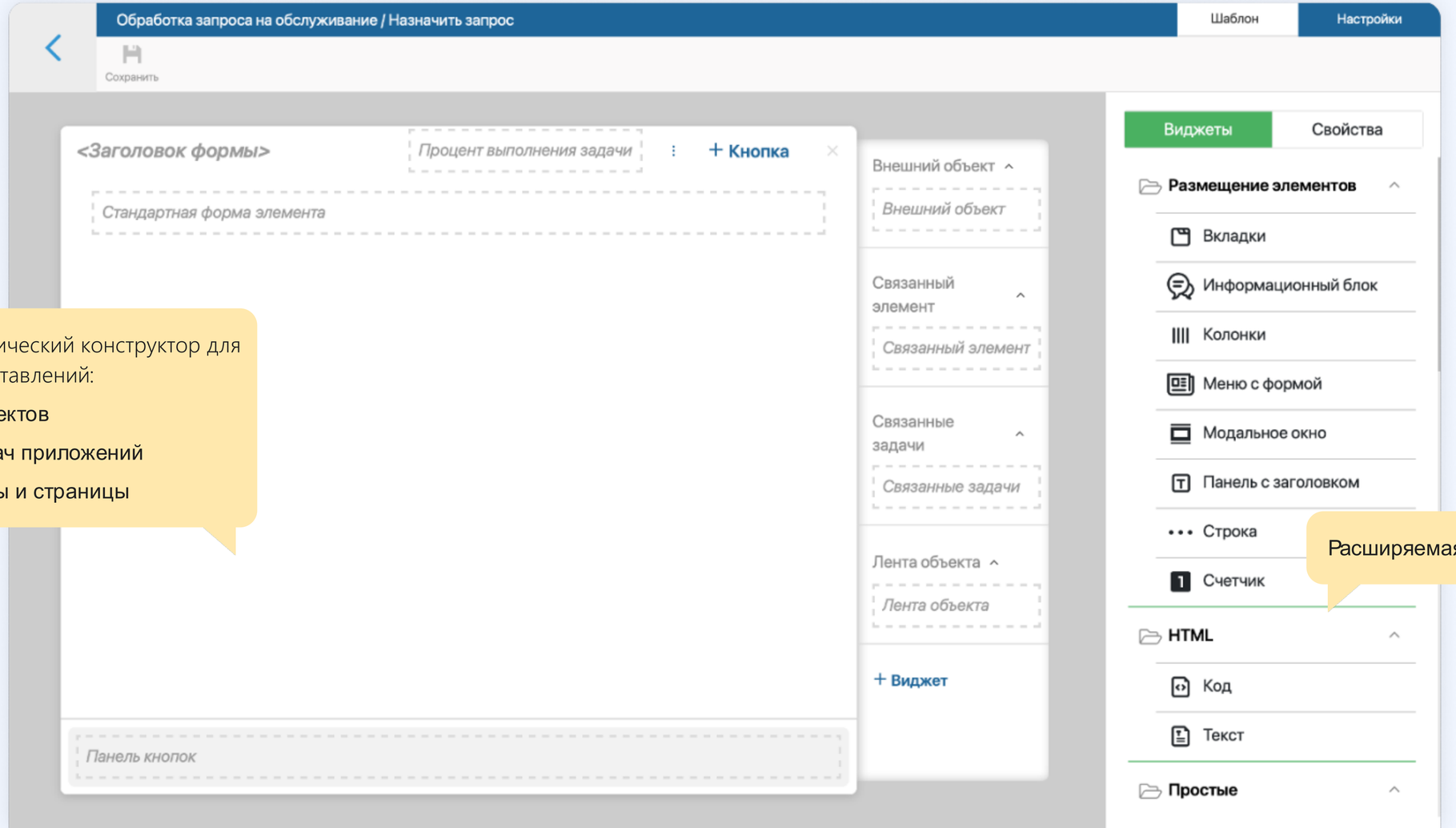
Low-code application platform (LCAP) – это среда разработки, используемая для создания корпоративного программного обеспечения с помощью графических пользовательских интерфейсов и конфигурации вместо традиционного ручного компьютерного программирования.
С помощью Low-code могут создаваться полностью рабочие приложения, при этом сокращается объем традиционного ручного кодирования, что позволяет ускорить «доставку» бизнес-приложений.
Общим преимуществом является то, что более широкий круг сотрудников может внести свой вклад в разработку приложения.

Low-code features. Расширяемая объектная модель



Создание объекта автоматизации внутри Приложения. Определяем произвольный набор данных для объекта

