ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ТЕЛЕКОММУНИКАЦИИ

РАДЗИЕВСКАЯ Анна
основательница, директор по продукту, Школа обучения No-code,
Россия, г. Москва

МЕТОДИКА РАЗРАБОТКИ IT-ПРОДУКТОВ НА ОСНОВАНИИ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ NO-CODE

Аннотация. *В статье представлена авторская методика разработки IT-продуктов на основании применения технологии No-code, которая позволяет создавать продукт (веб-сайты, мобильные приложения, автоматизации и др.) существенно дешевле и быстрее. В итоге применение No-code способствует росту показателей эффективности деятельности субъектов предпринимательской деятельности.*

Ключевые слова: *No-code, IT, разработка, технологизация, программирование, бизнес.*

Что такое No-code и почему он появился

Мы живем в эпоху, когда любому бизнесу нужно присутствие в онлайн, т.е. необходимы IT продукты для работы с клиентами и функционирования бизнеса. Для того чтобы разработать бизнесу под себя IT продукты, необходимо привлекать команду разработки, стоимость которой от 1,5 млн руб./месяц, а сроки разработки от 6 мес. до нескольких лет. Соответственно ранее позволить такие стоимости и сроки мог только средний и крупный бизнес, у которых есть достаточно кол-во средств для инвестирования в разработку.

В то же время стартапам и малому бизнесу также необходима разработка, но позволить себе ее могут только единицы, да и ждать по несколько лет никто не готов, ведь рынок становится все более динамичным и нужно действовать быстро.

То есть мы приходим в ситуацию, когда спрос на IT-разработку растет, классических разработчиков не хватает на рынке, стоимость такой разработки огромная, а скорость очень низкая и становится неконкурентной в условиях сегодняшней динамики рынка (рис. 1).

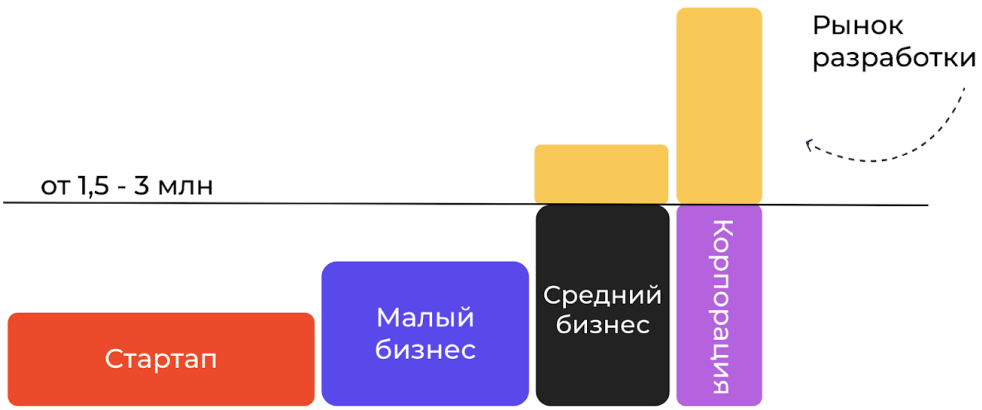

Рис. 1. Рынок классической разработки

Данная проблема подтолкнула к созданию новых разработок без участия IT-отдела. А именно появлению No-code платформ – это программы, благодаря которым можно разрабатывать IT продукты не используя код. Принцип работы, как у визуальных конструкторов, в

которых пользователь взаимодействует с элементами интерфейса, а в это время за него пишется код, но он его не видит и не работает с ним.

No-code платформы позволяют создавать продукт, как веб сайты, мобильные приложения, автоматизации и др. в десятки раз дешевле и быстрее, тем самым открывая целый новый рынок “быстрой и дешевой разработки” и помогают стартапам, малым бизнесам зайти на рынок, а среднему и крупному быстрее развиваться.

Помимо создания нового рынка, No-code стал конкурентом рынку классической разработки и постепенно “отъедает” его часть, т.е. мы наблюдаем случай дизрапшена рынка разработки методом разработки без кода (рис.2).

