Методология

- Методика разработки дизайна
- Стандарты Web UI
- Типовые ошибки в дизайне интерфейсов при переходе от “рисования” к реальной UI/UX-разработке
- Тренды в UI/UX: перенос архитектуры на этап дизайна
- Как понимать контекст проекта

Методика разработки дизайна

1. Введение

Процесс разработки дизайна для заказчика требует от специалиста не только владения инструментами и принципами визуальной коммуникации, но и умения выстраивать грамотное взаимодействие с клиентом.
Главная задача дизайнера заключается в создании эстетически привлекательного и функционального продукта, который решает конкретные задачи бизнеса и пользователей.

Для достижения этой цели дизайнер должен организовать рабочий процесс так, чтобы:

- все ключевые решения фиксировались и согласовывались с заказчиком на ранних этапах;
- коммуникация происходила напрямую, без посредников;
- визуальные и функциональные решения имели обоснование с точки зрения пользовательского опыта и профессиональных стандартов.

---

2. Этап визуальной концепции (мудборды)

2.1. Назначение и значение этапа

Мудборд (moodboard) — это инструмент визуализации стилистического направления проекта. Он представляет собой подборку изображений, шрифтов, цветовых сочетаний и композиционных решений, отражающих предполагаемое настроение и атмосферу будущего продукта.

Создание мудборда необходимо для:

- согласования визуального направления с заказчиком до начала детальной проработки макетов;
- минимизации количества правок на последующих этапах;
- формирования общего понимания эстетики проекта.

Рекомендуемые материалы:

- Что такое мудборд — Яндекс.Практикум
- Мудборд — глоссарий Contented

2.2. Практические рекомендации

- Подготовьте несколько вариантов мудбордов, отражающих разные стилистические решения.
- Обсудите и согласуйте выбранное направление напрямую с заказчиком.
- После утверждения мудборда зафиксируйте его как основу визуальной концепции проекта.

---

3. Типографика

Типографика определяет структуру и читаемость интерфейса. Правильный выбор шрифтов и их иерархия обеспечивают удобство восприятия информации пользователями.

Рекомендуемые материалы:

- 10 правил типографики в интерфейсах — Contented
- Типографика в веб-дизайне в цифрах — VC.ru

Рекомендации:

- Используйте не более двух-трёх шрифтовых пар в одном проекте.
- Формируйте чёткую иерархию заголовков, подзаголовков и основного текста.
- Проверяйте читаемость на различных разрешениях экранов.

---

4. Сетки и структурирование контента

Сетка является основой композиционной организации макета. Она обеспечивает визуальный порядок и согласованность элементов.

Рекомендуемые материалы:

- Сетки в веб-дизайне — VC.ru
- Сетки в веб-дизайне — Skedraw
- Пример сетки — Figma Community

Рекомендации:

- Используйте модульные и колоночные сетки.
- Соблюдайте пропорциональность и единообразие отступов.
- Структура должна быть адаптивной для различных устройств.

---

5. Цветовое решение

Цветовая палитра формирует эмоциональный отклик и визуальный баланс интерфейса. Цвет должен подбираться осознанно, с учётом целевой аудитории и контекста бренда.

Рекомендуемые материалы:

- Цветовая палитра для сайта — Яндекс.Практикум
- Цвет в веб-дизайне и его влияние — VC.ru

Рекомендации:

- Используйте ограниченную палитру: 1–2 основных цвета и 1 акцентный.
- Проверяйте контрастность и читаемость текста.
- При наличии брендбука следуйте корпоративным цветовым стандартам.

---

6. Композиция

Композиция определяет визуальный ритм, баланс и иерархию элементов интерфейса.

Рекомендуемые материалы:

- Основы композиции в дизайне — Contented
- Основные правила композиции — Яндекс.Практикум

Рекомендации:

- Используйте принципы визуального равновесия и акцентирования.
- Избегайте перегруженности экрана второстепенными элементами.
- Соблюдайте целостность восприятия интерфейса.

---

7. UI Kit (дизайн-система проекта)

UI Kit — это систематизированный набор компонентов интерфейса (кнопки, поля ввода, карточки, иконки, стили текста и пр.), который обеспечивает единообразие визуальных решений.

Рекомендуемые материалы:

- UI Kit — Яндекс.Практикум
- Глоссарий Contented: UI Kit
- UI Kits — Figma Community

Рекомендации:

- Создавайте компоненты со всеми необходимыми состояниями.
- Настраивайте связь UI Kit с автолейаутами.
- Используйте UI Kit как единый источник истины по стилю проекта.

---

8. Автолейауты

Автолейаут (Auto Layout) — инструмент Figma, позволяющий создавать гибкие и адаптивные интерфейсы.

Рекомендуемые материалы:

- Auto Layout в Figma — Contented
- Пример файла — Figma Community

Рекомендации:

- Используйте автоотступы и выравнивание для стабильности макета.
- Комбинируйте автолейауты с компонентами и вариантами.
- Применяйте для подготовки кликабельных прототипов.

---

9. Кликабельные прототипы

Кликабельный прототип имитирует поведение готового продукта и позволяет наглядно продемонстрировать структуру и логику интерфейса заказчику.

Рекомендуемый ресурс:

- Бесплатный курс по Figma — Tilda School

Рекомендации:

- Обязательно предоставляйте заказчику кликабельный прототип перед финальным утверждением.
- Прототип помогает выявить UX-проблемы на раннем этапе.
- Презентацию макета проводите лично, сопровождая её профессиональными комментариями.

---

10. Коммуникация с заказчиком

10.1. Прямое взаимодействие

Дизайнер должен **всегда взаимодействовать с заказчиком напрямую**, а не через посредников (например, менеджеров без дизайнерской компетенции).
Прямой контакт позволяет:

- корректно понимать требования и контекст задач;
- оперативно согласовывать решения;
- минимизировать искажения при передаче информации.

Отсутствие прямой коммуникации часто приводит к некорректным правкам и неверному пониманию целей проекта.

10.2. Работа с брендбуком

Если у заказчика имеется **брендбук**, дизайнер обязан строго следовать указанным в нём правилам:

- использовать утверждённые шрифты, цвета и логотипы;
- соблюдать композиционные и визуальные принципы бренда.

