

Профессия:

No-Code / Low-Code **разработчик**



Антон Золотавин

Антон Золотавин

Профессия: No-Code / Low-Code разработчик

<https://litres.ru/74621914>

SelfPub; 2026

Аннотация

Хотите создавать сайты, приложения и автоматизации, но не знаете сложного кода? Эта книга поможет освоить визуальную разработку и подготовиться к первым заказам.

Шаг за шагом вы соберёте сайт на Tilda, кабинет сотрудника в Bubble и автоматизацию в Make. Научитесь создавать формы, работать с данными, настраивать доступ и связывать сервисы. Проверите, что заявки не теряются, повторные события не создают лишних записей, а пользователи видят только разрешённую информацию.

Внутри — 26 практических работ, понятные объяснения, разбор ошибок и готовые шаблоны. Вы узнаете, как выяснять требования заказчика, оценивать стоимость работы, сдавать проект и оформлять портфолио.

Содержание

Глава 1. Что вы сможете создавать и продавать	6
Работа разработчика простыми словами	8
Какие задачи подходят для первого заработка	10
Как устроены инструменты книги	12
Доход без обещаний	14
Как учиться с этой книгой	16
Практика выбора первой услуги	18
Глава 2. Как превратить просьбу заказчика в задачу	20
Разговор о текущем процессе	21
Первая версия системы	23
Требования и критерии приёмки	24
Пользовательские роли	26
Состояния обращения	27
Что нужно получить от заказчика	29
Задание на одну страницу	30
Работа с новой просьбой	31
Глава 3. Веб, данные и события без сложного кода	32
Клиент и сервер	33
Типы значений	35
Записи и связи	36
Ответы и ошибки	39

Время и порядок	40
Практика чтения данных	41
Структура страницы	43
Сборка стандартными блоками	45
Текст и смысл кнопок	47
Изображения и права использования	49
Типографика и композиция	51
Форма обращения	52
Сообщения об ошибках и успехе	54
Подключение приёмника	55
Первая публикация	57
Проверка содержания	58
Глава 5. Мобильная версия, домен и запуск сайта	59
Мобильная проверка по действиям	60
Размеры и доступность	63
Конец ознакомительного фрагмента.	64

Антон Золотавин

Профессия: No-Code / Low-Code разработчик

Профессия: No-Code / Low-Code разработчик

Антон Золотавин

Глава 1. Что вы сможете создавать и продавать

Небольшой компании нужен сайт, на котором посетитель выбирает услугу и оставляет заявку. Сотруднику нужен экран, где можно увидеть новые обращения и изменить их статус. Руководителю нужно понимать, какие заявки ещё не обработаны. Для такой системы не обязательно начинать с изучения языка программирования. Сайт можно собрать из готовых блоков, приложение — из экранов и правил, а передачу данных — из связанных модулей. Именно такими задачами занимается No-Code / Low-Code разработчик.

В этой книге вы соберёте учебную систему для вымышленной мастерской «Точный ремонт». Она принимает заявки на ремонт домашней техники. Мы не будем подключать реальные платежи и собирать сведения настоящих клиентов. Для упражнений достаточно вымышленных обращений, тестовых учётных записей и небольшого каталога услуг. Такой проект позволит показать работодателю или заказчику не только красивый экран, но и работающий процесс.

У системы будет три части. На Tilda вы сделаете сайт с описанием услуг и формой. В Bubble создадите кабинет сотрудника, где заявки можно искать, просматривать и обрабатывать. В Make настроите получение событий, проверку

полей, защиту от повторов и уведомления. Затем подготовите мобильный вариант кабинета и разберётесь, когда нужен отдельный нативный проект. Эти части можно изучать по очереди; пытаться освоить три редактора в первый день не требуется.

Работа разработчика простыми словами

No-code означает сборку программного продукта в визуальном редакторе без самостоятельного написания основной программы. Low-code предполагает, что часть задачи решается небольшими фрагментами кода, выражениями или запросами. Между подходами нет строгой бытовой границы: один и тот же проект может состоять из готовых блоков и нескольких технических настроек.

Слово «разработчик» здесь оправданно. Вы определяете структуру данных, поведение кнопок, условия доступа и последовательность действий. Если пользователь нажал «Отправить», а обращение потерялось по дороге, проблема относится к разработке, даже когда весь процесс нарисован мышью. Заказчику нужен результат: заявка сохранена, ответственное лицо назначено, уведомление пришло, чужие данные недоступны.

Визуальные инструменты снижают порог входа, но не отменяют необходимость думать. Особенно важно различать отображение и хранение. Скрытая на экране строка может по-прежнему существовать в базе. Кнопка, которая выглядит неактивной, может иметь работающий обработчик. Сообщение «Успешно» не доказывает, что внешний сервис получил данные. По мере чтения вы научитесь проверять такие

вещи небольшими понятными опытами.

Первый полезный навык — перевести просьбу на язык действий. Вместо «сделайте современный сайт» нужно выяснить, что посетитель должен узнать, какую услугу выбрать, какие сведения оставить и что произойдёт дальше. Вместо «нужна автоматизация» — определить источник события, получателя, допустимые ошибки и человека, который заметит сбой.

Какие задачи подходят для первого заработка

Для начала подходят небольшие проекты с понятными границами: сайт услуги, каталог без сложной оплаты, форма записи, кабинет для нескольких сотрудников, передача заявки в CRM, напоминание о просроченной задаче. В таком проекте можно проверить основной путь целиком и объяснить заказчику, за что он платит.

Сложнее брать маркетплейс с несколькими типами продавцов, банковский продукт, медицинскую систему, приложение с высокой нагрузкой или сервис, где одновременно бронируются последние места. Эти задачи требуют больше внимания к безопасности, конкурентным изменениям, правовым требованиям и стоимости инфраструктуры. Отказ от них на старте помогает сохранить деньги и репутацию.