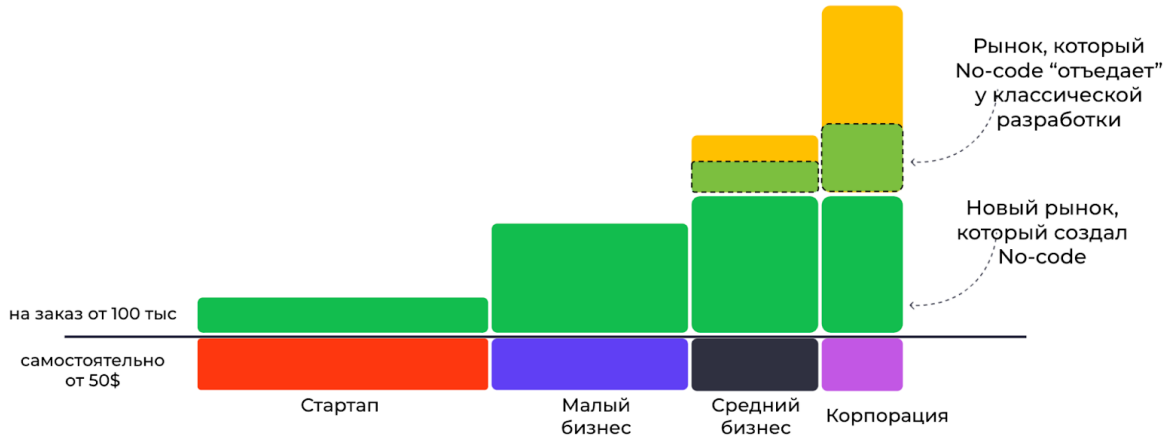

Рис. 2. Рынок разработки с приходом No-code

Отличия подхода No-code разработки от классической разработки

В классической разработке для того, чтобы разработать комплексно IT продукт от идеи до запуска необходима большая команда, где каждый участник команды имеет конкретную узкую квалификацию и выполняет определенный набор функционала. Таким образом все участники в большинстве своем являются не взаимозаменяемыми (например: продакт-менеджер не может выполнить работу backend-разработчика и наоборот) и отсутствие какого-либо члена команды является критичным для процесса разработки.

Основные члены команды и их функциональные роли в классической разработке.

– бизнес-оунер - формулирует бизнес-требования к задаче/проекту/продукту

– продакт менеджер – разрабатывает логику продукта и контролирует весь процесс по разработке

– UX/UI дизайнер – разрабатывает дизайн страниц, элементов, экосистемы продукта

– фронтенд разработчик - разрабатывает фронтальную составляющую продукта, с чем взаимодействует пользователь

– бэкенд разработчик – занимаются разработкой и поддержкой серверной части приложений

– тестировщик/QA инженер – отвечает за проверку работоспособности продукта, всего функционала

В отличие от классической разработки в No-code разработке не нужна большая команда. Используя No-code платформы, специалист может самостоятельно разработать весь продукт “под ключ” (от идеи до запуска), включая фронтенд, бэкенд, дизайн и т.д., то есть заменить целую команду, которая обычно нужна в классической разработке (см. рис 3). Благодаря такому подходу стоимость разработки сокращается в 10-ки раз, а время в 5-8 раз, что выгодно для любого бизнеса.

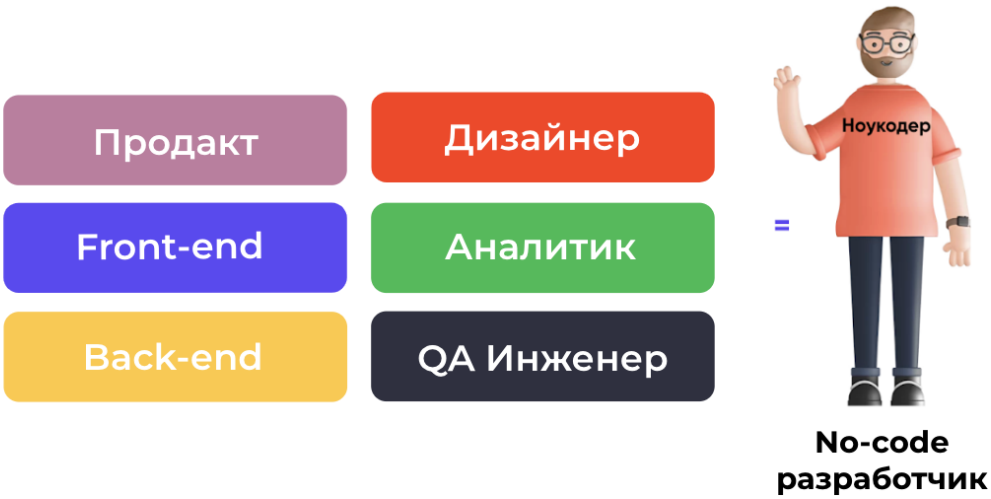

Рис. 3. Для разработки продуктов на No-code не нужна команда, ее может заменить один специалист – No-code разработчик

То, что у ноукода низкий порог входа и что специалист, владеющий им, может разработать самостоятельно продукт, приводит к тому, что в бизнесах не IT-отделы могут сами решать свои задачи, тем самым разработка больше не является бутылочным горлышком для развития бизнеса. Данный тренд очень важный, ведь чем большая доля IT-задач будут решаться вне отдела разработки, тем выше будет скорость развития бизнеса.

Область применения No-code подхода

No-code имеет широкую область применения и используется в бизнесах любого размера: стартапах, малом бизнесе, среднем и крупном, а также для любой сферы бизнеса: ритеил, EdTech, фармацевтика, сервисные продукты, ресторанный бизнес и т.д.

No-code можно использовать как для разработки “клиентских” (внешних) продуктов, так и для внутренних продуктов и процессов компании.

Типы продуктов, которые можно разработать на No-code:

– лендинги и сайты
– веб-приложения
– PWA (progressive web app)
– мобильные приложения
– чат-боты
– интеграции и автоматизации
– AI (искусственный интеллект)
– IoT (Internet of things)
– аналитику и отчеты для веб-сервисов и др.