Если брендбук отсутствует, **недопустимо ориентироваться** на формулировки вроде «сделайте, как у нас на сайте».
Существующий сайт может содержать устаревшие или некорректные решения, поэтому дизайнер должен опираться на профессиональные стандарты, принципы эргономики и визуальной гармонии.

10.3. Согласование и правки

- Все ключевые визуальные и функциональные решения фиксируются и утверждаются на этапе дизайна.
- После утверждения дизайн считается согласованным. Внесение изменений осуществляется **только при дополнительной оплате** или по технической необходимости.
- Любые корректировки должны иметь обоснование, связанное с задачами продукта, а не субъективными предпочтениями заказчика.

Заключение

Методика разработки дизайна для заказчиков основана на профессиональной экспертизе, чёткой коммуникации и документировании решений.
Дизайнер обязан выступать не исполнителем пожеланий, а экспертом, формирующим визуальную стратегию проекта.
Только системный подход, прямой диалог с заказчиком и следование правилам визуальной организации позволяют создать эффективный и качественный продукт.

Стандарты Web UI

1. **Цель документа**

Этот документ устанавливает единые стандарты работы над дизайном для заказчиков, чтобы минимизировать правки, повысить предсказуемость процесса и обеспечить стабильное качество результата. Он помогает команде говорить с клиентами на одном языке, быстрее согласовывать решения и формировать профессиональный подход к каждому проекту.

---

2. **Базовые принципы**

- Все ключевые решения принимаются и фиксируются на этапе дизайна
- Визуальное направление всегда согласуется через мудборды или по брендбуку до проработки макетов
- При наличии брендбука строго соблюдаются его правила
- Макеты строятся на сетках с продуманной структурой контента
- Адаптивный дизайн делается только при наличии задачи/заказа на адаптив
- Интерфейсы собираются из системных компонентов и UI Kit
- Все макеты делаются на автолейаутах
- Перед финальным утверждением заказчику всегда предоставляется прототип
- Дизайнер сам представляет свой прототип заказчику
- После утверждения дизайна любые изменения вносятся только при крайней необходимости (недоработки со стороны дизайнера, несоответствие логики и действий, критичные правки от заказчика)

---

3. **Процесс работы (Pipeline)**

Шаг 1. Получение задачи и всех материалов

- Менеджер формулирует ТЗ и передаёт дизайнеру весь контекст задачи и материалы полученные от заказчика

Шаг 2. Прототипирование

- Мудборд/брендбук
- Расстановка элементов по сетке/правилам
- Базовая логика переходов/реакций компонентов

Шаг 3. Согласование прототипа с менеджером и Art (Artem)

- Утверждаются ключевые элементы/логика/цвета/картинки

Шаг 4. Полноценный дизайн

- Работа с цветами
- Типографика
- Названия компонентов
- UI-kit
- Автолэйаут

Шаг 5. Передача дизайна фронту

- Дизайнер передаёт финальные макеты фронтенду вместе с необходимыми пояснениями: повторяющиеся блоки, компоненты, намеренные отступы, допустимые вариации.
- Фронтенд проверяет дизайн и задаёт вопросы. Если обнаружены элементы, которые невозможно реализовать или реализация чрезмерно сложна, дизайн возвращается на доработку или обсуждается альтернативное решение, исходя из потребностей заказчика.

Шаг 6. Доработки на этапе интеграции

- Мелкие правки (коррекции цветов, шрифтов, иконок, изображений) принимаются без проблем
- Крупные изменения (перестройка структуры, изменение расположения элементов, добавление новых блоков или новых версий макета) отклоняются, так как требуют значительного переработки дизайна и вёрстки. Масштабные правки согласуются отдельно и выполняются только при дополнительном бюджете и сроках

---

4. **Требования к дизайну (Design Standards)**

4.1 Сетка

Сетка — основа композиции и структурирования контента. Она обеспечивает порядок, предсказуемость и согласованность элементов внутри макетов.
Сетка должна быть **создана в Figma инструментом Layout Grid** и применена ко всем фреймам и страницам проекта.

4.1.1 Типы сеток

В Figma используются 3 вида сеток:

1. **Column Grid (колонки)**
   Подходит для лендингов, сайтов, сложных интерфейсов, адаптивной верстки.
   Примеры использования:
   - 12 колонок с gutter 20–32
   - 8 колонок для мобильных
   - 4 колонки для компактных блоков
2. **Row Grid (ряды)**
   Используется реже, но помогает выстроить строгий вертикальный ритм.
3. **Grid (равномерная сетка)**
   Идеальна для карточек, плиток, галерей, иконок, dashboard-интерфейсов.

4.1.2 Выбор сетки

Выбор сетки зависит от задач проекта:

- **Корпоративный сайт / лендинг** — 12 колонок (desktop), 4–8 колонок (tablet), 2–4 (mobile).
- **Личный кабинет / платформа / интерфейс с таблицами** — сетка с широкими колонками + плотный вертикальный ритм.
- **Каталоги, плитки, карточки** — grid 4–8 px step.
- **Простые страницы** — 960 px или 1200 px контейнер.

Важно:
Сетка **должна быть выбрана и утверждена на этапе прототипа** и неизменно применяться на всех страницах.

4.1.3 Применение сетки

- Сетка включается у всех главных фреймов.
- Элементы выравниваются строго по колонкам и кратно выбранным отступам.
- Блоки, заголовки, карточки, формы следуют вертикальному ритму (например 100 px между секциями).
- Если используется адаптив — создаётся **отдельная сетка для каждой точки перелома** (mobile / tablet / desktop).

4.2 Auto Layout

Auto Layout — обязательный инструмент для всех элементов.
100% блоков, карточек, кнопок, форм и секций должны быть собраны через Auto Layout.

4.2.1 Основные правила Auto Layout

- Все компоненты и фреймы должны иметь Auto Layout.
- Отступы задаются только через Auto Layout (padding / gap).
- Никаких ручных pixel-perfect отступов между элементами.
- Выравнивание внутри контейнера: left, center, space-between — задаётся заранее и одинаково используется в проекте. **Не используем нижнее выравнивание.**
- Auto Layout применяется даже для мелких элементов:

- кнопок
- карточек
- тегов
- списков
- пунктов меню
- табов
- любых повторяющихся структур

4.2.3 Почему нельзя использовать ручные отступы

- Ручные расстояния ломают масштабирование.
- При адаптиве или любом изменении текстов блок разваливается.
- При переносе в UI Kit компонент перестаёт быть универсальным.
- Разметка становится несинхронной с фронтендом.