Low-code features. Графический конструктор форм и интерфейсов

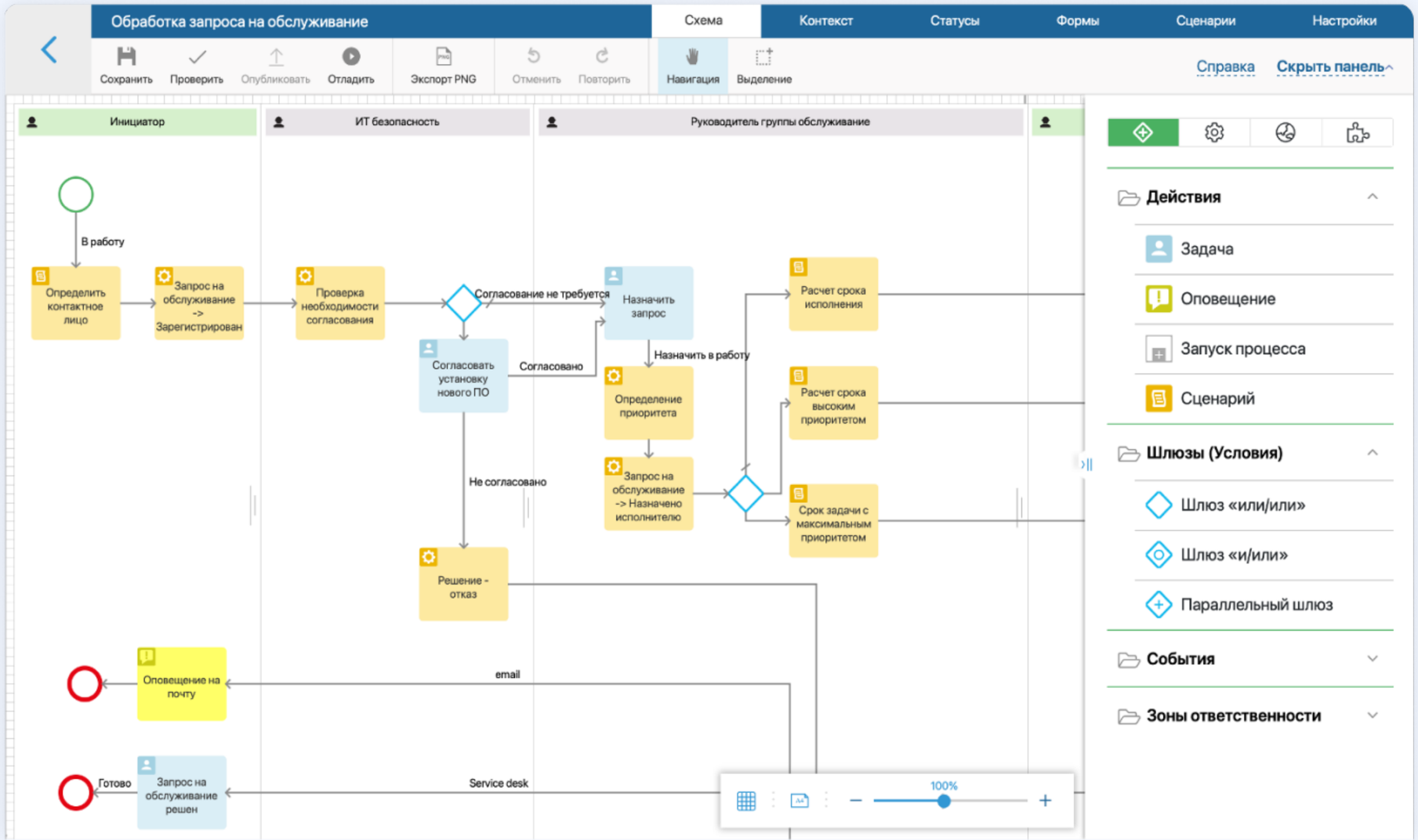


Единый графический конструктор для разных представлений:

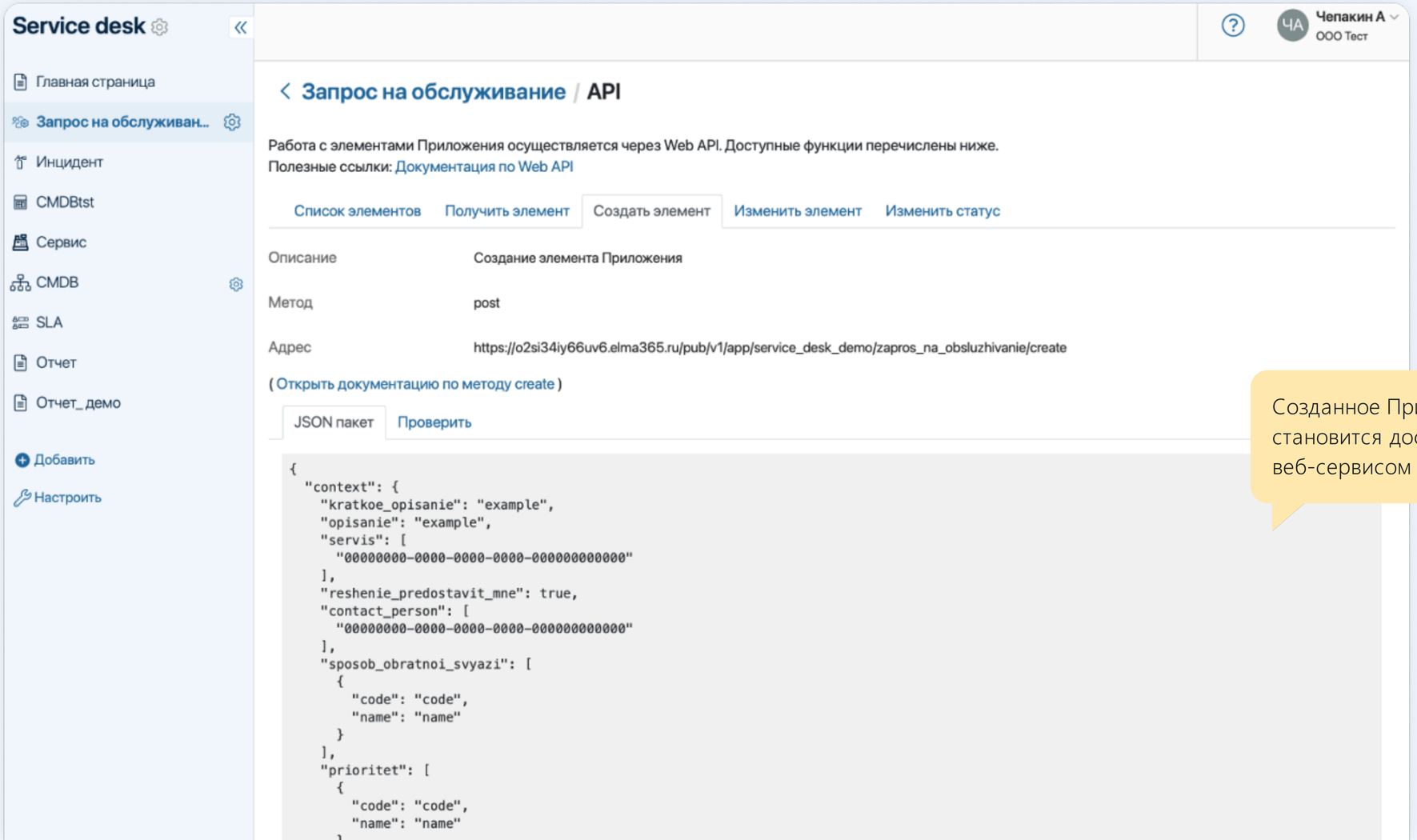
1. Формы объектов
2. Формы задач приложений
3. Интерфейсы и страницы