Где No-code применять нежелательно:

– в финансовых и государственных сервисах – такие продукты обычно требуют более высокие юридические требования к хранению и защите данных, поэтому лучше разрабатывать их на собственных серверах и классическими разработчиками;

– в высоконагруженных проектах, имеющих критическую важность - для данного типа проектов критичным является любой даже минутный сбой системы, поэтому для избежания рисков лучше привлекать классических разработчиков;

– в инфраструктурных проектах – обычно такие продукты важны для крупных компаний с собственной защищенной инфраструктурой. Такого типа решения могут также быть разработаны только в классической разработке;

– в высокотехнологичных продуктах – если вы разрабатываете продукт, который сам по себе является технологией, например, блокчейн, то тут необходимо привлекать классическую разработку, так как No-code не позволяет пока разрабатывать такого типа проекты.

Этапы разработки IT-продуктов, используя No-code

Ниже описана методика, которой стоит следовать, чтобы разработать продукт или задачи в IT, используя No-code. Данная методика разработана Анной Радзиевской, основательницей Code Breakers. Методика наиболее подходит при создании новых продуктов и бизнесов (например, при создании MVP), а также отдельным проектам, которые запускаются в рамках существующих бизнесов. Используя данную методику, можно самостоятельно разработать продукт на No-code, либо подключить других специалистов на локальные задачи (например: дизайнера, продакт-менеджера). Этапы разработки продуктов показаны на рис.4.

Рис. 4. Процесс создания IT-продукта

Шаг 1. Описание идеи продукта/проекта/задачи

Прежде чем начинать разработку необходимо четко сформулировать что, зачем и почему планируется разработать. Данный этап очень важен, т.к. ответив на него, можно прийти к выводу, что на самом деле нет надобности данного продукта.

Необходимо ответить на следующие вопросы:

Для нового бизнеса или продукта:

– что за бизнес/продукт, какую проблему он решает – сформировать гипотезу продукта, которую необходимо проверить;

– кто аудитория продукта, как она сейчас решает данную проблему, провести интервью с пользователями;

– достаточный ли объем рынка, чтобы делать там бизнес;

– тренды роста ниши;

– исследование рынка: есть ли конкуренты и как они оперируют, почему мы делаем то же самое или отличное от них;

– как будет происходить монетизация, кто платящий пользователь, выполнить просчет юнит-экономики, используя предлагаемые метрики;

– определить бюджет на разработку.

Для продукта/задачи в рамках существующего бизнеса:

– в чем суть идеи/задачи – сформировать гипотезу, которую проверяем;

– какая эффективность ожидается – сколько в денежном, временном эквиваленте удастся сэкономить/заработать, а также далее необходимо сравнить с тем, сколько готовы потратить на разработку;

– кто будет пользователем (стейкхолдером), пообщаться с ними и понять их нужды и требования, получить экспертное мнение;

– если разрабатывается сервис внутри существующего бизнеса, то нужно посмотреть есть ли альтернативные решения, которые можно использовать и не разрабатывать самостоятельно. Если есть, то что в них не хватает, почему не можем воспользоваться ими;

– как будет осуществляться монетизация (если применимо);

– бюджет на разработку;

– оценить риски внедрения: невозможность интегрировать с существующей IT-инфраструктурой, требования по защите, хранению данных, внутренние юридические требования компании, политика компании.

Шаг 2. Составление бизнес-требований

Составление бизнес-требований – один из ключевых и важнейших этапов разработки продуктов, ведь от качества его составления будет зависеть вся дальнейшая разработка, потраченный бюджет. На данном этапе необходимо продумать в деталях как будет работать весь продукт, а именно:

– как работает бизнес-модель: продумать всю цепочку процесса (от прихода пользователя до завершения заказа);

– кто вовлечен в использование продукта: клиенты, исполнители, рекрутеры, бухгалтерия и нужен ли им какой-либо инструментарий и функционал;

– в каких странах планируется запускать сервис и на какие языки нужно делать локализацию;

– какие платежные системы планируется использовать;

– какие дополнительные внешние сервисы необходимо использовать, производить интеграции;

Пример, как выглядит описание процессов бизнеса представлен на рис. 5.

