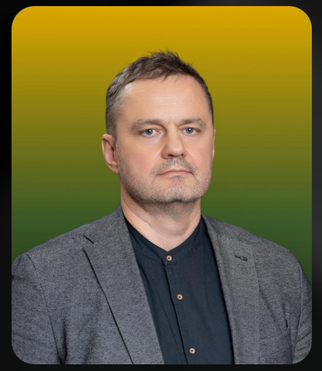
РСХБ

# ГДЕ МОЖНО ИСПОЛЬЗОВАТЬ LCNC



Станислав Тульчинский

Управляющий директор ДИТ АО «Россельхозбанк»
Руководитель блоков развития внутрибанк и кредитный
ООО «РСХБ-Интех»

# Где подходит no-code или low-code?

РСХБ

**Low-code более привлекателен когда:**

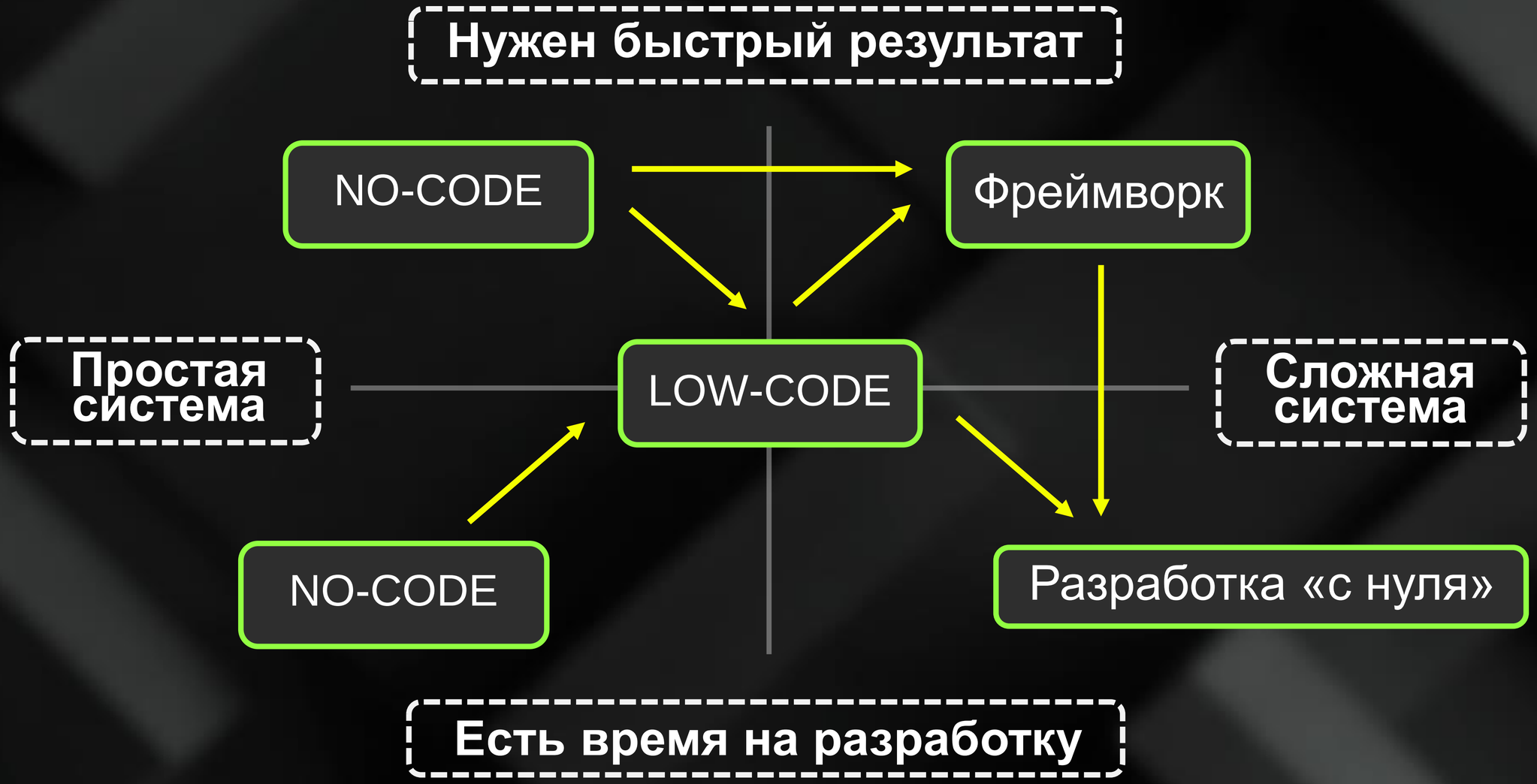
- меньше сложность системы
- нужны быстрее хоть какие-то результаты
- требуется более простой результат
- менее ультимативны требования к MVP