Продукт можно разложить на небольшие заказы. Мастерской сначала нужен сайт с тремя услугами и работающей формой. После запуска становится ясно, что обращения теряются в переписке: появляется задача кабинета. Когда кабинет используется, можно добавить уведомления и отчёт. Такая последовательность даёт возможность проверить пользу каждого этапа и не оплачивать заранее функции, которыми никто не воспользуется.

Чтобы выбрать первый тип услуги, оцените собственные

сильные стороны. Если вам легко работать с текстами и композицией, начните с сайтов. Если нравится разбирать порядок действий и искать ошибки, попробуйте автоматизации. Если интересны формы, таблицы и пользовательские роли, изучайте приложения. Основы остальных направлений всё равно пригодятся, но специализация упростит первые предложения клиентам.

Как устроены инструменты книги

Tilda удобна для публичных страниц: посетитель видит предложение, фотографии, ответы на вопросы и форму. Встроенные блоки помогают быстро собрать аккуратную страницу. Zero Block нужен, когда требуется собственная композиция. Он не является обязательным условием качественного сайта: стандартные блоки часто удобнее поддерживать.

Bubble подходит для приложений, в которых есть данные и действия пользователей. Вы задаёте типы записей, создаёте интерфейс и связываете события с изменениями базы. Например, нажатие на кнопку переводит обращение из статуса New в InProgress. Мобильный веб-интерфейс открывается в браузере телефона. Нативная мобильная сборка — отдельный способ доставки приложения, который требует своих проверок.

Make связывает сервисы. Его сценарий начинается с события или проверки по расписанию и выполняет последовательность модулей. В старых материалах встречается название Integromat: это прежнее название продукта. Для новых упражнений используйте актуальный интерфейс Make, а старые инструкции рассматривайте как подсказку о логике, а не как точную карту кнопок.

Инструменты не нужно соединять любой ценой. Если для

задачи достаточно формы Tilda и штатной интеграции с CRM, дополнительный сценарий Make только увеличит число зависимостей. В учебном проекте мы соединяем сервисы, чтобы освоить передачу событий. В коммерческом проекте выбирайте наименьшее количество компонентов, которое обеспечивает согласованный результат.

Доход без обещаний

Зарплата в вакансии, стоимость проекта и деньги, которые останутся у исполнителя, — разные величины. Из выручки оплачиваются подписки, подрядчики, налоги и время на поиск клиентов. Поэтому нельзя оценить профессию по одной красивой сумме из объявления. В книге используются учебные расчёты, а не обещания определённого заработка.

Например, вы оценили небольшой сайт в 32 000 рублей. На согласование, сборку, проверку и передачу ушло 28 часов. Ещё восемь часов заняли переговоры и подготовка предложения. При расходах 4 000 рублей остаётся 28 000 рублей до налогов. Деление на 36 часов даёт примерно 778 рублей за час. Это условный расчёт; он полезен тем, что учитывает работу, которую обычно забывают включить в цену.

Для исследования рынка соберите собственную небольшую выборку. Найдите не менее десяти свежих вакансий по словам Tilda, Bubble, no-code, low-code, интеграции и автоматизация. Выпишите обязанности, обязательные инструменты, опыт, формат занятости и указанное вознаграждение. Отдельно отметьте суммы до вычета налогов и на руки. Не объединяйте оплату по проекту с месячной зарплатой.

После этого посмотрите десять предложений услуг. У каждого отметьте объём: сколько страниц, есть ли тексты, включены ли мобильная версия, подключение домена, инте-

грации, обучение и поддержка. Цена без состава работы почти ничего не говорит. Предложение за небольшую сумму может относиться к переносу готового текста в шаблон, а более дорогое — к проектированию и запуску всей системы.

Как учиться с этой книгой

Читайте один раздел, выполняйте действие и сохраняйте результат. Если вы лишь просмотрели инструкции, знакомство с интерфейсом будет казаться пониманием. Настоящая проверка начинается, когда вы можете повторить действие без текста и объяснить, почему выбрали именно такие настройки.

Создайте папку проекта. В ней должны быть техническое задание, тексты сайта, модель данных, тестовые записи, перечень проверок, журнал ошибок и описание кейса. Для каждого значимого изменения записывайте дату, причину и результат. Такая привычка облегчает возвращение к работе после перерыва и помогает доказать, что проблема исправлена.

У каждой практической части будет наблюдаемый результат. После сайта — опубликованная учебная страница с проверенной формой. После приложения — две учётные записи, которые видят только разрешённые обращения. После автоматизации — тестовое событие, которое обрабатывается один раз, и понятное поведение при ошибке. Эти результаты важнее количества просмотренных уроков.

Названия кнопок и расположение настроек меняются. Когда интерфейс отличается от описанного, сначала найдите нужное действие в актуальной справке сервиса. Не пытайтесь

тесь угадать опасную настройку по похожему названию. В книге приведены технические имена там, где они помогают поиску, и объяснено, какой результат должен получиться.

Практика выбора первой услуги

Запишите три задачи, которые вы готовы освоить: например, сайт локального сервиса, кабинет менеджера и уведомления о новых обращениях. Для каждой укажите конкретного пользователя, проблему и способ проверки. Формулировка «помочь бизнесу» слишком широка. Подойдёт «сотрудник видит все необработанные заявки в одном списке и открывает нужную за два действия».

Затем выберите одну задачу для первого предложения. Опишите её пятью предложениями: кому нужна, что вы соберёте, какие материалы потребуются, что не входит в работу и как будет происходить приёмка. Пока не назначайте цену. Сначала нужно понять объём, иначе сумма станет случайной.

Для нашей мастерской начальное предложение звучит так: «Соберу сайт с тремя услугами и учебной формой обращения. Размещу описание порядка ремонта, ответы на вопросы и кнопку перехода к форме. Проверю страницу на компьютере и телефоне. Для запуска потребуются утверждённые тексты, фотографии и сведения об организации. Готовность подтвердим проверкой всех ссылок и доставкой тестовой заявки».