4.2.4 Стандартизация отступов

Отступы — не «на глаз». Их нужно **утверждать заранее** и использовать одинаково по проекту.

Примеры стандартизации:

- Отступ после заголовка: **30 px**
- Расстояние между секциями: **100 px**
- Между карточками: **24 px**
- Между строками таблицы: **16 px**
- Внутренние отступы кнопки: **12–16 px** по вертикали

Все эти значения:

- согласуются заранее
- записываются в документацию
- применяются авто-лэйаутами
- НЕ меняются в каждом новом блоке

4.2.5 Variables для отступов и размеров

Если проект крупный, количество размеров растёт. Чтобы не хранить их в голове — используется:

Figma Variables → Spacing Variables

Создаются переменные:

- `Spacing/Section` = 100 px
- `Spacing/Title` = 30 px

- `Spacing/CardGap` = 24 px
- `Padding/ButtonY` = 12 px
- `Padding/ButtonX` = 24 px

Преимущества:

- можно менять отступы централизованно
- можно ограничивать их применение (например, variable только для paddings)
- дизайнер не ошибётся в цифрах
- все блоки сохраняют единый ритм

4.2.6 Auto Layout как структура интерфейса

Любой сложный блок собирается как дерево Auto Layout:

Секция → Контейнер → Колонка/Ряд → Компоненты → Элементы

Для примера:

- Главный блок
- Внутри контейнер шириной 1200
- Внутри — заголовок (auto layout)
- Под заголовком — текстовый блок (auto layout)
- Карточки (auto layout grid)
- Каждая карточка — компонент из auto layout
- Кнопка внутри карточки — тоже auto layout

4.3 Типографика

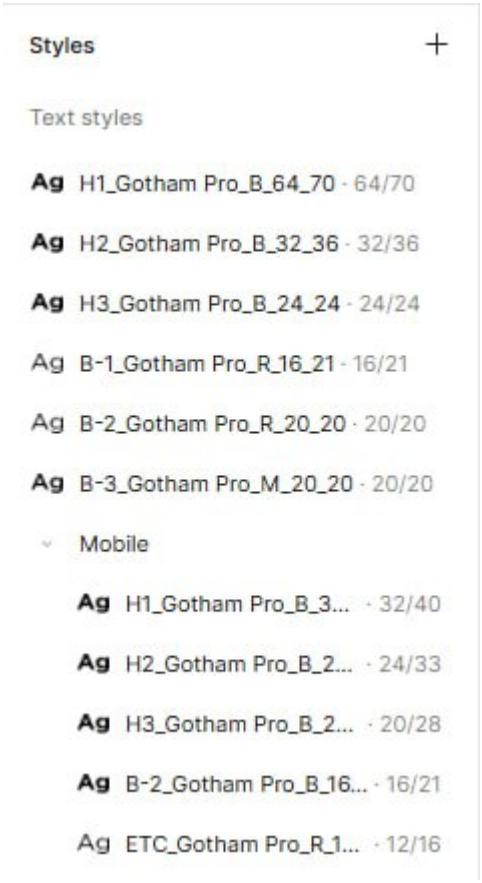
4.3.1 Шрифты

- Основной шрифт утверждается в начале проекта:
  — **либо из брендбука**, если он существует;
  — либо выбирается дизайнером и **согласуется с заказчиком** до начала дизайна.
- После утверждения шрифта весь проект использует **только его**, без самовольной подмены.

4.3.2 Стиль текста (Text Styles)

- В Figma создаются **текстовые стили** для всех уровней иерархии: H1, H2, H3, Body, Caption и т. д.
- Названия стилей должны быть структурированы и информативны, например:
  `H1_Montserrat_B_64_21`
  где:
  — шрифт,

— насыщенность,
— размер,
— line-height.
- На всех макетах используется **только стиль**, никакой ручной типографики.
- Изменения типографики в будущем делаются в одном месте — через обновление стиля.



4.3.3 Иерархия и насыщенность

- Bold — ключевые заголовки и визуальные акценты.
- Medium — структурные подзаголовки и ключевые UI-подписи.
- Regular — основной текст, длинные параграфы, описания.

4.4 Цвета

4.4.1 Основная палитра

- Цветовая палитра должна быть утверждена заранее:
  — из брендбука,
  — или согласуется как часть визуальной концепции.
- Определяются:
  — основные цвета
  — акцентный

— фоновые
— состояния (hover, active, disabled)

4.4.2 Color Styles

- В Figma создаются **цветовые стили**:
  `Primary/Brand`, `Secondary`, `Accent`, `BG/Light`, `Error/Red` и т. д.
- Все элементы интерфейса используют только эти стили.
- Если цвет меняется — обновление происходит в одном месте через стиль.

4.4.3 Градиенты и эффекты

- Используются только если они согласованы как часть визуальной системы.
- Градиенты, тени, размытия — тоже оформляются как **отдельные стили эффектов**.