Расширяемая палитра виджетов

Low-code features. Графический конструктор процессов

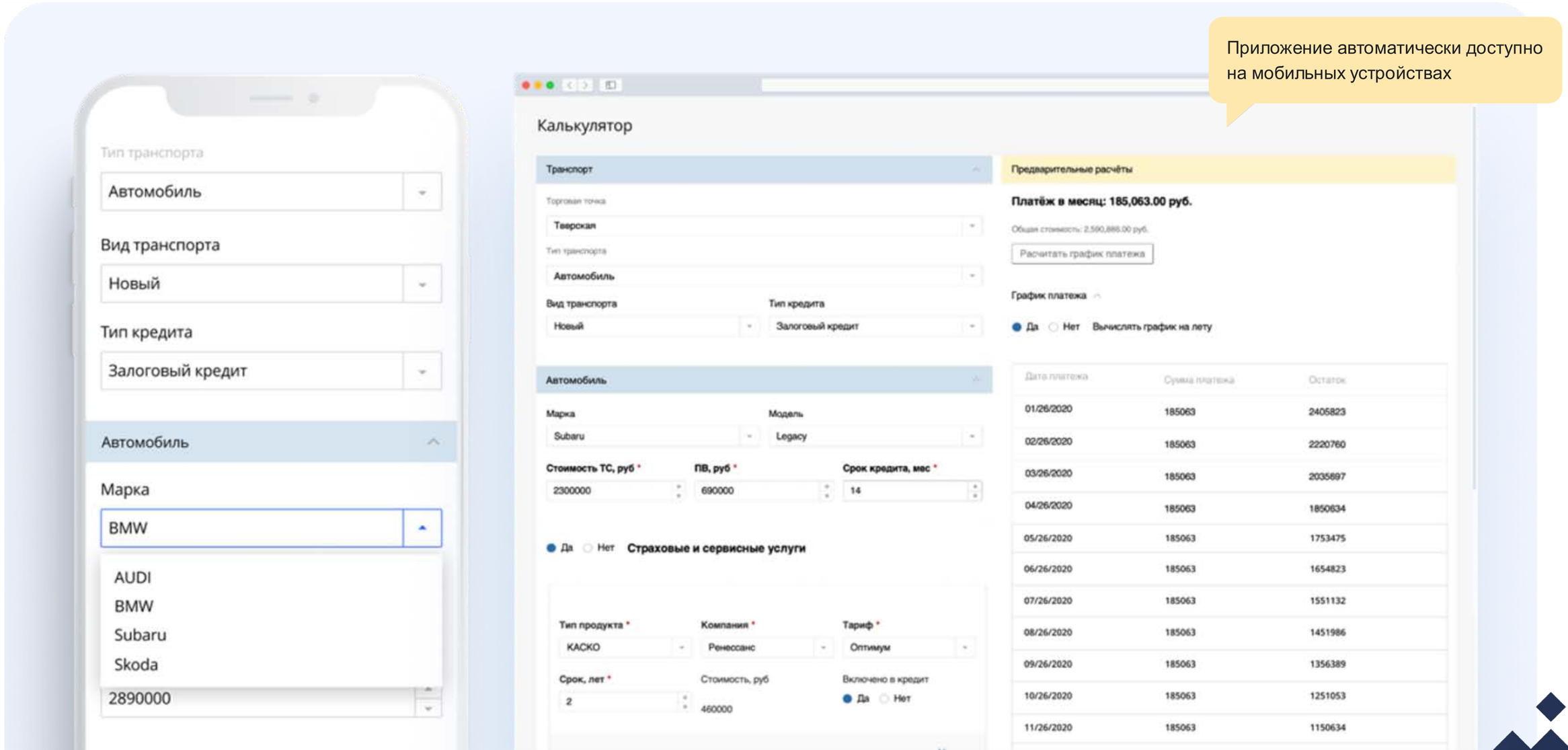


Low-code features. Автоматическая генерация API



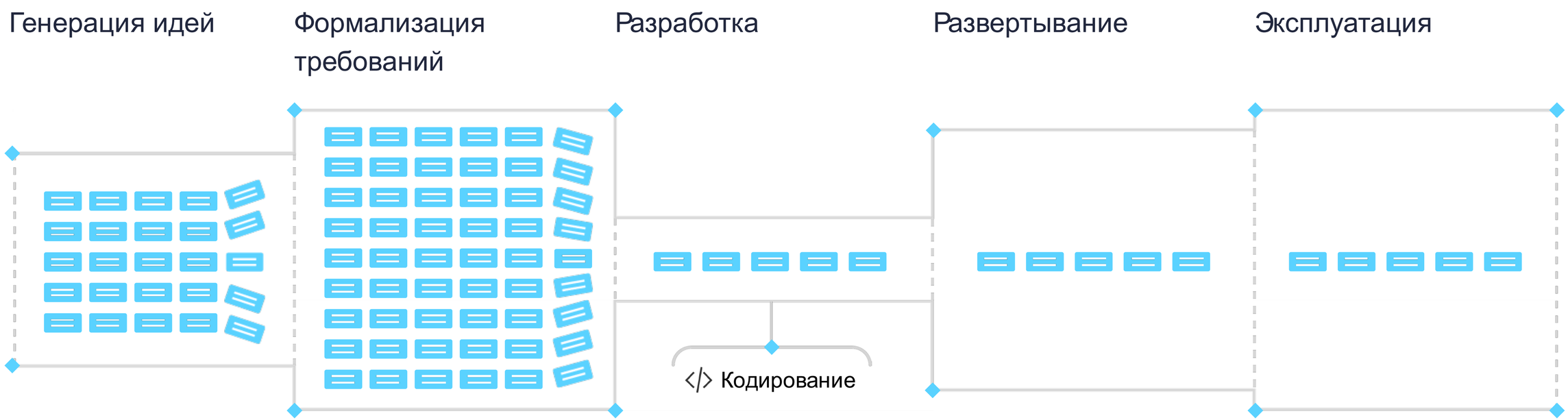
Созданное Приложение автоматически становится доступным для запуска извне веб-сервисом