Сохраните собственную версию. В конце книги вы вернётесь к ней, уточните границы и добавите стоимость. Тогда

предложение будет опираться на выполненную работу, а не на впечатление от редактора.

Глава 2. Как превратить просьбу заказчика в задачу

Работа начинается с разговора о процессе. Если мастерская говорит «нам нужен сайт и приложение», ещё неизвестно, какая задача стоит перед компанией. Возможно, достаточно исправить форму на существующей странице. Возможно, сайт уже привлекает посетителей, а проблема заключается в том, что обращения никто не назначает сотрудникам. Разработчик сначала выясняет, где возникают потери, затем предлагает решение.

У учебной мастерской нет сложного расписания выездов. Она принимает обращение, уточняет детали и согласует время вручную. Это намеренное ограничение. Если сразу добавить автоматическое бронирование, оплату, склад деталей и расчёт маршрутов, первая версия станет слишком большой. Мы научимся доводить небольшой процесс до надёжного состояния и только потом обсуждать расширение.

Разговор о текущем процессе

Попросите заказчика описать последнее типичное обращение по шагам. Где человек нашёл мастерскую? Что написал? Кто получил сообщение? Где сохранили контакт? Как назначили ответственного? Что произошло, когда сотрудник заболел? Такие вопросы помогают увидеть действительные действия вместо общего описания «работаем через мессенджеры».

Затем разберите необычное обращение. Клиент отправил форму дважды, указал неверный контакт или просит услугу, которую мастерская не оказывает. Что сейчас делает сотрудник? В ответах часто скрываются правила будущей системы. Если проблему решают вручную, это не обязательно плохо. Иногда ручное решение редкого случая дешевле, чем отдельная автоматизация.

Не спрашивайте клиента, какую базу данных он хочет. Обычно ему нужен понятный результат. Спросите, кто должен видеть обращение, когда оно считается обработанным и сколько времени можно ждать ответа. Технический инструмент выбирается после этого. Если клиент уже использует CRM, уточните, почему нельзя продолжить работу в ней.

Результат разговора запишите коротким описанием. Например: «Заявки поступают через сайт. Администратор проверяет их и назначает исполнителя. Исполнитель видит толь-

ко назначенные ему обращения. После контакта он меняет статус. Руководитель смотрит общий список и просроченные задачи». Здесь уже есть участники, действия и ограничения доступа.

Первая версия системы

Первую версию часто называют MVP — минимальным продуктом, с помощью которого можно проверить полезность решения. Минимальность не означает сломанную форму или отсутствие защиты. Она означает небольшое количество функций при работающем основном пути.

В нашей версии посетитель выбирает одну из трёх услуг, указывает тестовый способ связи и отправляет обращение. Администратор видит список, открывает запись и назначает сотрудника. Сотрудник меняет статус New на InProgress, затем на Done или Cancelled. Make отправляет уведомление о новой записи и регистрирует результат обработки события.

В первую версию не входят платежи, автоматическая запись в календарь, хранение фотографий клиента, публичные отзывы и чат. Это не запрет на будущее развитие. Отложенные функции получают отдельную оценку после проверки основного процесса. Список исключений защищает обе стороны от разных представлений о готовности.

У каждой функции должен быть пользователь и причина. «Красивый личный кабинет» не помогает оценить работу. «Сотрудник открывает назначенную заявку и видит услугу, описание и текущий статус» уже можно проверить. Если функция не поддерживает основной путь и не устраняет значимый риск, её разумно отложить.

Требования и критерии приёмки

Требование описывает, что система должна делать. Критерий приёмки объясняет, как вы докажете, что она это делает. Удобно записывать их рядом. Для формы требование — обязательный выбор услуги. Критерий — при отправке без выбора запись не создаётся, а пользователь видит понятное сообщение возле поля.

Для списка сотрудника требование — показать только назначенные обращения. Критерий — под первой учётной записью видна заявка А, под второй — заявка В; попытка открыть чужую запись по прямой ссылке не раскрывает её содержимое. Проверка только внешнего вида списка недостаточна: нужно проверить доступ к самой записи.

Для автоматизации требование — обработать повторную доставку без повторного действия. Критерий — два события с одинаковым `event_id` создают одну итоговую запись и одно уведомление либо второе событие явно помечается как повтор. Как именно будет обеспечена защита, зависит от выбранного хранилища и режима обработки.

Пишите критерии так, чтобы их мог повторить человек, не участвовавший в разработке. Укажите исходные данные, действие и ожидаемый результат. «Форма работает корректно» ничего не объясняет. «Отправить `service_code=washer` с `event_id=E-101`; в журнале должна появиться одна запись со

статусом received» — проверяемое утверждение.

Пользовательские роли

Роль — набор разрешённых действий. Она не обязана совпадать с должностью. В маленькой мастерской владелец может быть одновременно администратором и исполнителем, но для системы важно понимать, какие права он получает в каждом случае.

В учебной системе гость видит публичные услуги и форму. Исполнитель читает назначенные ему обращения и меняет разрешённые статусы. Администратор читает все обращения, назначает исполнителей и исправляет ошибочные сведения. Ни один пользователь через обычный профиль не может самостоятельно сделать себя администратором.

Разделите чтение, создание, изменение и удаление. Если человек может читать запись, из этого не следует право менять её. Если администратор может исправить описание, из этого не следует право незаметно удалить историю. Удаление особенно опасно: оно может уничтожить доказательства ошибки. В первой версии лучше использовать признак `archived` и скрывать запись из обычного списка.

Запишите матрицу прав до сборки интерфейса. Для каждого типа данных укажите, кто видит какие поля и какие действия доступны. Контакты и технический журнал не должны автоматически становиться доступными всем сотрудникам лишь потому, что им открыт список заявок.

Состояния обращения

Статус должен описывать состояние процесса, а не настроение сотрудника. Мы используем четыре значения: New — получено; InProgress — принято в работу; Done — завершено; Cancelled — отменено. Названия можно показать пользователю по-русски, сохранив постоянные технические коды.