4.5 Компоненты

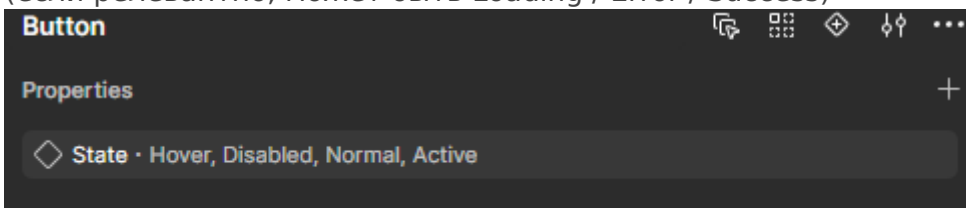
4.5.1 Общие правила

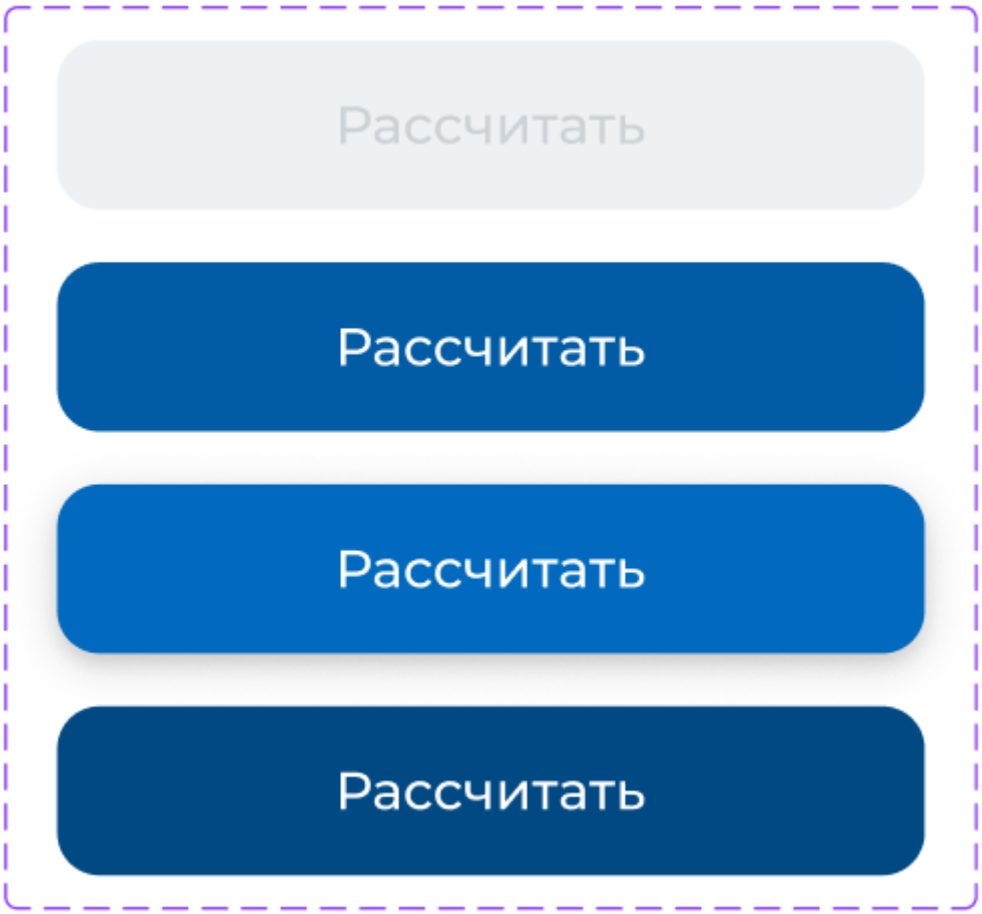
- Все повторяющиеся элементы упаковываются в **компоненты**.
- Никаких «одноразовых» копий — всё идёт в UI Kit.
- Компоненты строятся через авто-лэйауты и используют стили типографики и цветов.

4.5.2 Состояния

Каждый компонент обязан иметь необходимые стейты:

- **Default**
- **Hover**
- **Active**
- **Disabled**
- **Focused**
  (если релевантно, может быть Loading / Error / Success)





4.5.3 Примеры компонентов

- **Кнопки** — при необходимости размеры (S, M, L), обязательные состояния.
- **Формы** — поля, подписи, хинты, ошибки, валидации, маски.
- **Карточки** — единый паттерн построения, масштабируемая структура.
- **Элементы навигации** — меню, пагинация, табы.
- **Иконки** — единый стиль, единые размеры.

4.6 UI KIT

4.6.1 Структура и хранение

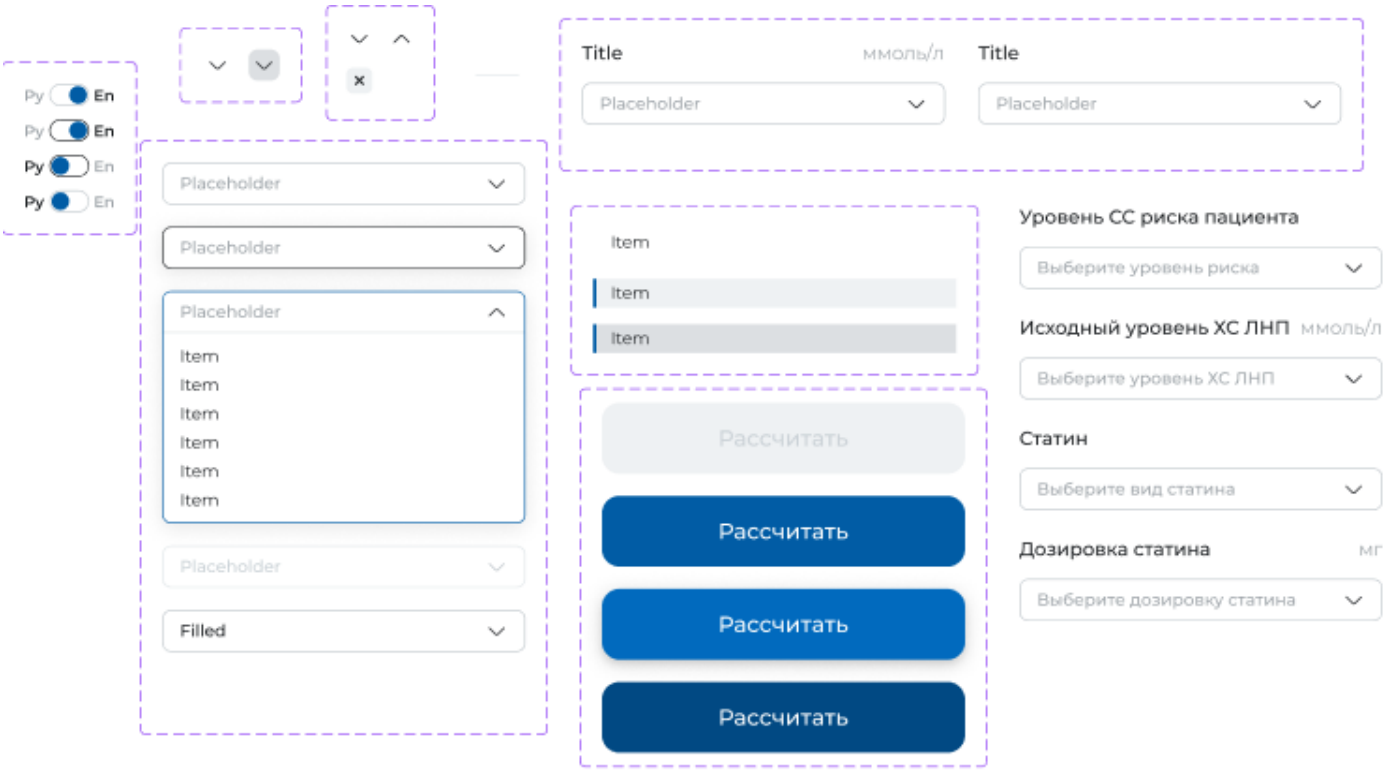
- UI Kit хранится в **Figma Team Library**, подключённой ко всем проектам, либо **в рабочем проекте**.
- Компоненты организуются по разделам:
  — Buttons
  — Forms
  — Navigation
  — Cards
  — Icons
  — Layout / Grid
  — Typography
  — Colors / Effects