Low-code features. Адаптивное мобильное приложение



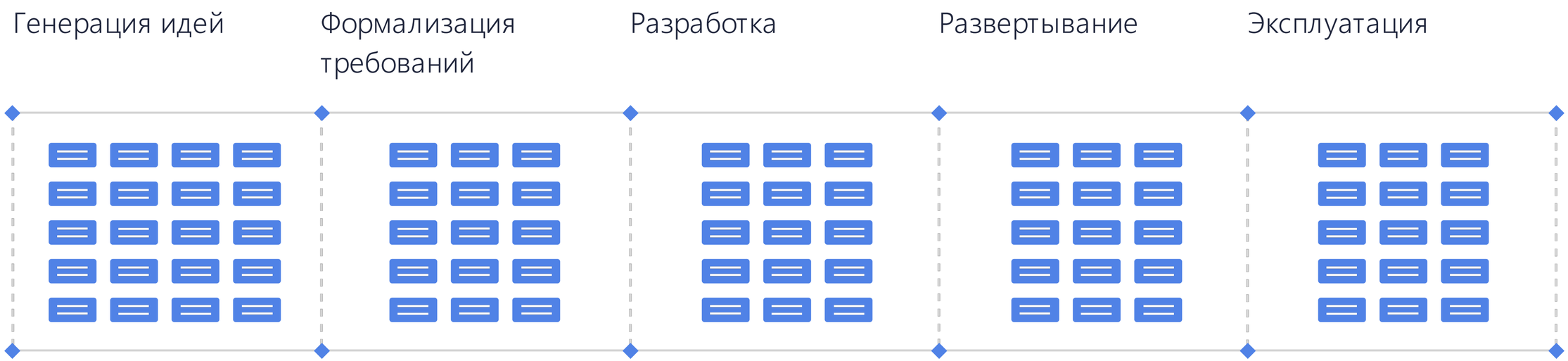
Ценность Low-code BPMS

Low-code для бизнеса



Традиционная разработка корпоративных приложений

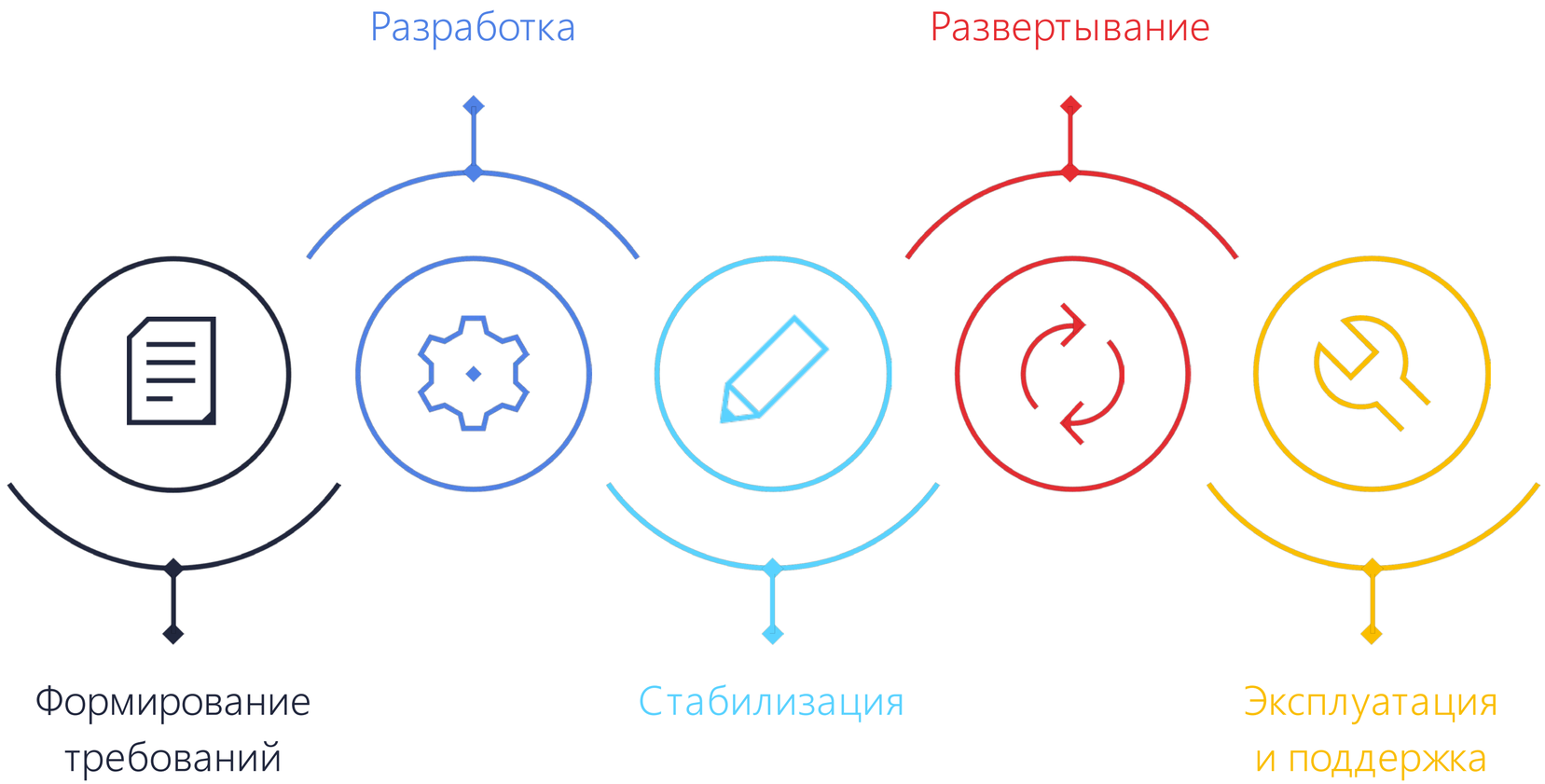
Low-code для бизнеса



Разработка Low-code приложений

Снижение бэклога ИТ создает новое качество разработки, теперь бизнес получает возможность тестировать свои гипотезы в режиме реального времени

Трансформация процесса разработки

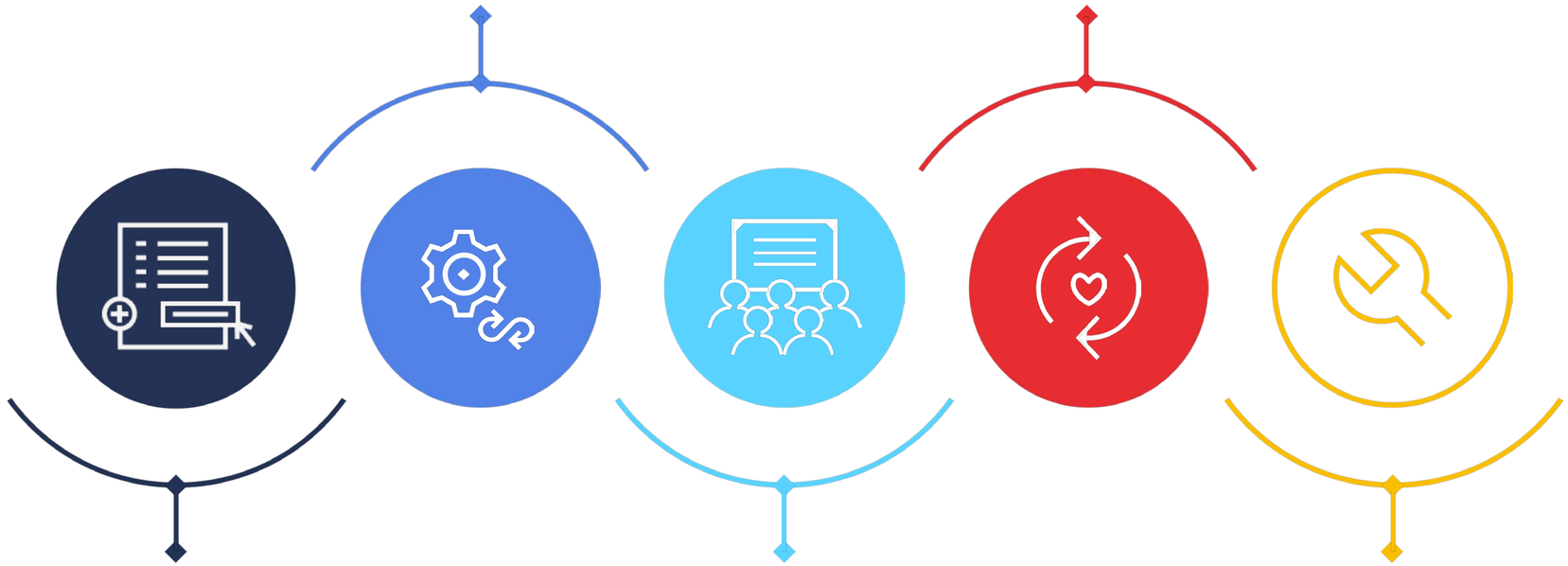


- Time-to-market новых решений или MVP = 2 недели

Трансформация процесса разработки

Agile разработка

Мягкое развертывание



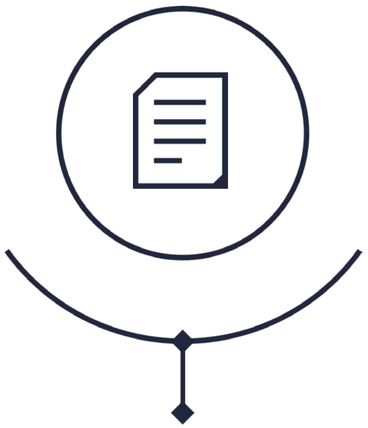
Прототипирование

Демо в режиме реального времени

Эксплуатация и поддержка

- Time-to-market новых решений или MVP = 2 недели

Формирование требований



Формирование требований

Классический подход

- Продолжительная и трудоёмкая формализация требований
- Необходимость оставлять пробелы в требованиях с определением «по ходу проекта»
- Серьёзные требования к аналитикам, работающими над тех. заданием

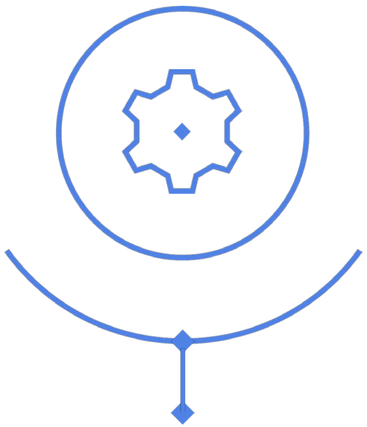
Low-code

- Требования формируются быстро в лайт–формате
- Подготовка прототипа за несколько дней
- Обсуждение требования на прототипе облегчает понимание Бизнеса и ИТ