Разрешённые переходы задаются отдельно. New можно перевести в InProgress или Cancelled. InProgress — в Done или Cancelled. Возврат из Done в работу требует решения администратора и записи причины. Если разрешить любой переход любой кнопкой, отчёт о выполненных обращениях быстро перестанет быть надёжным.

Статус обработки события в Make относится к другой сущности. Received означает, что событие принято; processed — что его действие завершено; failed — что нужна повторная обработка или разбор. Не смешивайте эти значения со статусом ремонта. Обращение может находиться в New, хотя уведомление о нём уже processed.

У каждого статуса должен быть ответственный за дальнейшее действие. Для New это администратор, для InProgress — назначенный исполнитель. Если у записи нет исполнителя, не позволяйте переводить её в InProgress без отдельного правила. Такой небольшой запрет предотвращает появление

задач, которые формально в работе, но фактически никому не принадлежат.

Что нужно получить от заказчика

До начала сборки запросите тексты услуг, фотографии с правом использования, контактные сведения организации, доступ к домену, сведения о существующей CRM и ответственного за согласование. Уточните, кто владеет аккаунтами сервисов и кто оплачивает подписки. Сайт на личном аккаунте разработчика становится проблемой, если сотрудничество прекращается.

Не просите переслать пароль в обычную переписку. Используйте приглашение участника, отдельный доступ или другой предусмотренный сервисом способ. Если возможности разграничения ограничены тарифом, обсудите это заранее. Передача доступа — часть проектирования, а не действие, которое можно оставить на последний день.

Отдельно составьте перечень данных. Для учебной формы достаточно кода услуги и вымышленного контакта. Для реального сервиса заказчик определяет цель обработки, необходимый состав сведений и правовое основание. Не добавляйте дату рождения, адрес и паспортные данные «на всякий случай». Каждое поле усложняет защиту, поддержку и объяснение пользователю.

Задание на одну страницу

Составьте собственный документ из восьми частей: цель, пользователи, основной путь, функции первой версии, исключения, данные, приёмка и зависимости. Его можно написать обычными абзацами. Важно, чтобы он снимал неоднозначность, а не выглядел как большой официальный документ.

Для нашей мастерской цель — не терять учебные обращения между сайтом и сотрудником. Пользователи — посетитель, администратор и исполнитель. Основной путь — отправка, сохранение, назначение, обработка. Приёмка — успешный проход и проверки повторов, пустых полей, чужого доступа и недоступного получателя.

Последняя часть — зависимости — помогает правильно оценить срок. Если нет утверждённых текстов, создание страниц нельзя считать полностью готовым. Если сервис недоступен или не оплачивается разрешённым способом, не обещайте интеграцию к определённой дате. Запишите, кто и к какому сроку должен закрыть зависимость.

Работа с новой просьбой

После первого показа заказчик может попросить «добавить ещё небольшую функцию». Не оценивайте её по размеру кнопки. Новый календарь затрагивает данные, часовые пояса, конфликты записи, уведомления и отмену. Даже маленький элемент интерфейса способен изменить всю логику.

Опишите просьбу, её причину и влияние. Затем предложите выбор: заменить часть согласованного объёма, перенести функцию в следующую версию или увеличить бюджет и срок. Это нормальное управление проектом. Конфликт возникает, когда изменение молча принимают, а позже называют дополнительные деньги.

В учебном задании добавьте просьбу «клиент должен сам выбирать время». Запишите вопросы, которые нужно задать до оценки: сколько исполнителей, какие интервалы, возможны ли переносы, кто подтверждает, сколько мест в одном времени, что происходит при одновременной записи. Вы увидите, почему проектирование экономит больше времени, чем быстрый выбор красивого шаблона.

После этой главы у вас должен быть документ, по которому можно начать сборку. Сохраните его версию и дату. Дальше мы разберём, как браузер, база данных и внешние сервисы выполняют описанный путь.

Глава 3. Веб, данные и события без сложного кода

Когда человек открывает сайт, браузер запрашивает материалы у сервера и показывает страницу. Когда он отправляет форму, сведения должны попасть к получателю и получить дальнейшую обработку. Между двумя действиями существует несколько технических шагов. Понимание этих шагов помогает найти ошибку без многолетнего опыта программирования.

Представьте, что форма сообщила об успехе, но сотрудник не получил заявку. Нужно проверить, что именно произошло: браузер отправил запрос, Tilda приняла данные, интеграция передала их, Make запустил сценарий, база сохранила запись, а модуль уведомления завершился. Если искать проблему сразу во всех местах, работа затянется. Если проверять путь по участкам, причина обычно становится понятнее.

Клиент и сервер

Клиентская часть работает на устройстве пользователя. В веб-приложении это браузер. Здесь отображаются тексты, открываются окна и вводятся значения. Серверная часть хранит данные и выполняет действия, которым пользователь не должен произвольно управлять. Например, она проверяет право читать запись и хранит секрет подключения к внешнему сервису.

Данные, уже отправленные в браузер, нельзя надёжно защитить скрытием элемента. Пользователь может изучить полученный ответ техническими средствами. Поэтому ограничение доступа должно действовать до передачи сведений. В Bubble для чтения используются `privacy rules`, а изменение через `workflows` требует собственных проверок условий и полномочий.

Не все действия обязаны выполняться на сервере. Открыть меню или изменить выбранный фильтр можно на устройстве. Но расчёт прав, подтверждение оплаты и привилегированное изменение статуса нельзя доверять только внешнему виду экрана. Простой вопрос помогает выбрать место выполнения: что произойдёт, если пользователь изменит это значение самостоятельно?

**Адрес, домен и
HTTPS**

URL — адрес конкретного ресурса. Домен — понятное человеку имя сайта. Путь после домена указывает на страницу или другой ресурс. Параметры после вопросительного знака передают дополнительные значения. Например, в учебном адресе <https://example.com/request?id=R-101> значение `id` обозначает нужную запись. Само наличие такого значения не предоставляет право читать её.