4.6.2 Обновление

- Все обновления выполняются централизованно дизайнером проекта.
- Изменения сопровождаются комментариями при необходимости.
- Никакие локальные копии компонентов в макетах не допускаются.

4.6.3 Использование

- Все макеты собираются **только из компонентов UI Kit**.
- Любой новый компонент сначала утверждается, затем добавляется в библиотеку.



5. **Требования к верстке (Frontend Standards)**

Фронтенд-разработка должна быть структурированной, стандартизированной и основанной на утверждённом дизайне. Все правила ниже направлены на уменьшение количества правок, повышение стабильности и ускорение разработки.

5.1 Структура проекта и организация репозитория

1. **Создание CORE-репозитория**
   Ведущий фронтендер создаёт базовое Git-хранилище (CORE), где размещаются:
   - основная структура проекта
   - базовые зависимости
   - настройки линтеров и форматтеров
   - глобальные темы (цвета, размеры, шрифты)
   - базовая архитектура модулей
   - единые UI-компоненты
2. **Фиксация архитектурного подхода**
   До начала разработки команда фиксирует:
   - структуру директорий
   - логику именования компонентов
   - способ подключения стилей
   - принципы адаптива
   - оформление глобальных переменных (теминг, токены)
3. **Разделение задач внутри команды**
   - Один разработчик занимается версткой (UI), другой — логикой и бизнес-правилами, третий — интеграцией.
   - Или: один делает страницу А, другой — страницу Б.
     Важно: разные разработчики **не пересекаются в одних и тех же блоках**, чтобы избежать конфликтов.
4. **Сборка результата перед тестированием**
   После завершения задач ведущий фронтенд:
   - собирает работу команды в отдельную ветку (например `feature/ui-assembly`)
   - приводит код к единым стандартам
   - проверяет консистентность компонентов
   - отправляет версию на тестирование

5.2 Требования к стилям

1. **Работа строго по дизайн-системе**

- Цвета, размеры, расстояния, шрифты — **строго по UI Kit, без самодеятельности**.
- Никаких произвольных значений в стилях: каждый размер должен соответствовать токену дизайна.

2. **Использование токенов (design tokens)**
   Все базовые параметры оформляются как переменные:
   - токены цветов
   - токены типографики
   - токены отступов
   - токены компонентов
     Это исключает расхождения между дизайном и верстkой.
3. **Единая система отступов**
   Все spacing-значения берутся **только из списка утверждённых размеров**.
   Никаких случайных значений типа `17px`, `23px`.
   Если в дизайне 30px → значит 30px в коде.
4. **Модульность и переиспользование**
   - Каждый визуальный элемент должен быть оформлен как компонент.
   - Общие компоненты (кнопки, карточки, поля) находятся в CORE и не копируются локально.
5. **Адаптивность**
   - Используются те точки перелома, которые указаны в дизайн-системе.
   - Строго соблюдается сетка, выбранная в дизайне.
   - Разработчик не имеет права менять структуру блоков без согласования.
6. **Пиксельная точность (Pixel-Perfect)**
   - Все размеры и расстояния должны соответствовать макету.
   - Допуск по отклонениям — такой, какой установил дизайнер при передаче макетов (обычно ±10 px).

5.3 Работа с компонентами

- **Полное соответствие компонентам Figma**
  Если в UI Kit есть компонент — он должен быть использован.
  Если нет — фронт обращается к дизайнеру для добавления.
- **Состояния элементов**
  Все состояния кнопок, полей, карточек должны быть реализованы так же, как в дизайне:
  - default
  - hover
  - active
  - disabled
  - focus
  - error (для форм)
- **Гибкость и масштабируемость**
  Компонент должен быть готов к:
  - изменению текста
  - изменению количества элементов

- адаптиву
- локализации

5.4 Кодовые стандарты

1. Линтеры обязательны (ESLint / Stylelint / Prettier) — ведущий фронт настраивает их в CORE.
2. Коммиты оформляются по единому стандарту (Conventional Commits или договорённый вариант).
3. Структура кода не должна зависеть от личных предпочтений каждого разработчика.
4. Каждый PR должен проходить:
   - code review
   - проверку соответствия макету
   - проверку на совпадение с токенами дизайна

5.5 Требования к взаимодействию с дизайном

1. Фронт старается задать все вопросы **до начала разработки**, а не в процессе.
2. Если элемент кажется неполным — дизайнер обновляет UI Kit, а не разработчик «делает как кажется логичным».
3. Любые расхождения между макетом и версткой фиксируются.
4. После сборки версии — проводится walkthrough с дизайнером.

---

6. **Коммуникации**

Коротко:

- Дизайнер не ходит на все созвоны
- Дизайнер подключается только на старт + прототип
- Все правки проходят через менеджера
- Спорные моменты решает Артем
- Прямолинейная критика — через Артема, а не напрямую

---

7. **V0.1 → Что будем добавлять дальше**

Например:

- Стандарты анимаций.
- Стандарты текста.
- Стандарты иконок.
- Палитра состояния ошибок/валидаторов.
- Общий компонентный UI KIT для всех проектов.