Формирование требований

видео

Разработка приложения



Разработка

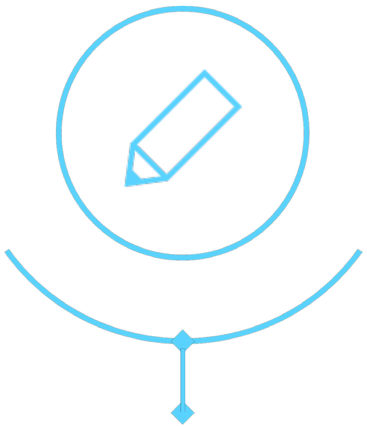
Классический подход

- Высокие требования к команде разработки
- Уточнения и пересогласования требований во время разработки увеличивают срок выхода приложения
- Ротация команды разработки увеличивает сроки и стоимость

Low-code

- Создание приложения без кодирования
- Снижение требований к составу команды
- Полное соответствие agile-подходу к разработке – движение спринтами с детализаций требований «по месту»

Стабилизация



Стабилизация

Классический подход

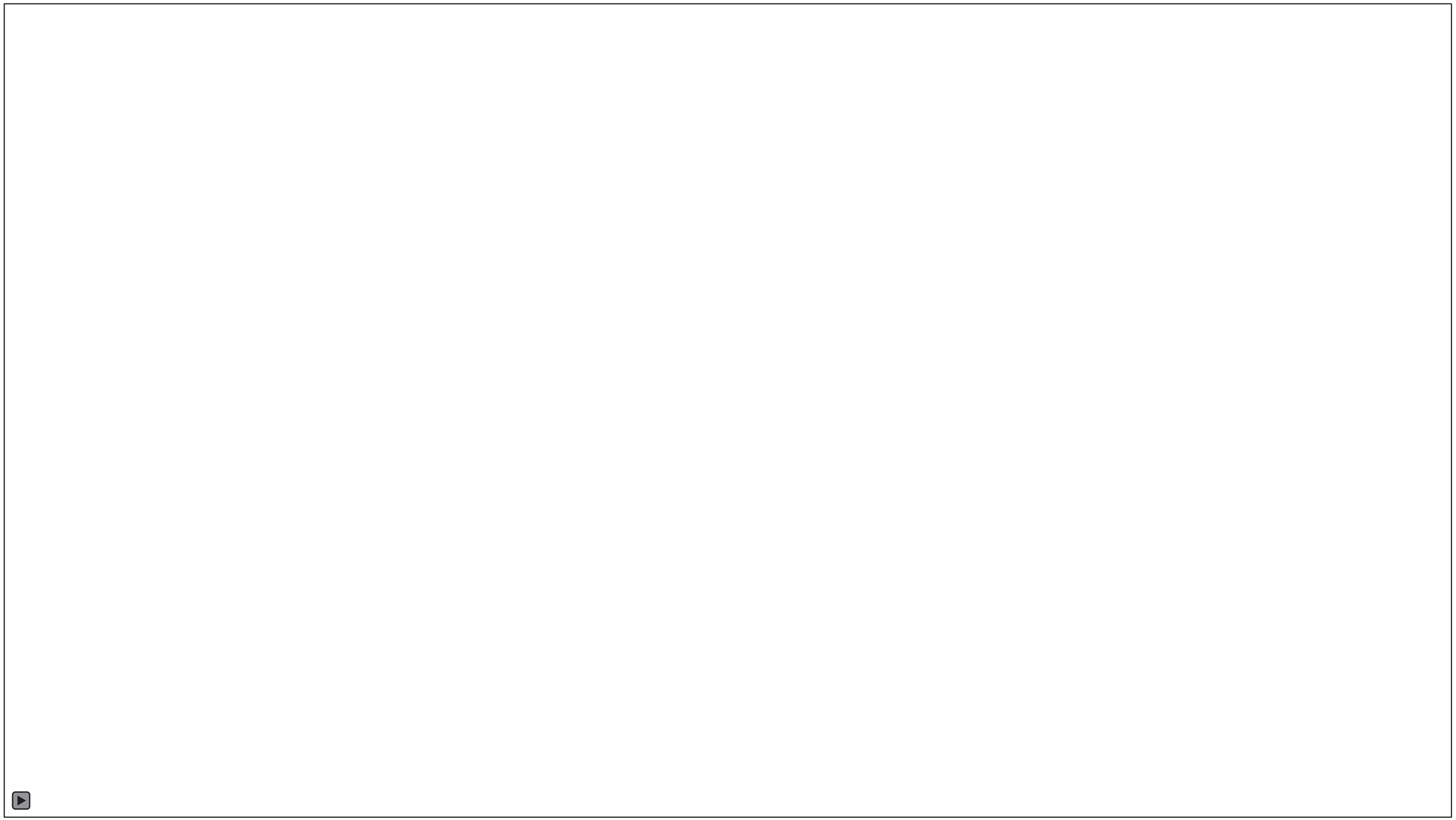
- Устранение замечаний вероятностно приводит к фундаментальной переработке приложения
- Изменения требований и тех. задания по инициативе бизнеса в момент демонстрации финального результата
- Сложность сдачи интерфейсной части и удобства использования, т.к. эти аспекты часто откладываются «на потом»

Low-code

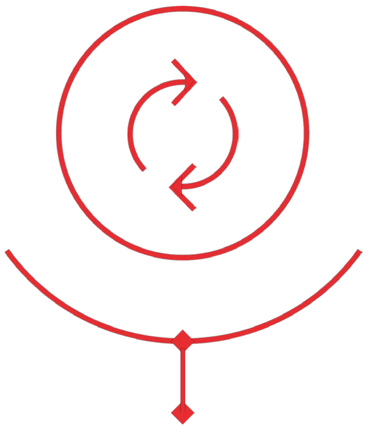
- Быстрые демо-встречи с бизнес-заказчиками
- Внесение изменений и улучшений «на лету»
- Проработка и адаптация интерфейсов по запросу бизнеса