**Возможно последовательное развитие кода:**

- сначала no-code решение, что-то очень простое
- если оно нравится, но есть ограничения no-code, то выкидывают все и делают в low-code
- если и этого мало, то выкидывают low-code и переходят на фреймворк
- в крайнем случае при ощутимых ограничениях фреймворка - собственное решения с нуля

# Где подходит no-code или low-code?

РСХБ



# LC/NC — это не замена классической разработке

РСХБ

LC/NC стратегический инструмент для демократизации разработки и ускорения бизнес-процессов. Их применение целесообразно в следующих областях:

**Внутренние бизнес-процессы и автоматизация рабочих потоков (Workflow Automation)**

**Прототипирование и MVP (Minimum Viable Product)**

**Интеграция данных и создание панелей мониторинга (Dashboards)**

**Порталы для клиентов и партнеров**

# LC/NC — это не замена классической разработке

РСХБ

## Внутренние бизнес-процессы и автоматизация рабочих потоков (Workflow Automation):

- **Что:** автоматизация согласований, заявок, отчетностей, управления документами.
- **Почему LC/NC:** бизнес-пользователи (например, в HR, финансах, юриспруденции) лучше всех понимают свои процессы. LC/NC дает им инструмент для самостоятельной оптимизации без очереди в IT-отдел.

## Прототипирование и MVP (Minimum Viable Product):

- **Что:** быстрое создание прототипа приложения для проверки гипотезы перед выделением бюджета на полноценную разработку.
- **Почему LC/NC:** скорость - за дни или недели можно получить работающий макет, получить обратную связь от пользователей и принять взвешенное решение о дальнейшем развитии.

# LC/NC — это не замена классической разработке

РСХБ

## Интеграция данных и создание панелей мониторинга (Dashboards):

- **Что:** создание простых интерфейсов для агрегации данных из разных источников (Excel, БД, CRM, API) для оперативной аналитики.
- **Почему LC/NC:** позволяет бизнес-аналитикам самостоятельно настраивать визуализацию данных без написания SQL-запросов или кода.

## Порталы для клиентов и партнеров:

- **Что:** разработка форм для подачи заявок, отслеживания статусов, кабинета клиента.
- **Почему LC/NC:** относительно стандартизированные задачи, где важна скорость вывода на рынок и возможность быстрых изменений силами бизнес-подразделения при поддержке IT.

# Смешанные команды

РСХБ

## Гражданские:

- Владелец продукта
- Бизнес-аналитики
- Технологи
- Разработчики и тестировщики пользовательских приложений
- Полуторная линия сопровождения
- Частично третья линия

## Профессионалы:

- Архитектора (корпоративный, солюшен…)
- Скрам-мастер
- Системные аналитики «базовых» систем и интеграции
- Разработчики и тестировщики «базовых» систем и интеграции
- Тестировщики ИФТ, НТ
- Частично третья линия
- Информбезопастность

# Риски для CIO

РСХБ

**"Теневое IT" (Shadow IT):** бизнес-пользователи начинают создавать критичные для бизнеса приложения без ведома IT-отдела. Результат: отсутствие контроля за данными, их целостностью и безопасностью.

**Управление доступом и правами:** в LC/NC-платформах часто упрощенные модели управления доступом. Риск: несанкционированный доступ к конфиденциальным данным из-за ошибки в настройке прав "простыми кнопками".

**Качество данных и целостность:** приложения, созданные непрофессионалами, могут не иметь корректной валидации данных, что ведет к их порче ("мусор на входе — мусор на выходе").

**Сложность аудита и соответствия требованиям (Compliance):** трудно отследить, кто, что и когда изменил в логике приложения, особенно если используется визуальное программирование. Это проблема для 152-ФЗ, PCI DSS и других стандартов.

**Уязвимости платформы:** сама LC/NC-платформа — это программное обеспечение со своими уязвимостями. Атака на платформу может скомпрометировать все созданные на ней приложения.