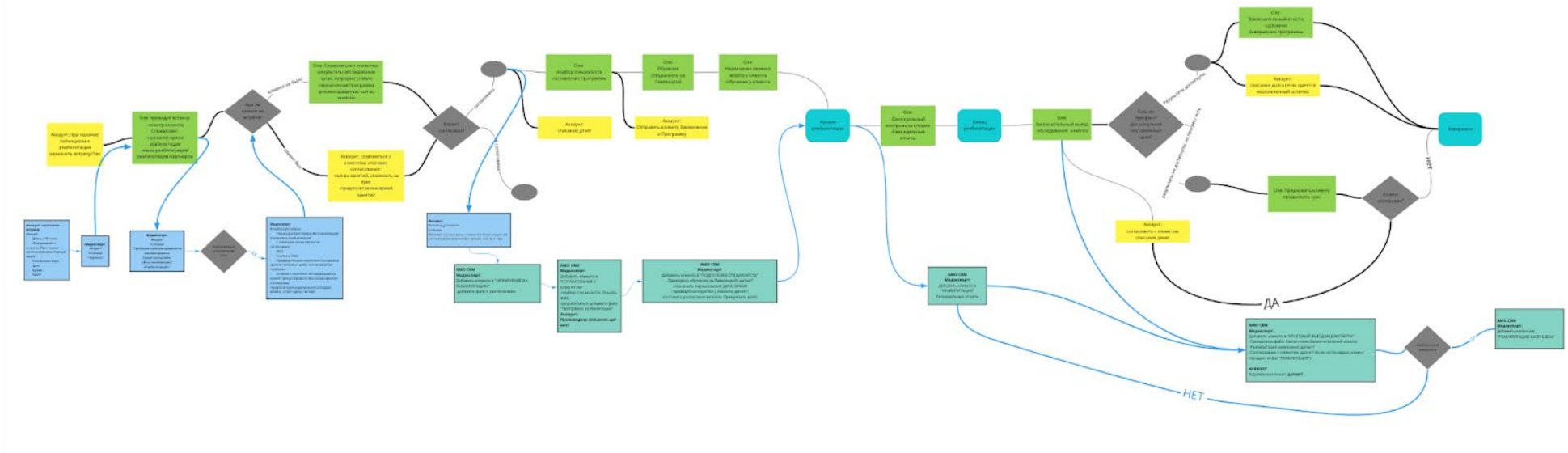

Рис. 5. Схема описания бизнес-процессов в продукте и клиентского пути

Шаг 3. Выделение MVP (минимально жизнеспособный продукт)

На данном этапе необходимо из бизнес-требований сформировать минимальный набор критически необходимого функционала, который позволит проверить бизнес-гипотезу при этом не потратив лишние ресурсы и время на разработку "не главных" функций. Основная задача – проверить жизнеспособность идеи, а не разработать полнофункциональный продукт. Самое важное – сфокусироваться на критическом функционале, без которого сама бизнес-идея перестает работать, т.е. отделить главное от второстепенного.

Что часто является второстепенным, что не стоит разрабатывать в первой версии (MVP):

– множество опций, категорий, дополнительных услуг и т.д., т.е. все, что имеет смысл "дополнительного", "вариабельного" для MVP лучше убрать.

– личные кабинеты не всегда обязательны для MVP, их можно заменить просто формой "Оставить заявку". *Например, если мы разрабатываем сервис по заказу уборки онлайн, то в MVP версии нет смысла создавать личный кабинет клиента, где он будет регистрироваться, видеть статус заказа, предыдущие уборки (это уже стоит делать в версии Продукт 1.0 или 2.0 после подтверждения работоспособности бизнес-модели). Вместо личного кабинета достаточно сделать просто лендинг с описанием услуги и форму заявки с данными клиента – этого будет достаточно для начала.*

– многие процессы лучше оставить в ручном формате и не создавать дополнительных продуктов для других участников бизнес-процессов, а заменить, например на Google Sheets. *Например, вместо того, чтобы создавать личный кабинет клинеров, которые откликались бы на заказы, для MVP достаточно было бы создать Google Sheet с заявками и потенциальными клинерами, который прозванивал бы обученный человек и предлагал выполнить заказ.*

Один из важных принципов данного этапа – создать очень гибкий продукт, который при ситуации неудачной бизнес-модели можно легко пересобрать и сделать pivot.

Шаг 4. Подбор стека инструментов

Понимая, какой функционал необходимо реализовать в MVP, а какой остается для последующих версий продукта, мы можем приступить к подбору стека (набора) No-code инструментов для реализации MVP.

Важно понимать, что мы именно делаем подбор "стека", т.е. не значит, что необходимо использовать только 1 инструмент для реализации, это может быть микс из 2-6 разных No-code платформ и сервисов.

Но важно понимать, чем больше таких сервисов интегрировать, тем больше возникает риск, т.к. система/продукт становятся более хрупкими: если одно из звеньев не сработает, то это может привести к падению всей системы. Поэтому лучше ориентироваться на оптимальное кол-во инструментов в стеке и всегда отслеживать их работу.

Очень важно также учесть возможность подключения к платежным сервисам того стека, который выбран, а также проверить на возможность интеграции с необходимыми внешними сервисами.

При подборе стека важно мыслить не только что нужно для создания MVP, а как потом (при условии успешности бизнес-модели) можно будет масштабировать продукт. Важно, чтобы набор инструментов для MVP не был преградой к масштабированию.

Пример подбора стека инструментов для MVP онлайн магазина представлен на рис. 6.