Стабилизация

видео



Развертывание



Стабилизация

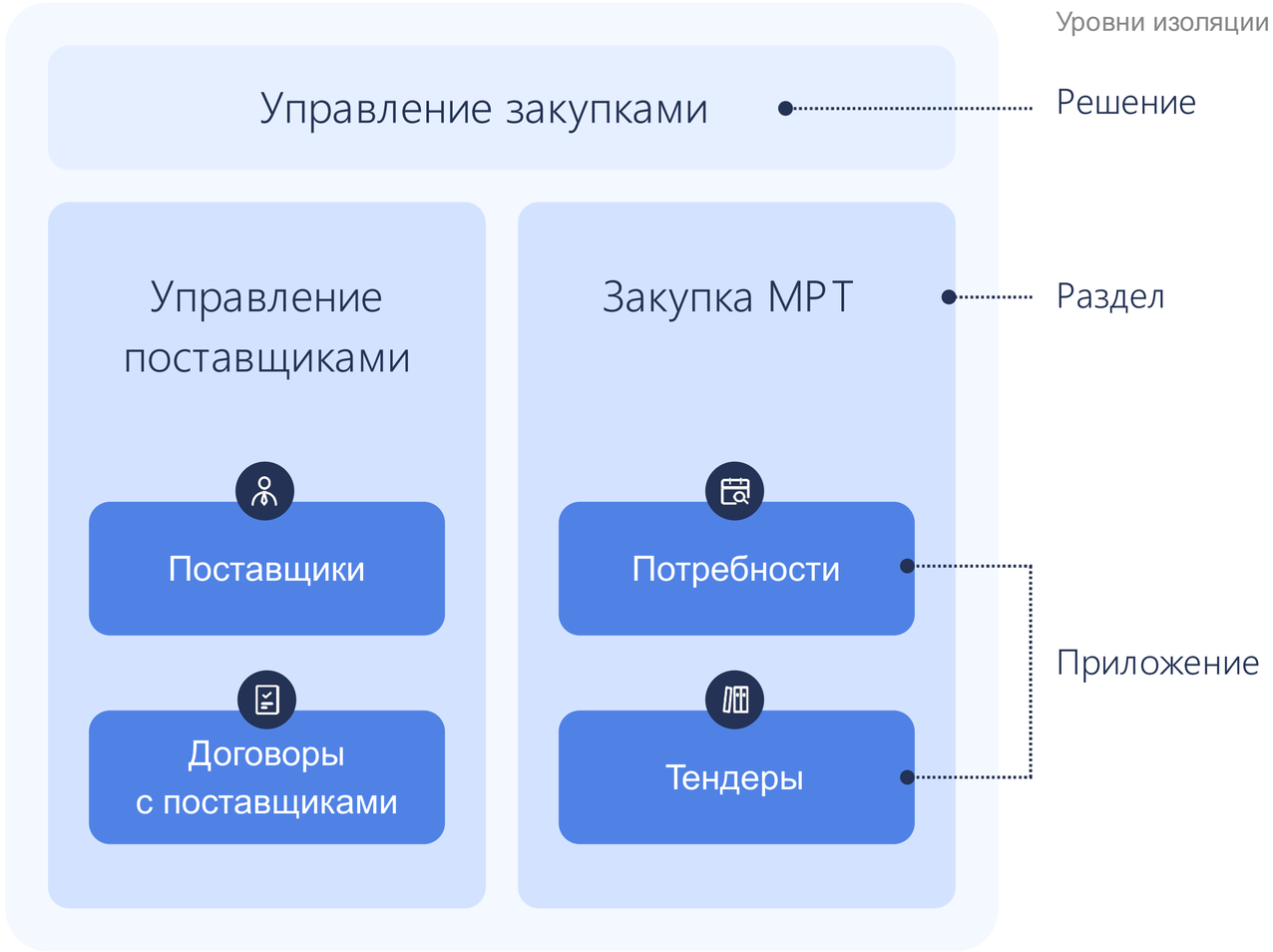
Классический подход

- Сложная процедура управления обновлениями и изменениями
- Запуск нового приложения порождает риски сопряжения с текущим ИТ-ландшафтом
- Дополнительные усилия и меры для обеспечения непрерывности работы приложений

Low-code

- Архитектурная изоляция приложений
- Развертывание приложений без остановки системы
- Разделение сред разработки, тестирования и эксплуатации и быстрый перенос приложений между ними

Развертывание



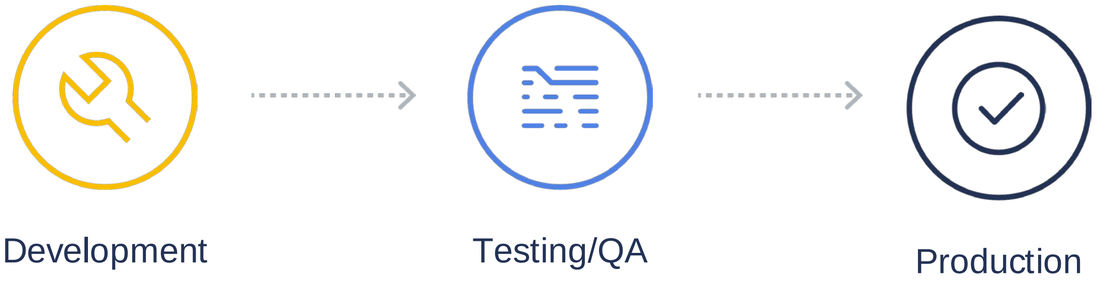
Работа системы 24 на 7, не требуется перезагрузка при внесении каких-либо изменений

Готовые приложения могут быть легко экспортированы и импортированы

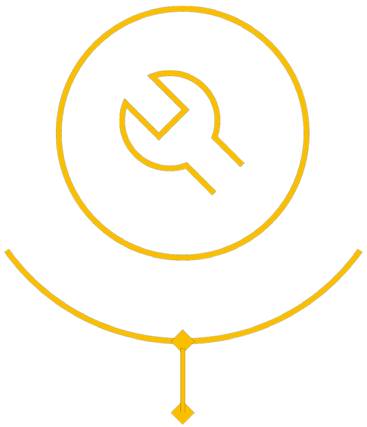
Приложения и разделы изолированы не друг от друга. Изменения внутри приложения не касаются окружения

Развертывание

Приложение	Приложение	Приложение	Приложение	Приложение
Контрагенты	Расчёты с поставщиками	Управление категориями	Договоры закупки	Заказы на закупку



Эксплуатация и поддержка



Эксплуатация и поддержка

Классический подход

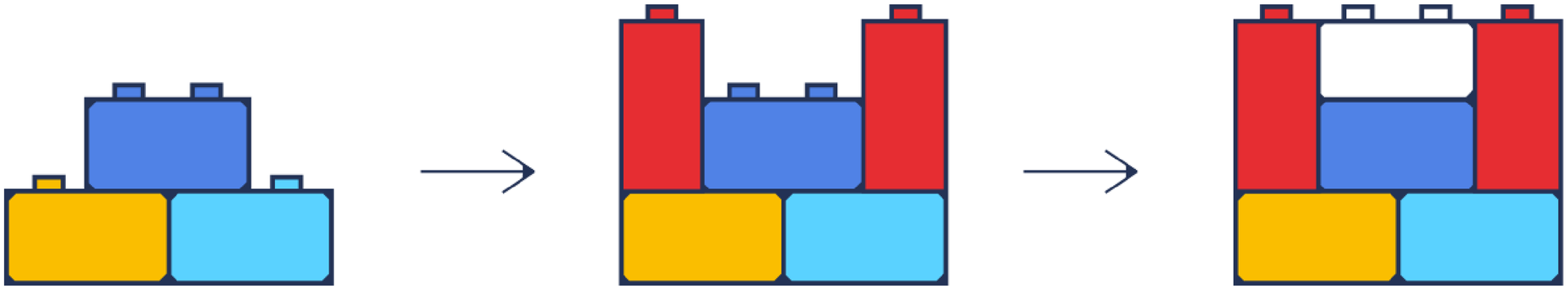
- Дополнительные меры и усилия по обеспечению доступности приложений
- Значительные расходы на масштабирование при увеличении нагрузки на приложение
- Идеи бизнеса по развитию приложения формируют бэклог в ИТ