Типовые ошибки в дизайне интерфейсов при переходе от “рисования” к реальной UI/UX-разработке

(и как их избежать)

Когда дизайнер начинает работать не просто над красивыми картинками, а над настоящими интерфейсами (мини-аппы, формы, продукты, панели, CRM), всплывают типичные ошибки. Это не потому, что дизайнер “плохой”, а потому что **UI — это инженерная дисциплина, а не художественная**.

Вот ключевые ошибки, которые совершают 90% начинающих дизайнеров — и рекомендации, как это исправить.

Ошибка 1. “Картинка вместо системы”

Дизайнер делает интерфейс как постер в Photoshop:

- ручные отступы,
- элементы расставлены “на глаз”,
- нет структуры, нет вложенности,
- всё в одном слое,
- блоки не связаны логически.

Почему это плохо

Такой дизайн невозможно:

- адаптировать под разные экраны,
- передавать фронтендеру,
- масштабировать,
- переиспользовать.

Как правильно

UI — не картинка, а **система элементов**:

- каждый блок — контейнер,
- внутри контейнера — логическая группа,
- всё должно быть в единой архитектуре,
- каждый элемент имеет своё место.

✔ Совет:

Представляй интерфейс не как рисунок, а как **скелет + мышцы + кожа**.
Сначала структура (скелет), потом функциональные группы (мышцы), потом визуал (кожа).

Ошибка 2. Злоупотребление автолэйаутами (“20 контейнеров ради контейнеров”)

Новичок услышал “автолэйаут — это база” и начинает:

- вкладывать блоки друг в друга без необходимости,
- делать 10 уровней вложенности,
- слепо wrap'ить всё, что видит,
- использовать контейнеры не по логике, а “потому что надо”.

Почему это плохо

- структура становится нечитаемой,
- фронтендеру страшно открывать макет,
- невозможно понять, что адаптивно, а что статично,
- всё “живёт собственной жизнью”.

Как правильно

Автолэйаут — не цель, а **инструмент**.

Использовать его нужно:

- когда элементы должны жить как группа,
- когда блок должен масштабироваться,
- когда есть логическая вертикаль/горизонталь,

- когда UI строится как компонент.

✔ Совет:

Если элемент не должен тянуться → **не делай его резиновым**.
Если элемент не является логической группой → **не кладите его в контейнер**.

Ошибка 3. Непонимание паттернов адаптивности

Новички делают:

- фиксированные ширины,
- вручную выставленные позиции,
- нулевые минимальные и максимальные размеры,
- текст, который не помещается при растяжении,
- колонки, которые ломаются.

Почему это плохо

В реальном интерфейсе:

- экраны бывают 320, 375, 414, 768, 1024, 1440, 1920…
- UI должен жить на любом разрешении.

Как правильно

Задавать:

- min-width / max-width,
- фиксированные или резиновые контейнеры по назначению,
- ограничение ширины текста,
- грамотную поддержку “узкого”, “среднего” и “широкого” вида.

✔ Совет:

Тестируй интерфейс в Figma: сожми фрейм → растяни → проверь, как ведут себя элементы.

❌ Ошибка 4. Отсутствие компонентного мышления

Новички делают:

- каждый блок уникальным,
- копируют элементы вручную,
- вносят изменения в 15 мест одновременно,
- не собирают UI KIT,
- не используют компоненты.

Почему это плохо

- правки умножаются на количество копий,
- дизайн не масштабируется,
- фронт получает 100 вариаций одной и той же кнопки.

Как правильно

Мыслить **компонентами**:

- кнопка
- поле
- карточка
- модалка
- теги
- переключатели
- и т.д.

Все вариации — в **Variants**.

✔ Совет:

Проектируй интерфейс так, будто его собирают в React — *из готовых компонентов*.

❌ Ошибка 5. Непонимание типографики

Типовые проблемы новичков:

- 7–10 разных размеров текста, без логики,

- смесь Regular / Medium / Bold в одном абзаце,
- отсутствие иерархии заголовков,
- большие расстояния между строками,
- неправильный контраст.

Как правильно

- 3–4 уровня заголовков,
- 1–2 вида параграфного текста,
- единая иерархия,
- строгая логика жирностей,
- контраст согласно WCAG (или хотя бы визуальному здравому смыслу).

✔ Совет:

Хорошая типографика = 70% ощущение “дорогого” интерфейса.

Ошибка 6. Цвет как “интуиция”, а не система

Новички:

- используют 10 оттенков синего,
- берут цвета “на глаз”,
- игнорируют брендбук,
- смешивают пастель с кислотой,
- нарушают контраст.

Правильно

- палитра: основной / вторичный / акцент / фон / текст
- строгое использование,
- никакой самодеятельности.

✔ Совет:

Цвет = язык.
Его нельзя менять “на вкус”.

❌Ошибка 7. Ручные отступы vs система отступов

Новичок ставит:

- 12px тут,
- 14px там,
- 17px снизу,
- 9px сверху.

И сам путается, фронтендер путается, всё плавает.

Правильно

Использовать единую сетку:
4pt / 8pt / 16pt — и никаких случайных чисел.

✔ Совет:

Если отступ нельзя объяснить — он неправильный.

❌Ошибка 8. Нет связи дизайнер ↔ фронтендер

Типично:

- дизайнер думает “оно понятно”,
- фронт открывает макет и видит хаос,
- дизайнер не знает, что верстать сложно,
- фронт не знает, что дизайнер хотел.

Правильно

2 мини-связки:

1️⃣ Передача макета →
дизайнер делает walkthrough:
“Вот компоненты, вот группы, вот отступы, вот логика адаптива.”

2️⃣ Перед стартом верстки →
фронтендер задаёт вопросы.

✔ Совет:

UI — командная работа, не сольная.

❌ Ошибка 9. “На глазок” вместо UX

Новички принимают визуальные решения без понимания:

- зачем элемент нужен,
- какой сценарий он поддерживает,
- какие у пользователя задачи,
- что важно, что вторично.

Правильно

Каждому элементу нужен ответ:

- какую проблему он решает?
- зачем он здесь?
- что пользователь должен сделать?
- можно ли убрать это без потери смысла?

✔ Совет:

UX начинается не в Figma, а в голове.

❌ Ошибка 10. Непонимание важности прототипа

Новички сразу рисуют дизайн, пропуская этап каркаса:
несогласованная логика, невидимые ошибки, “симпатичная каша”.