DNS связывает доменное имя с адресами и другими настройками. Для подключения сайта вы вносите записи, которые указаны в актуальном кабинете сервиса. Не копируйте адрес сервера из старого урока: инфраструктура может измениться. Перед изменением сохраните существующие записи, особенно относящиеся к почте.

HTTPS защищает передачу между браузером и сайтом с помощью шифрования. Он не делает форму автоматически законной и не исправляет ошибочные права доступа. Проверяйте, что защищённый адрес открывается без предупреждений и что внешние материалы страницы не вызывают проблем. Если сертификат ещё выпускается, не пытайтесь обойти предупреждение и объявить сайт готовым.

Типы значений

Текст подходит для описания и кода услуги. Число — для количества и арифметики. Дата — для момента создания или запланированного действия. Логическое значение принимает два состояния, например yes/no. Ссылка на запись связывает сущности: у обращения есть назначенный пользователь.

Телефон, почтовый индекс и идентификатор лучше не хранить как обычные числа. У них могут быть ведущие нули, знаки и буквы; арифметика с ними не нужна. Если идентификатор E-101 превратить в число, смысл будет утрачен. Тип поля выбирайте по будущим действиям, а не по внешнему сходству значения.

Пустая строка, отсутствующее поле и нулевое значение различаются. Стоимость 0 может означать бесплатную услугу. Отсутствующая стоимость означает, что цену ещё не определили. Если автоматизация смешивает эти состояния, она отправит неверное предложение. Для каждого важного поля заранее задайте, что делать при отсутствии значения.

Записи и связи

База данных хранит отдельные записи. У обращения есть уникальный идентификатор, услуга, описание, статус, дата создания и назначенный сотрудник. У услуги — код, название и признак активности. У пользователя — учётная запись и роль. Не нужно повторять полное описание услуги в каждом обращении, если достаточно ссылки на соответствующую запись.

Однако иногда полезно хранить снимок значения. Если цена услуги меняется, старое согласованное предложение не должно неожиданно получить новую сумму. В такой ситуации в обращении сохраняют `agreed_price` — цену, принятую на момент договорённости. Ссылка на текущую услугу и снимок согласованной цены решают разные задачи.

Уникальный идентификатор не должен зависеть от имени клиента или номера строки. Имя меняется, строки сортируются. Для учебных данных мы используем понятные коды R-101 и E-101. В рабочей системе обычно применяется идентификатор, созданный платформой или согласованным механизмом. Красивый номер для человека можно хранить отдельно.

JSON

и сопоставление полей

JSON — распространённый текстовый формат переда-

чи данных. Он описывает объекты, списки и значения. Не нужно учиться писать большую программу, чтобы понимать небольшой объект. Каждая пара состоит из имени поля и значения. Кавычки вокруг текста важны; число записывается без кавычек, если получатель ожидает именно числовой тип.

В учебном событии поля `event_id`, `service_code` и `description` обозначают идентификатор события, код услуги и описание. Получатель должен заранее знать, какие поля обязательны. Если один сервис отправляет `service`, а другой ищет `service_code`, значение не появится автоматически. Это и есть задача сопоставления полей — `mapping`.

Сначала нарисуйте соответствие на бумаге: поле формы `Service` → `service_code` события → `Service` записи приложения. Затем настройте его в редакторе. Так проще заметить, что русское отображаемое название и технический код используются по-разному. Для пользователя показывается «Ремонт стиральной машины», между сервисами передаётся `washer`.

Список в JSON может содержать несколько объектов. Если внешний сервис вернул три записи, следующий модуль иногда запускается для каждой. Именно поэтому после поиска внезапно появляется три письма вместо одного. Проверьте количество полученных элементов и то, как редактор преобразует список в отдельные выполнения.

API

и вебхук

API — согласованный способ обратиться к сервису программно. Вы отправляете запрос на определённый адрес с нужными параметрами и получаете ответ. Например, спрашиваете список услуг или создаёте новую запись. Документация API объясняет адреса, типы запросов, авторизацию и формат результата.

Вебхук сообщает о событии, когда оно произошло. Источник отправляет данные на заранее заданный адрес получателя. В нашей учебной связке форма может вызвать вебхук Make. Другой подход — проверка по расписанию: автоматизация регулярно спрашивает, появились ли новые записи. Вебхук обычно сокращает ожидание, но требует продуманной обработки повторов и ошибок.

Не путайте адрес вебхука с публичной страницей. Длинный случайный адрес затрудняет угадывание, но сам по себе не всегда достаточен для защиты. Там, где источник поддерживает подпись или другой способ подтверждения, используйте его. Если такой возможности нет, поставьте контролируемый приёмник, ограничьте полномочия и не передавайте чувствительные данные без проверки архитектуры.

Ответы и ошибки

HTTP-ответ содержит код состояния. Коды семейства 2xx обычно означают успешное принятие запроса на соответствующем уровне. 400 часто указывает на неверный запрос, 401 — на отсутствие подходящей авторизации, 403 — на запрет, 404 — на ненайденный ресурс, 429 — на превышение ограничений, 5xx — на проблему на стороне сервера. Точное значение уточняется в документации API.

Полученный 200 не всегда доказывает завершение всего бизнес-процесса. Приёмник мог только поставить событие в очередь. Поэтому для важной операции нужны отдельный статус обработки и возможность проверить итог. Аналогично тайм-аут не доказывает, что действие не произошло: получатель мог сохранить запись, а ответ потерялся по дороге.

Повторять запрос нужно с пониманием последствий. Повторное чтение обычно менее опасно, чем повторное создание заказа. Для изменения данных полезен ключ идиempotentности — значение, по которому получатель понимает, что это та же операция. В нашей модели таким ключом служит `event_id`, но его наличие в объекте ещё не обеспечивает защиту. Получатель должен действительно проверять и учитывать его.