Low-code

- Оптимальное использование вычислительных ресурсов
- Быстрое масштабирование мощности при высоких нагрузках
- Неограниченные возможности разработки

Место для HARD-разработки

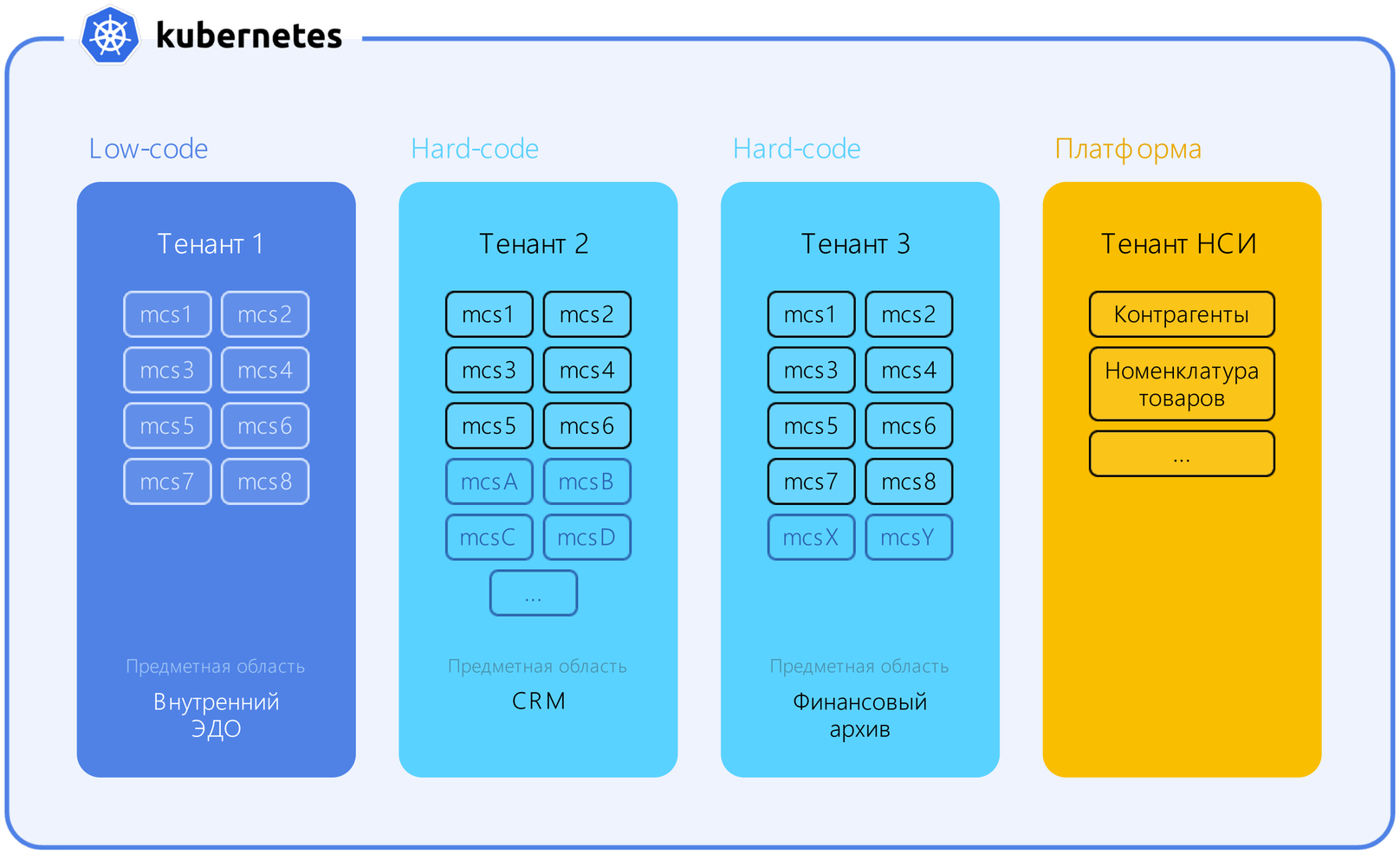
Принципы: Легкость входа / Многослойность



ELMA Low-code BPM обеспечивает легкость входа – это значит, что приложение может быть создано быстро и просто. При этом, это приложение может усложняться до необходимой степени с помощью DevOps-инструментария.

ELMA Low-code BPM позволяет создавать сколько угодно сложные приложения.