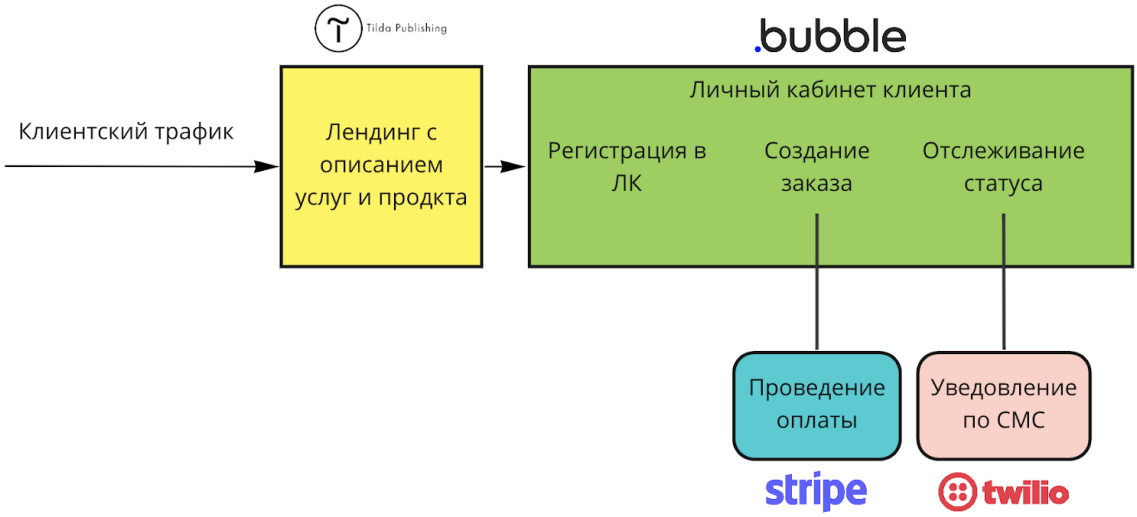

Рис. 6. Схема: Пример стека инструментов для MVP онлайн магазина. Лендинг, на который ведутся клиенты с рекламы выполнен на Tilda, затем регистрация и создания заказа клиентов происходят через личный кабинет, разработанный на Bubble.io, проведение оплаты при заказе происходит через платежную систему Stripe, а при смене статуса заказа приходит оповещение клиенту через Twilio

Шаг 5. Составление продуктово-технического задания

После понимания что именно за функционал будет реализован в MVP и каким набором No-code инструментов, необходимо составить детальное продуктово-техническое задание. Оно включает в себя:

– детальную проработку и описание логики клиентских путей, состава и связей каждой из страниц/экранов,

– описание ролей пользователей и доступов в соответствии с ними, настройки приватности,

– статусную модель заказов/оплат,

– отрисовка вайрфреймов в Figma и прототипирование функционала с целью проверки бизнес-логики,

– определить реализацию интеграций с внешними сервисами.

Данный этап очень важный, так как определяет всю дальнейшую разработку и является картой создания продукта.

Пример, как выглядят вайрфреймы мобильного приложения, представлен на рис. 6.

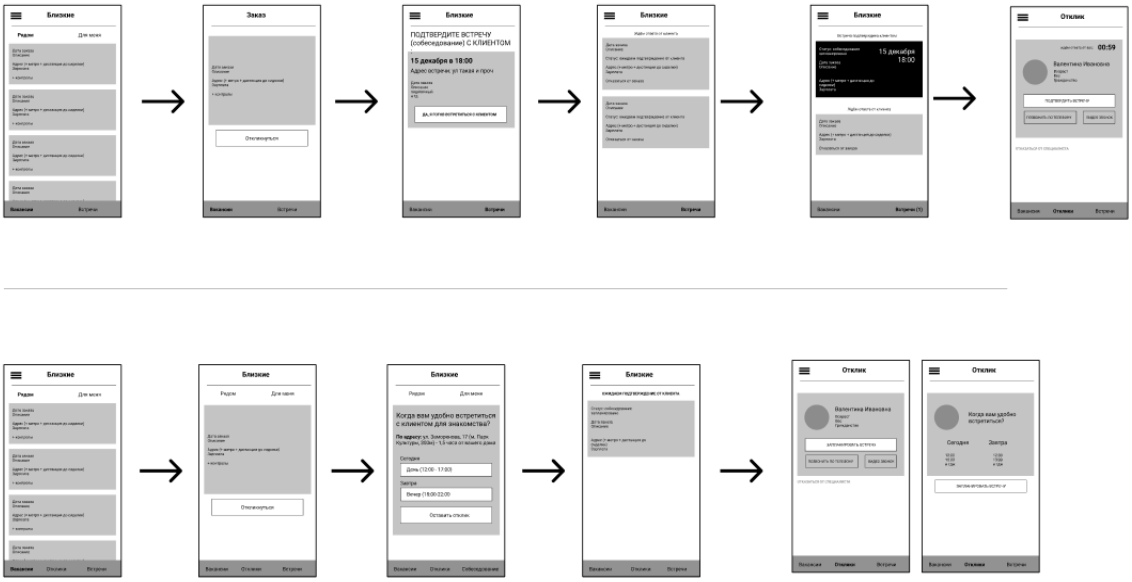

Рис. 7. Пример отрисовки вайрфреймов продукта

Шаг 6. Составление базы данных

База данных включает все, что содержит информацию о продукте и о клиентах. Для интернет-магазинов это каталоги, прайс-листы, данные покупателей, указанные в их профиле и т.д. Все, что хранится в базе данных доступно для изменения и извлечения при необходимости. Система базы данных представляет собой хранилище, куда приложение заносит полученную информацию. В No-code разработке есть базы данных, как внешние (Airtable, Google sheet), так и встроенные внутри приложения (базы данных Bubble, Glide и др. full stack no-code платформ).