Время и порядок

Храните момент события в согласованном формате и явно указывайте часовой пояс. Для технического обмена удобно использовать UTC — всемирное координированное время. Для сотрудника показывайте местное время. Россия использует несколько часовых поясов, поэтому нельзя автоматически считать время всех клиентов московским.

Дата без времени подходит для дня рождения или отчётного дня. Момент события требует времени и зоны. Строка «14.10.2026 10:00» не сообщает, в каком регионе произошло действие. Запись 2026-10-14T07:00:00Z однозначно обозначает UTC; отображение в конкретной зоне выполняется отдельно.

События могут прийти не по порядку. Уведомление об изменении статуса может обогнать уведомление о создании. Поэтому надёжная интеграция использует идентификаторы, время, версию записи или другой способ проверки актуальности. В первой учебной версии мы ограничим процесс последовательной обработкой и явно отметим, чего эта мера не гарантирует в более сложной системе.

Практика чтения данных

Создайте пять вымышленных обращений. Одно полное, одно без описания, одно с неизвестной услугой, одно с пустым `event_id` и одно с повторяющимся `event_id`. Для каждого заранее запишите ожидаемую реакцию. Полное принимается, неизвестная услуга отклоняется, отсутствие необязательного описания допустимо, пустой ключ запрещён, повтор не вызывает повторного действия.

Теперь объясните путь одного обращения человеку, который не знает редакторов. Он должен услышать понятную последовательность: посетитель отправляет форму, получатель проверяет поля, система сохраняет запись, уведомляет администратора, администратор назначает сотрудника. Если объяснение сразу превращается в набор названий модулей, вернитесь к действиям пользователя.

Сохраните словарь полей. Для каждого укажите техническое имя, тип, обязательность, пример и назначение. Этот документ будет использоваться в Tilda, Bubble и Make. Когда все три части называют одно значение одинаково, интеграции становятся заметно проще.

Глава 4. Сайт на Tilda

от пустой страницы до формы

В этой главе вы соберёте сайт мастерской. Его задача про-

ста: объяснить, какие услуги доступны, показать порядок обращения и довести посетителя до формы. Мы не будем начинать с анимации и необычной композиции. Сначала сделаем понятную страницу, затем проверим её на разных экранах и подключим учебный приём данных.

Перед работой создайте отдельный учебный проект. Не используйте действующий сайт организации для экспериментов. В заголовке формы и рядом с кнопкой отправки укажите, что это демонстрация и настоящие контактные данные вводить не нужно. Для публикации используйте доступный учебный адрес; собственный домен понадобится позже.

Структура страницы

На странице будет семь смысловых частей: первый экран, услуги, порядок работы, основания доверять мастерской, ответы на вопросы, форма и контактный блок. Это не универсальный рецепт для любого бизнеса. Такая структура подходит нашему короткому пути: человек должен понять предложение и отправить обращение.

Первый экран отвечает на три вопроса: что предлагается, где оказывается услуга и что сделать дальше. Учебный заголовок — «Ремонт домашней техники с предварительным согласованием работ». Под ним можно написать: «Выберите тип техники и опишите неисправность. Сотрудник уточнит детали и предложит следующий шаг». Кнопка — «Оставить учебную заявку».

Не обещайте конкретный срок или бесплатную диагностику, если это не подтверждено заказчиком. В учебном тексте можно обозначить условие как часть вымышленного задания, но в коммерческом сайте оно должно соответствовать действительной услуге. Красивый заголовок не оправдывает ложное обещание.

Блок услуг содержит три карточки: стиральные машины, холодильники и посудомоечные машины. У каждой есть название, краткое описание и ссылка к форме. Цена пока не указывается: в нашей модели она определяется после уточ-

нения. Если добавить «от 500 рублей» без объяснения состава, посетитель может ожидать полный ремонт за эту сумму.

Порядок работы описывает действия: отправить обращение, обсудить неисправность, согласовать условия, получить результат. Не используйте формулировки «индивидуальный подход» и «высочайшее качество» вместо конкретики. Читатель должен понимать, что произойдёт после нажатия кнопки.

Сборка стандартными блоками

Создайте новую страницу и выберите пустой вариант или подходящий шаблон. Если используете шаблон, сразу удалите лишние разделы и замените весь демонстрационный текст. Случайно оставленный чужой номер телефона или вымышленный отзыв выглядит как мелочь, но делает сайт ненадёжным.

Добавляйте блоки по смыслу. Для первого экрана подойдёт обложка с заголовком, текстом и кнопкой. Для услуг — блок карточек. Для процесса — последовательность шагов. Для вопросов — раскрывающийся список, если выбранный блок поддерживает такое поведение. Для формы — стандартный блок ввода. Пока оставьте типографику и отступы простыми.

В каждом блоке различайте содержание и настройки. Содержание — тексты, изображения, ссылки и поля. Настройки — размер, цвет, отступы, фон и видимость. Если менять оба слоя одновременно, трудно понять, что испортило результат. Сначала заполните содержание всей страницы, затем займитесь внешним видом.

После каждого крупного изменения открывайте предпросмотр. Он помогает увидеть последовательность страницы, но не заменяет проверку опубликованной версии. Некоторые интеграции и внешние действия становятся доступ-

ны только после публикации. Не оценивайте работу формы
лишь по редактору.

Текст и смысл кнопок

Кнопка должна объяснять действие. «Подробнее» подходит для перехода к описанию; «Отправить заявку» — для отправки формы. «Начать» и «Далее» могут быть понятны внутри конкретного процесса, но на первом экране часто не сообщают, что произойдёт. Посетитель должен понимать действие до нажатия.

Проверьте, что все кнопки с одинаковым назначением ведут к одному ожидаемому месту. Если одна карточка открывает окно, вторая прокручивает страницу, а третья ведёт на неизвестный адрес, поведение становится непредсказуемым. Различные действия допустимы, когда они ясно обозначены текстом.