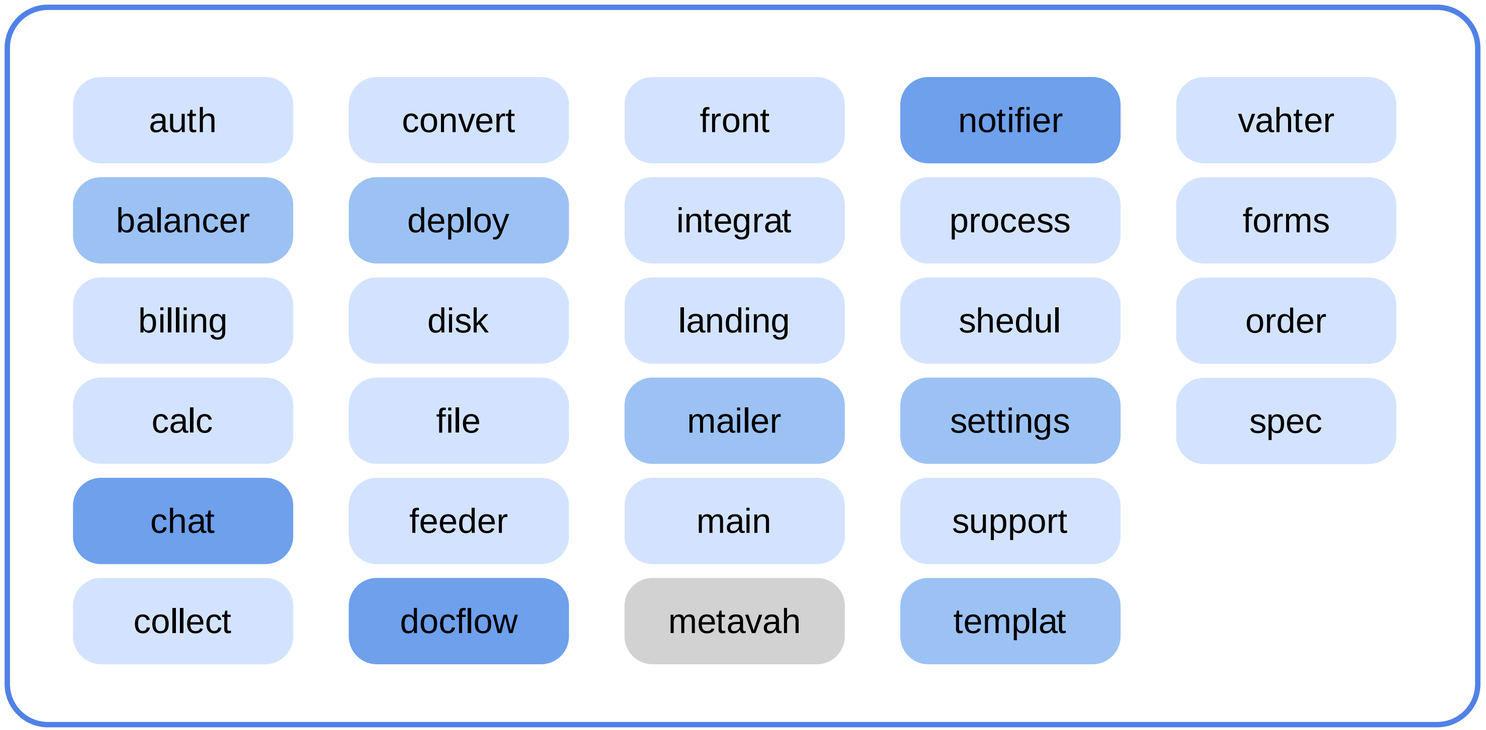
Концепция разработки на ELMA365



ELMA365 объединяет три архитектурных подхода

1. Микросервисная архитектура
2. Мультитенантность
3. Low-code разработка

Микросервисная архитектура

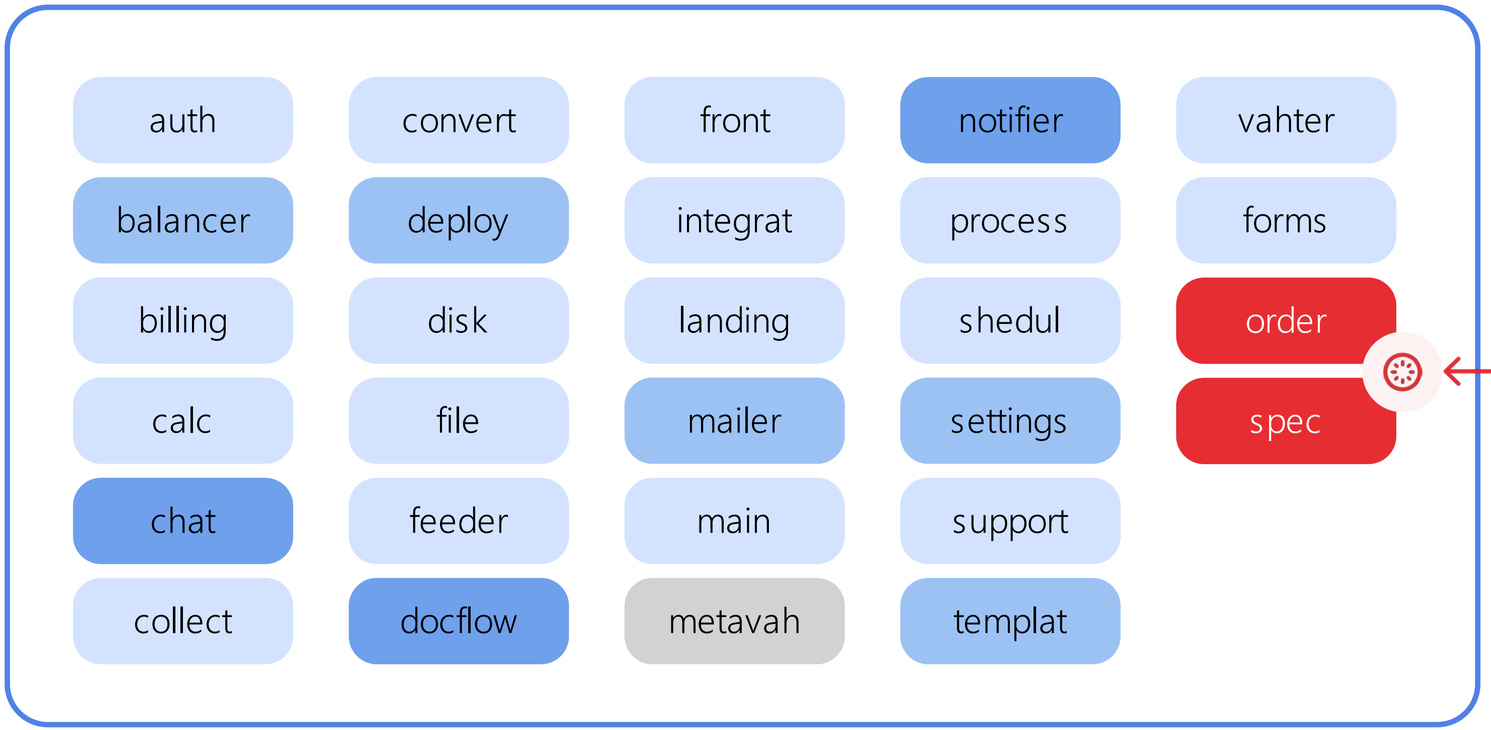


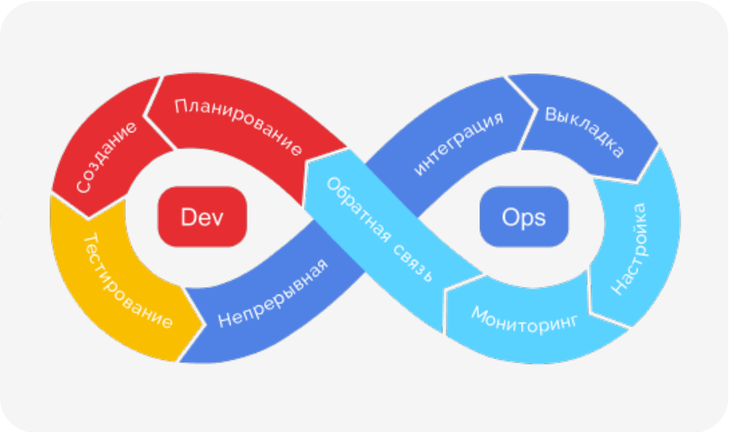
Микросервисная архитектура — вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей — микросервисов, получивший распространение в середине 2010-х годов в связи с развитием практик гибкой разработки и DevOps.

Плюсы подхода

- Функции «по запросу»
- Независимое развертывание микросервисов
- Отказоустойчивость
- Масштабирование микросервисов
- Технологическое разнообразие на уровне языков разработки
- Эффективность разработки в командах
- Низкая связанность и высокая связность

Микросервисная архитектура. Функции по запросу





Независимое развертывание — это концепция, которая позволяет вносить изменения в один микросервис, разворачивать его и предоставлять эти изменения пользователям без необходимости изменять другие микросервисы.

Для возможности независимого развертывания необходимо обеспечить слабую связанность между микросервисами. Между сервисами должны существовать явные, четко определенные и стабильные контракты.

Мультитенант. Асинхронный цикл разработки

Отдельные тенанты могут иметь собственный цикл разработки. Обновление тенанта не влияет на работоспособность соседних тенантов.

Мультитенант. Масштабирование вертикальное и горизонтальное

Мультитенантная архитектура является кардинальным способом снижения стоимости вычислительных ресурсов. Это происходит за счет разделяемой инфраструктуры: ресурсы вычислительного кластера делятся между тенантами.

Для различных тенантов вычислительные мощности выделяются в режиме реального времени.

Создание нового тенанта происходит за считанные минуты и не требует дополнительного развертывания вычислительных мощностей.

Вопросы