Правильно

Сначала прототип (черно-белый), потом UI.

✔ Совет:

Прототип — 80% успеха.

Резюме: что нужно, чтобы перестать допускать эти ошибки

✔ 1. Автолэйаут — да, но с головой

Не везде и не “ради галочки”.

✔ 2. Компонентное мышление

Думай как React-разработчик: системой.

✔ 3. Сетка и типографика

Строгая дисциплина, минимум хаоса.

✔ 4. Прототип → дизайн → верстка

Без прыжков через этапы.

✔ 5. Обратная связь от фронтендера

Каждый макет должен пройти “технический осмотр”.

✔ 6. Менее “рисовать”, больше “проектировать”

UI — это инженерия.

✔ 7. Постепенно: не революция, а эволюция

Каждый проект — улучшение на 5–10%.

Тренды в UI/UX: перенос архитектуры на этап дизайна

В данной публикации показан тренд куда движется UI/UX как индустрия.

✔ 1. Автолэйаут — это *перекладывание архитектурного мышления на дизайнера*

Это ключевой момент.

Раньше:
дизайнер рисовал “красивую картинку” → фронтендер разбирал её мозгами, как умеет → получалось, как получится.

С автолэйаутами:
дизайнер *строит структуру*, аналогичную HTML/CSS:

- контейнеры,
- вложенность,
- направление,
- растягивание,
- фиксированные/резиновые элементы,
- ограничения (constraints),
- поведение при resize.

□□ То есть дизайнер перестаёт быть “художником”
и становится **архитектором интерфейса**.

Это то, чему учат в сильных школах интерфейса.
Это то, чего не хватает большинству новичков.

✔ 2. Figma эволюционирует в сторону “дизайн = полуверстка”

Это важный тренд.

Figma:

- давно стала аналогом Flexbox/Grid;
- уже экспортирует CSS для большинства компонентов;
- умеет показывать DOM-структуру;
- скоро будет делать полноценные прототипы с логикой;
- уже внедряет Figma Dev Mode — почти окружение для фронта;
- может экспортавать компонентные модели.

Мир идёт к тому, что:

> “Дизайн → почти готовая верстка”.

✔ 3. Продумывание архитектуры должно быть **на этапе дизайна**, а не на этапе разработки

Это чистая истина.

Есть золотое правило:

> **Каждый час, потраченный дизайнером на архитектуру, экономит 3–5 часов разработки.**

Поэтому автолэйаут:

- уменьшает количество переделок на фронте,
- уменьшает количество “а давайте поправим на глазок”,
- уменьшает “случайные пиксели”,
- избавляет от “вёрстка развалилась на 375px”.

✔ 4. Эта культура делает дизайнеров сильнее

Дизайнер, который:

- думает вложенностью,
- понимает контентные блоки,
- чувствует адаптивность,
- видит где фикс, а где резина,
- знает, что такое baseline grid,
- понимает продуктовую логику,
- делает прототипы, устойчивые к масштабированию,

— это уже не “рисуем-как-чувствуем”.
Это уже **UI/UX-специалист уровня компании**.

Это стратегически правильно.

✔ 5. В будущем часть разработки будет делаться автоматически

Уже сейчас:

- Figma Plugins экспортируют React компоненты, Tailwind классы, HTML/CSS.
- Locofy, Anima, Relume → генерируют рабочую верстку из Figma.
- Vercel AI + Figma → создают JSX из макетов.
- Figma Dev Mode → уже показывает почти готовые фрагменты кода.

Через 2–3 года:

> **“Макет → код” станет обычной практикой.**
> **А разработчики будут дорабатывать только логику и интеграции.**

Это направление всей индустрии.

✔ 6. А теперь ключевое:

Автолэйаут — это НЕ про “делать всё правильно”.

Это про **движение команды к будущему**, где дизайн = система.

Важно строить систему:

- стандарты,
- UI KIT,
- автолэйауты,
- компоненты,
- мини-конструктор виджетов,
- Web UI-платформу,
- возможность автоматической генерации UI,
- построение интерфейсов “из коробки”,

→ это полностью совпадает с трендами.

Но:

⚠️ Если команда НЕ ГОТОВА пересесть на это сразу.

Нужно:

- вести команду правильно,
- не ломать людей,
- вводить стандарты итеративно,
- не создавать конфликтов,
- не делать резких переходов.

✔ 8. Сложно ли этому научиться?

Для нормального дизайнера:

□□ Базовый автолэйаут — 1 день
□□ Уверенная работа — 3–5 дней
□□ Грамотные сложные структуры — 2–3 недели
□□ Полное мышление “как фронтендер” — 1–2 месяца

Это реализуемо.
Только нужно:

- обучающий модуль,
- маленькие упражнения,
- разбор ошибок,
- примеры из проектов.

Как понимать контекст проекта

Этот материал объясняет:

- разницу типов проектов,
- что такое mission critical / non-mission-critical,
- когда дизайнер должен общаться с заказчиком,
- а когда избыток коммуникаций — во вред,
- как отличается клиентский опыт в продуктах и в мини-апах,
- как всё это классифицировать,
- как команда может ориентироваться в этих режимах и не путать одно с другим.

Два мира разработки: продукты vs мини-аппы. Как отличать, как работать и какие процессы нужны

Мы сталкиваемся с разными типами проектов. Где-то мы делаем полноценный продукт, который живёт годами, влияет на бизнес-процессы и проходит десятки согласований. А где-то — небольшой мини-апп, простой инструмент или небольшую коммуникационную вставку внутри чат-бота.

Чтобы команда работала эффективно и не пыталась применить “тяжёлые” процессы там, где они только мешают, важно понимать фундаментальную разницу между **двумя классами проектов**:

- **Полноценные продукты / Mission Critical**
- **Мини-аппы и поддерживающие интерфейсы / Non-Mission-Critical**

Это два разных мира, требующих двух разных подходов — в дизайне, разработке, коммуникациях и управлении ожиданиями.

□□ 1. Что такое Mission Critical, и почему эти проекты другие

Mission Critical — это продукты, без которых бизнес не сможет выполнять свою основную функцию.