Для перехода к форме используйте якорную ссылку. Якорь — отметка внутри страницы, к которой браузер может прокрутить содержимое. Задайте устойчивое имя, например `request-form`, и подключите его к кнопкам. После публикации проверьте переход с начала и середины страницы. Если закреплённое меню закрывает заголовок формы, скорректируйте отступ или поведение меню.

В карточках услуг не обязательно заранее выбирать значение формы. На первом этапе достаточно привести человека к нужному полю. Автоматический выбор добавляйте только после того, как основной переход работает и вы понимаете

те, как выбранный блок передаёт значение. Сложная микро-функция не должна задерживать запуск простой страницы.

Изображения и права использования

Для учебного сайта можно использовать собственные фотографии, разрешённые стоковые материалы или специально созданные иллюстрации. Сохраняйте источник и условия использования. То, что картинка находится в поиске, не означает, что её можно разместить в коммерческом проекте.

Выбирайте изображение, которое поддерживает смысл. Фотография мастера рядом с техникой помогает понять услугу. Абстрактные сияющие фигуры могут выглядеть эффектно, но не объясняют предложение. Если используется сгенерированное изображение, проверьте детали: руки, инструменты, надписи, логотипы и технически невозможные действия.

Подготовьте размеры файлов до загрузки. Большая фотография с телефона может занимать несколько мегабайт, хотя на странице видна небольшой карточкой. Сократите размеры до нужных и используйте подходящий формат. Проверьте результат визуально: чрезмерное сжатие создаёт артефакты и ухудшает впечатление.

Альтернативный текст описывает содержание значимого изображения для пользователей вспомогательных технологий и случаев, когда картинка не загрузилась. Для декоративной картинки подробное описание может быть лишним. Не превращайте alt в набор поисковых ключей. Он нужен,

чтобы сообщить смысл, который несёт изображение.

Типографика и композиция

Начните с одного семейства шрифта и небольшой системы размеров. Заголовок первого экрана заметнее заголовков разделов, обычный текст меньше, но легко читается на телефоне. Не меняйте размер каждого абзаца отдельно: это усложняет поддержку и создаёт визуальную случайность.

Длина строки влияет на чтение. Очень широкий абзац трудно просматривать, слишком узкий создаёт много коротких строк. В стандартных блоках используйте предусмотренную ширину текста. Если фраза выглядит тяжёлой, сначала сократите её или разделите по смыслу, а затем меняйте размер шрифта.

Цвет основной кнопки должен отличаться от окружающих элементов. Для учебной страницы достаточно одного акцентного цвета. Проверьте контраст текста и фона. Светло-серые подписи могут выглядеть аккуратно на вашем мониторе и плохо читаться на другом устройстве или при ярком освещении.

Отступы должны показывать отношения. Заголовок расположен ближе к своему тексту, чем к предыдущему разделу. Карточки одинакового уровня имеют похожие промежутки. Если страница выглядит разрозненной, причиной часто являются случайные отступы, а не отсутствие дорогой фотографии.

Форма обращения

Добавьте четыре поля: выбор услуги, тестовый контакт, описание и согласие с условиями учебной демонстрации. В реальном проекте состав юридических документов и основание обработки определяются отдельно. Учебная подпись не является универсальным текстом согласия для коммерческого сайта.

Назначьте технические имена `service_code`, `contact_demo` и `description` там, где выбранный блок позволяет задать имена переменных. Отображаемая подпись может быть русской. Проверьте, какие значения отправляет выпадающий список: видимый текст, технический код или оба значения. Не предполагайте, что слово `washer` появится автоматически, если в поле видна «Стиральная машина».

Если блок не позволяет удобно отделить подпись от значения, передавайте видимое название и преобразуйте его в код на стороне получателя через явную таблицу соответствия. Это лучше, чем скрытое предположение. В таблице должно быть ровно одно соответствие для каждого допустимого варианта.

Обязательными сделайте услугу и тестовый контакт. Описание оставьте необязательным, но ограничьте разумную длину, если это поддерживается выбранным способом обработки. Пример подсказки: «Для демонстрации используйте

te test@example.com». Домен example.com предназначен для примеров; не отправляйте туда реальные уведомления и не рассчитывайте на получение письма.

Подпись поля остаётся видимой после ввода. Не полагайтесь только на placeholder — текст-подсказку внутри поля, который исчезает, когда человек начинает писать. Посетитель должен помнить, что требуется указать, даже после ошибки отправки.

Сообщения об ошибках и успехе

Пустое обязательное поле должно вызывать сообщение рядом с ним или другой явно связанный сигнал. Сообщение «Ошибка» слишком короткое. Подойдёт «Выберите услугу» или «Укажите тестовый контакт». Если внешний сервис не отвечает, не обвиняйте пользователя: «Не удалось отправить. Попробуйте позже» понятнее, чем технический код на весь экран.

Сообщение об успехе зависит от того, что подтверждено. Если форма только принята платформой, текст «Мастер уже назначен» неверен. Для учебной версии используйте «Учебная заявка отправлена. Проверьте её появление в журнале». Для реального сайта обещайте только согласованный следующий шаг.

Проверьте поведение повторного нажатия. Даже если кнопку временно блокирует интерфейс, пользователь может отправить ту же форму после обновления страницы. Поэтому защита от повторов не заканчивается в браузере. Её нужно учитывать на стороне обработки событий и записей.

Подключение приёмника

У Tilda есть встроенные сервисы приёма данных. Общая последовательность такова: подключить сервис в настройках сайта, выбрать его в конкретной форме, сохранить и опубликовать страницу. Сам факт появления подключения в настройках сайта ещё не означает, что новая форма отправляет данные именно туда.

Для первого опыта можно использовать доступный штатный приёмник с вымышленными данными. Если условия выбранного тарифа не позволяют подключить нужную интеграцию, не покупайте её вслепую. Сначала проверьте, какие функции включены, можно ли пройти учебный путь и что потребуется для публикации. Информация о тарифах изменяется, поэтому книга не фиксирует их стоимость как постоянную.