# Риск зависимости от поставщика

РСХБ

- **"Черный ящик" архитектуры:** логика приложения "запечатана" внутри проприетарной платформы вендора. Перенос на другую платформу или собственную разработку требует практически полного переписывания.
- **Лицензионная политика:** вендор может резко поднять цены, изменить модель лицензирования (например, с "за пользователя" на "за транзакцию"), и у компании не будет альтернатив, кроме как платить.
- **Стратегия вендора:** вендор может прекратить поддержку определенных функций или всей платформы, оставив компанию с неработающим критичным приложением.
- **Нужен рефакторинг кода:** техдолг (в т.ч. Дубли в «картинках») и изменение архитектуры.

# Риск зависимости от поставщика

РСХБ

- Сложно выстроить автоматизированное тестирование
- Маловероятна микросервисная архитектура для реиспользования
- Нужен контроль версий и мердж кода разных разработчиков
- Осложнено построение конвейера CI/CD
- Код внутри low-code **увеличивает требовательность** к описанному выше

# Риски персонала

РСХБ

**Возможности развития силами собственного персонала:**

- **Low-Code:** позволяет профессиональным разработчикам работать значительно быстрее, используя визуальные конструкторы для рутинных задач (UI, базовые CRUD-операции) и добавляя кастомный код для сложной логики.
- **No-Code:** позволяет привлекать бизнес-аналитиков и продвинутых пользователей для создания и поддержки несложных приложений.

**Риски привлечения собственного персонала на такие работы:**

**Для профессиональных разработчиков:**
- профессиональная стагнация
- демотивация

**Для бизнес-пользователей:**
- отвлечение от основных обязанностей
- карьерный тупик

# Возможности реализации требований, выходящих за рамки архитектуры

РСХБ

**Интеграция через API (наиболее распространенный путь):**

- **Подход:** LC/NC-приложение выступает как интерфейсный слой (front-end). Вся сложная логика, вычисления и работа с данными выносится в отдельные сервисы (микросервисы), написанные на традиционных языках программирования (Java, Python, Go). LC/NC-приложение взаимодействует с этими сервисами через REST API или GraphQL.
- **Пример:** требуется сложный алгоритм скоринга. Он реализуется как отдельный микросервис на Python, а LC-приложение просто отправляет ему данные и получает результат.

# Возможности реализации требований, выходящих за рамки архитектуры

РСХБ

**Кастомизация с помощью кода (расширения/плагины):**

- **Подход:** многие продвинутые LC-платформы позволяют добавлять кастомный код (JavaScript, C#, Java) для создания собственных компонентов, подключения к специфичным базам данных или реализации нестандартной логики.
- **Пример:** нужно интегрироваться со специализированным оборудованием через его SDK. Разработчик пишет кастомный компонент на C#, который затем используется внутри LC-приложения.

# Возможности реализации требований, выходящих за рамки архитектуры

РСХБ

**Гибридный подход (LC/NC + Traditional Dev):**

- **Подход:** использовать LC/NC для быстрого создания 80% функционала (стандартные интерфейсы, формы, отчеты), а сложные 20%, выходящие за рамки, разрабатывать классическим способом и интегрировать с основным приложением.
- **Пример:** создание кабинета клиента. Личный кабинет (просмотр заказов, обратная связь) делается на LC. А вот высоконагруженный механизм рекомендаций товаров разрабатывается отдельной командой на Scala и интегрируется через API.

# Другими словами low-code

РСХБ

- подходят вовсе не для любой задачи и не для любой стадии жизненного цикла этой задачи;
- стоят в итоге гораздо дороже, чем про них говорят;
- требуют квалифицированных специалистов;
- не инструмент разработчика;
- инструмент бизнеса: **быстрый MVP для стартапа**;
- инструмент бизнес-аналитика / руководителя проекта: **прототип приложения**;
- инструмент бизнес-аналитика: **прототип бизнес-процесса.**

# Главный вывод для CIO:

**Успешное использование LC/NC** — это не про саму платформу, а про выстраивание процессов управления. Нужно создать экосистему, где LC/NC сосуществует с классической разработкой, используется в правильных местах, контролируется с точки зрения ИБ и минимизирует риски поставщика. Это стратегическое решение, требующее архитектурного подхода и сильного централизованного управления.

# Благодарю за внимание!

РСХБ

TulchinskiySE@rshb.ru

РСХБ

РСХБ | ЦИФРА