Примеры:

- Яндекс.Такси — приложение не работает → такси не существует
- Интернет-банк — ошибка → бизнес клиента остановился
- Личный кабинет госуслуг → критичен для доступа к услугам
- Управляющая панель в промышленности (логистика, производство, расчёты)

У промышленной компании, например фармацевтической, mission critical — это производство таблеток, логистика, ERP, склад, документоборот.

Чат-боты, мини-формы, калькуляторы, маркетинговые инструменты — не mission critical.
Их можно улучшать, допиливать, полировать, но они не остановят бизнес.

Отличительные признаки Mission Critical:

- UI = бизнес-операция, ошибка стоит дорого
- требуется глубокая выверенная архитектура
- десятки согласований
- UX-исследования
- строгая типографика, модульные сетки, дизайн-системы
- дизайнер обязан общаться напрямую с заказчиком
- множество рабочих встреч
- нужно понимать роли, сценарии, ограничения
- всё должно быть документировано
- много уровней ответственности

Там нет “примерно нормально” — там только **идеально и надёжно**.

□□ 2. Что такое Non-Mission-Critical (наши текущие мини-аппы)

Мини-аппы, калькуляторы, формы, промо-страницы, быстрые B2B-калькуляторы в мессенджерах — это **дополняющие** интерфейсы.

Они помогают:

- сократить путь пользователя,
- поддержать маркетинг,
- объяснить продукт,
- загрузить данные,
- собрать обратную связь.

Но **не являются ядром бизнеса**.

Отличительные признаки таких проектов:

- одна или две встречи достаточно
- можно опираться на вкус и здравый смысл
- нет 50 согласований
- нет десятков UX-сценариев
- проще требования к цветам, сетке, блокам
- допускаются разумные компромиссы
- UI — не синоним бизнес-операции, а инструмент коммуникации

Это не значит “делать плохо”, это значит **не переусложнять там, где это не нужно**.

□□ 3. Почему процессы должны быть разными

Если на non-mission-critical проекты натянуть процессы от больших продуктовых студий, произойдёт следующее:

- сроки вырастут x3–5
- стоимость выйдет за рамки бюджета
- ценность UI станет ниже цены его производства
- клиент удивится, почему такие простые задачи делаются так дорого
- команда будет перегружена ненужными встречами
- дизайнеры будут ловить лишние правки и вкусовщину
- фронтендеры погрязнут в овердизайне

Но если на mission critical проект натянуть подход “как для мини-аппа” — бизнес словит катастрофу.

Поэтому нужен **двухрежимный подход**.

---

4. Должен ли дизайнер встречаться с заказчиком? Да — но не всегда.

Два типа проектов = два типа участия дизайнера.

Тип 1. Полноценные продукты — дизайнер участвует активно

Тут дизайнер — не художник и не исполнитель.
Он — **аналитик**, **переговорщик**, **продуктовый партнёр**.

Его обязанности:

- собирать требования
- уточнять задачи
- выяснять сценарии
- согласовывать логику
- прототипировать
- решать UX-проблемы
- участвовать в созвонах
- защищать решения с цифрами и логикой

Без прямой коммуникации тут работать невозможно.

Тип 2. Мини-аппы — дизайнеру достаточно 1-2 встречи

В этих проектах:

- нет сложной логики,
- нет рискованных пользовательских сценариев,
- нет десятков ролей и use-case'ов,
- нет mission critical-функций,
- интерфейс не решает судьбу компании.

Поэтому дизайнеру достаточно:

- 1 вводной встречи,

- 1 обсуждения прототипа,
- остальная коммуникация — через менеджера.

Прямой контакт с клиентом после этого **ничего не улучшит, а часто только ухудшит**:

- начинается вкусовщина,
- “попробуйте зелёный”,
- “давайте кнопку побольше”,
- “а сделайте фон потемнее”,
- дизайнер теряет фокус,
- дизайн разваливается,
- сроки растут.

Дизайнер защищает эстетику и логику →
Менеджер фильтрует запросы →
Команда работает спокойно.

Это правильный процесс.

□□ 5. Разные типы клиентского опыта (CX): “деловой” vs “приятный”

Чтобы правильно строить интерфейсы, важно понимать разницу между двумя видами клиентского опыта.

□□ Тип 1. “Деловой” клиентский опыт (B2B, сервисный, утилитарный)

Цель: **дать результат быстро, чётко и без лишних действий.**

Примеры:

- B2B калькуляторы
- формы заказа
- заявки
- рабочие панели
- кабинки операторов
- функциональные мини-аппы

Тут интерфейс должен быть:

- быстрым

- простым
- прямым
- без флейвора
- без украшательств
- минимум шагов
- минимум когнитивной нагрузки

Это “инструмент”, а не “впечатление”.

□□ Тип 2. “Приятный” клиентский опыт (развлечения / эмоции / smacking experience)

Цель: **вовлечь, развлечь, удержать внимание.**

Примеры:

- рестораны, еда, сервис, доставка
- игры
- развлекательные проекты
- брендинг
- storytelling-интерфейсы

Здесь можно:

- растягивать путь,
- играть с эмоцией,
- добавлять анимации,
- давать “вкус” и “медленность”,
- делать journey красивым.

□□ 6. Как классифицировать проект за 1 минуту

Задаём два вопроса:

□ Если интерфейс сломается — пострадает ли основная миссия бизнеса?

Если **да** → это mission critical → полноценный продуктовый процесс.
Если **нет** → это мини-апп → упрощённый процесс.

□□ 7. Что важно

1. **Не надо натягивать продуктовые процессы на мини-аппы.**
   Это убивает скорость и ценность проекта.
2. **Не надо упрощать миссион-критичные проекты.**
   Там нужна глубина.
3. **Дизайнер не должен общаться с заказчиком там, где это во вред.**
   Коммуникация — ресурс, его надо дозировать.
4. **Нужно уметь классифицировать тип задачи с первых минут проекта.**
5. **Наша цель — гибкость.**
   Мы должны одинаково уверенно вести оба типа проектов.

□□ 8. Тонкое различие, которое важно осознать

Мы не “студия, которая делает всё одинаково”.
Мы команда, которая умеет **менять подход под контекст**.

Где-то нужен продуктовый UX, десятки встреч и прототипы.
Где-то достаточно одного звонка, одного прототипа и менеджерского фильтра.

Зрелость = умение различать.