Когда дойдёте до Make, подключите вебхук к отдельной учебной форме. Не меняйте приёмник действующего сайта ради эксперимента. Отправьте одну запись и сравните фактические имена полей с вашим словарём. До этой проверки нельзя считать сопоставление завершённым.

Если используются несколько получателей, выясните, как проявляется частичный сбой. Заявка может появиться в одном месте и отсутствовать в другом. Ваш журнал должен помогать проверить каждый участок, а не просто повторять со-

общение пользователя «ничего не пришло».

Первая публикация

Сохраните страницу и опубликуйте её на учебном адресе. Откройте адрес в другом окне, где вы не работаете в редакторе. Проверьте заголовок, все кнопки, порядок блоков и форму. Затем обновите текст в одном блоке, повторно опубликуйте и убедитесь, что изменение действительно появилось.

Публикация — отдельное действие. Сохранённые в редакторе изменения не всегда немедленно становятся видны посетителям. Это особенно важно при исправлении юридических сведений, цены и контактов. После обновления проверяйте публичную страницу, а не только сообщение редактора.

Проведите пять тестов: полная запись, отсутствие услуги, отсутствие контакта, длинное описание и повторная отправка. Результат каждого запишите. Пока можно вручную сверять данные в приёмнике; автоматический журнал появится позже. Если форма потеряла строку или передала неожиданный код, исправьте проблему до работы над декоративными эффектами.

Проверка содержания

Прочитайте страницу как посетитель. Можно ли понять предложение без прокрутки? Видно ли, что это учебный проект? Есть ли конкретный следующий шаг? Не обещаны ли реальные условия вымышленной организации? Совпадают ли названия услуг на карточках и в форме?

Попросите другого человека пройти страницу без объяснения. Не подсказывайте, где находится нужная кнопка. Наблюдайте, на каком месте он останавливается. Если человек ищет цену, а страница молчит о способе её определения, добавьте объяснение. Если он не понимает, что форма учебная, сделайте это заметнее.

В конце этой главы у вас должна быть опубликованная страница, которая объясняет услугу и передаёт вымышленную заявку. Это первый законченный результат. Следующая задача — довести сайт до состояния, которое можно передать другому человеку и поддерживать без постоянной помощи автора.

Глава 5. Мобильная версия, домен и запуск сайта

Сайт нельзя считать завершённым, если он хорошо выглядит только на широком мониторе. Посетитель может открыть страницу на телефоне, увеличить текст, перейти по ссылке из мессенджера или вернуться после ошибки формы. В этой главе вы проверите такие ситуации и подготовите сайт к передаче.

Мы также разберём Zero Block, домен, HTTPS, базовые настройки поиска и аналитики. Эти задачи относятся к одной цели: посетитель должен найти страницу, понять её и выполнить действие. Подключение десятка инструментов без проверки основного пути не улучшает запуск.

Мобильная проверка по действиям

Начните с реального телефона, если он доступен. Откройте опубликованный адрес, прокрутите страницу и заполните форму. Затем повторите проверку через изменение ширины браузера. Эмуляция помогает быстро находить проблемы, но не показывает полностью поведение клавиатуры, касаний и системных элементов телефона.

Проверьте первый экран. Заголовок не должен перекрывать изображение так, что текст становится нечитаемым. Кнопка должна помещаться и оставаться доступной. Если фотография обрезается, важная часть изображения должна сохраняться. Не уменьшайте весь экран до микроскопического размера, чтобы повторить композицию компьютера.

Посмотрите на карточки услуг. На узком экране они могут стать вертикальным списком. Это нормальное изменение. Важнее, чтобы названия и действия читались, а пользователь не искал спрятанную кнопку. Горизонтальная прокрутка всей страницы обычно означает, что один элемент вышел за доступную ширину.

Заполните форму с открытой клавиатурой. Видно ли текущее поле? Можно ли выбрать услугу? Не перекрывает ли закреплённая кнопка нижний элемент? После ошибки сохраняются ли уже введённые значения? Эти вопросы выявляют проблемы, которые не заметны по неподвижному

скриншоту.

Когда нужен

Zero

Block

Zero Block позволяет создать собственную композицию из отдельных элементов. Он полезен, когда стандартный блок не поддерживает нужное расположение или визуальная идея имеет значение для проекта. Но свобода означает дополнительную ответственность за разные ширины экрана.

Выберите один учебный участок — например, небольшой блок «Как проходит обращение». Соберите в Zero Block заголовок, три шага и кнопку. Не переводите в него весь сайт сразу. Так вы сможете сравнить трудоёмкость и понять, оправдывает ли собственная композиция дополнительную поддержку.

У элементов есть размеры, положение и привязка к контейнеру. Контейнер сетки помогает держать содержимое в общей ширине, контейнер окна ориентируется на пространство браузера. Когда элемент неожиданно уезжает, сначала проверьте, к чему он привязан и как задана ширина. Случайное перетаскивание часто только скрывает ошибку на одной конкретной ширине.

Создавайте адаптации от крупных экранов к узким, но каждый вариант проверяйте самостоятельно. На телефоне три шага разумно поставить друг под другом. Высота блока должна соответствовать новому содержимому. Если уве-

личился заголовок или изменилась подпись, убедитесь, что нижняя кнопка не перекрывает текст.

Не используйте анимацию для сообщения, которое пользователь обязан прочитать. Если элемент появляется с большой задержкой, человек может решить, что нужной информации нет. Для учебного сайта достаточно небольшого ненавязчивого эффекта или полного отсутствия анимации. Важнее завершённая форма и понятная мобильная версия.

Размеры и доступность

Кнопки и ссылки должны быть достаточно удобны для касания. Не помещайте несколько маленьких действий вплотную. Если пользователь должен попасть пальцем в крошечный значок, увеличьте активную область или добавьте текстовую кнопку. Такие детали помогают пользоваться интерфейсом без лишних усилий.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.