Но важно понимать, что база данных требуется не каждого продукта. Если у вас одностраничный сайт (лэндинг), который предназначен для рекламы и ознакомления с товаром или услугой, то создание базы данных вовсе не требуется.

В No-code разработке используются реляционные базы данных. Реляционные базы данных – это базы, где вся информация хранится в таблицах, связанных друг с другом специальными отношениями. Эти отношения позволяют нам извлекать и объединять данные из одной или нескольких таблиц с помощью одного запроса.

Структура база данных представлена тремя уровнями от большего к меньшему (рис. 8):

– база данных – это высокоуровневое понятие, которое означает объединение совокупности данных, хранимых для выполнения одной цели.

– таблица – часть базы данных. Это один из ее компонентов. В одной БД может храниться огромное количество таблиц.

– запись – меньший уровень из всей системы. Это часть таблицы, то есть ее содержимое.

Пример готовой базы данных приложения, показан на рис. 9.

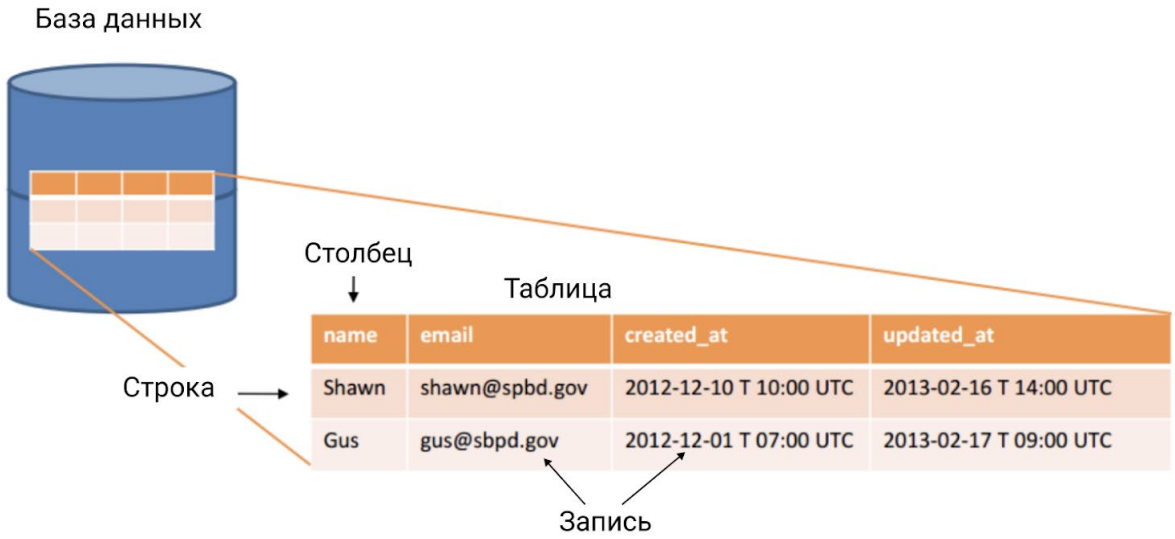

Рис. 8. Как устроена база данных

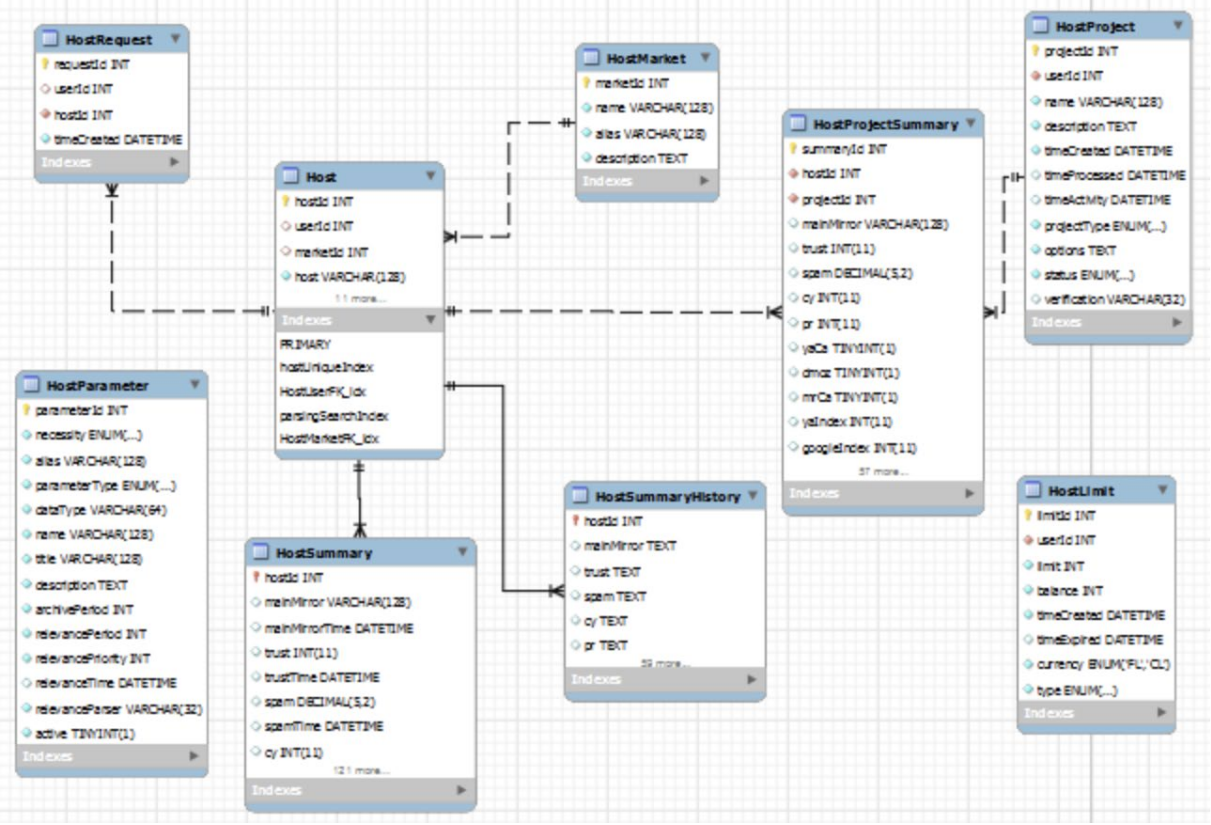

Рис. 9. Пример структуры базы данных продукта

Шаг 7. Разработка дизайна, макетов, прототипирование

Имея продуктовое ТЗ и вайрфреймы, а также зная на каком No-code стеке будет реализован каждый из элементов продукта, нужно отрисовать дизайн там, где есть возможность кастомного дизайна (позволяет создать уникальный дизайн No-code инструмент). Бывает, что в некоторых No-code инструментах нет возможности кастомизировать дизайн полностью, а только изменить какие-либо характеристики: цвет, тени, внешний вид, фон и т.д. Данные ограничения также нужно учесть.

Если выбранный No-code инструмент позволяет сделать кастомный дизайн, то, если вы создаете дизайн самостоятельно можно воспользоваться шаблонами, либо привлечь дизайнера.

На данном этапе также важно продумать всю цветовую гамму и стиль продукта и всех его составляющих. Также нужно выполнить копирайтинг основные текстов (название меню, разделов, страниц и т.д.)

Шаг 8. Разработка: frontend, backend

В No-code подходе разработка frontend и backend (рис. 10) не имеет такого явного разграничения, поэтому выполняется совместно и параллельно. Лучше всего приложение разделить на части и последовательно создавать функциональность каждой из частей: личный кабинет клиента, который разделить на части: главная страница, страница входа, личный кабинет, профиль и т.д.

Процесс разработки лучше всего начинать с создания базы данных. База данных будет связывать все страницы/вкладки и части вашего приложения, т.е. являться основой. Далее уже создается frontend и параллельно задается функциональность backend:

Frontend разработка:

– Создание структуры страниц
– Разработка элементов интерфейса
– Адаптация интерфейса под разные устройства и др.

Backend разработка

– создание функционала и логики для каждого элемента/действия/страницы
– задание внутренних процессов/расчетов системы
– создание регистрации и авторизации пользователей, ролей и настройки приватности/доступов и др.

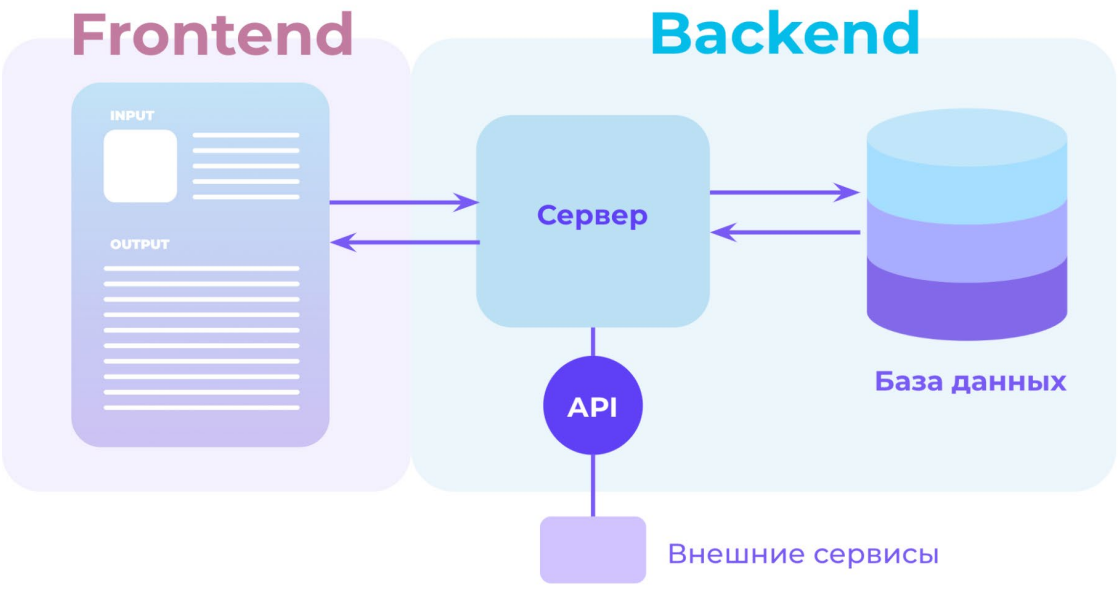

Рис. 10. Структура IT-продукта

Шаг 9. Настройка интеграций с внешними сервисами

Интеграции в No-code инструментах с внешними сервисами обычно бывают уже нативно встроены, либо выполнены при помощи плагинов. Если данные варианты отсутствуют, то необходимо делать кастомную интеграцию при через API (рис. 11). Поэтому важно на этапе подбора стека проверить есть ли открытое API у сервиса, и может ли No-code инструмент выполнить данную интеграцию.

Основные интеграции, которые нужно учесть для продукта:

– платежные системы
– сервисы рассылок: email, sms
– интеграция с внутренними сервисами: crm, slack
– интеграция с существующей инфраструктурой, которая уже разработана
– сервисы аналитики
– сервисы конференций zoom и др.

Важно иметь в виду, что в некоторых интеграциях может понадобиться помощь разработчиков, если нет встроенной интеграции у инструмента и сервиса.

Рис. 11. Внешние интеграции продукта

Шаг 10. Тестирование продукта

При тестировании очень важно проверить работоспособность приложения и корректность функциональности, которая была изначально заложена при составлении продуктово-технического задания. Основные этапы тестирования:

– Составление тестовых сценариев,
– Проведение тестирования,
– Устранение замечаний,
– Повторное тестирование.

Типы тестирования, которые необходимо провести:

1) функциональное тестирование. Под функциональным тестированием понимается проверка соответствия продукта функциональным требованиям, указанным в техническом задании.

2) нефункциональное тестирование. В данном тестировании проверяется:

– удобство использования и эргономика приложения
– как приложение открывается в различных браузерах/устройствах
– адаптация различных устройств
– нагрузка пользователей
– корректность настроек приватности в соответствии с ролями пользователей.

Также важно не забыть протестировать все необходимые интеграции с внешними сервисами, в особенности платежи, передачу данных о заявках пользователей и заказах.

Шаг 11. Запуск продукта на рынок

После проведения тестирования, устранения всех замечаний, необходимо подготовить продукт к запуску на пользователей:

– наполнить приложение контентом (картинки товаров и услуг и др.),
– выполнить корректный копирайтинг текстов,
– добавить пользовательское соглашение, соглашение на обработку персональных данных и др.,
– добавить контакты компании и варианты связи с компанией,
– сделать настройки SEO, если необходимо.

После, важно также проверить командой соответствие всем необходимым первоначальным требованиям и начинать тестирование на пользователей.

После запуска продукта необходимо следить за его работоспособностью и выполнять поддержку, очень внимательно обращать на обратную связь от пользователей и проводить с ними интервью, чтобы понять, совпадает ли те смыслы и идеи, которые вы закладывали при разработке, с тем, как понимает пользователь ваш продукт и использует его.

Масштабирование продуктов на No-code

Для того чтобы при масштабируемости продукта/бизнеса No-code не стал бутылочным горлышком, важно особенно тщательно продумать при подборе No-code стека какие инструменты использовать.

Важно помнить, что чем больше инструментов задействовано, тем более хрупкой становится система и тем больший риск в перебоях работы. Поэтому, при масштабировании продуктов и добавлении с ростом бизнеса более сложного функционала, важно также ставить в приоритет full stack No-code инструменты (на которых сразу разрабатывается frontend, backend и база данных). Именно использование full stack платформ позволяет создавать более кастомную функциональность и избежать рисков интеграций множества платформ.

Также нужно помнить, что, если где-то функциональность No-code инструмента недостаточна, то можно привлечь разработку и расширить функционал.

Переход на классическую разработку

Важно учитывать, что переход стоит делать тогда, когда No-code действительно стал бутылочным горлышком и есть ограниченность в функционале, которая мешает развитию бизнеса.

Для перехода на классическую разработку лучше всего использовать частичную параллельную замещающую разработку. То есть параллельно с работой существующего продукта и процессов начать разработку наиболее критической части продукта на классической разработке. Затем постепенно перевести данные пользователей на новый продукт, разработанный на классической разработке и уже потом заменить сам продукт. Так итерационно можно полностью перевести все части бизнеса с ноукода на код.

Начало перехода обычно начинается с клиентских внешних продуктов, которые требуют больше качества в дизайне, нативных функциях и т.д. Внутренние продукты можно замещать позже.

Но важно помнить, что ноукод довольно мощный и позволяет усложнять функциональность с масштабированием бизнеса, поэтому многие продукты/бизнесы так и остаются на No-code, т.к. бизнесы не сталкиваются с ограничениями No-code и масштабируют продукты без разработчиков.

RADZIEVSKAYA Anna
Founder, Product Director, No-code Learning School,
Russia, Moscow

METHODS OF DEVELOPMENT OF IT-PRODUCTS BASED ON THE USE OF NO-CODE TECHNOLOGY

Abstract. *The article describes the author's methodology for developing IT products based on the use of No-code technology, which allows you to create a product (websites, mobile applications, automation, etc.). significantly cheaper and faster. As a result, the use of No-code contributes to the growth of performance indicators of business entities.*

Keywords: *No-code, IT, development, technologization, programming, business.*