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

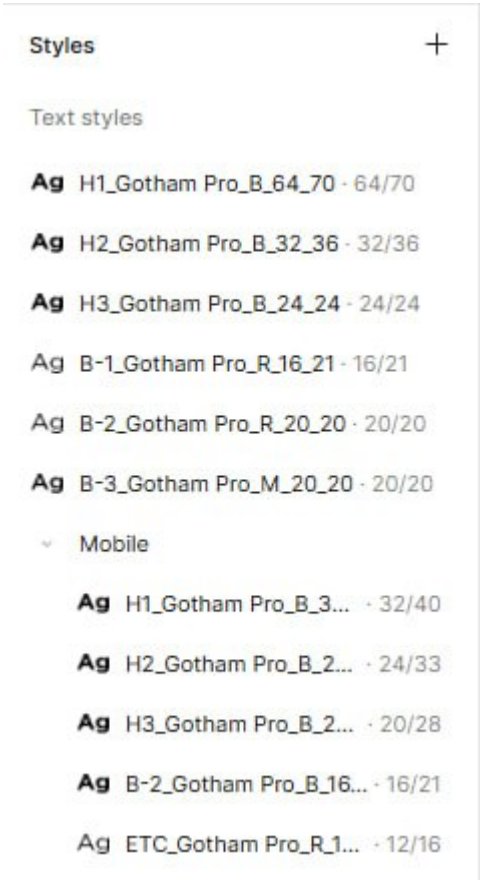
4.3.1 Шрифты

- 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

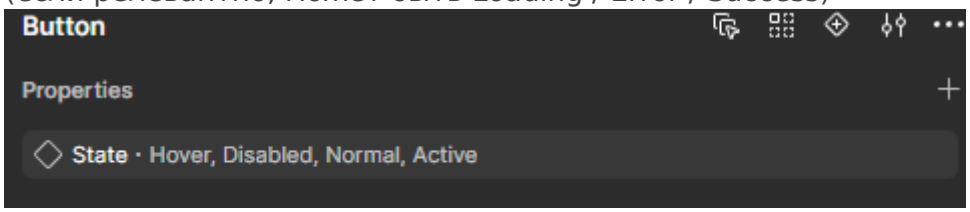
4.5.1 Общие правила

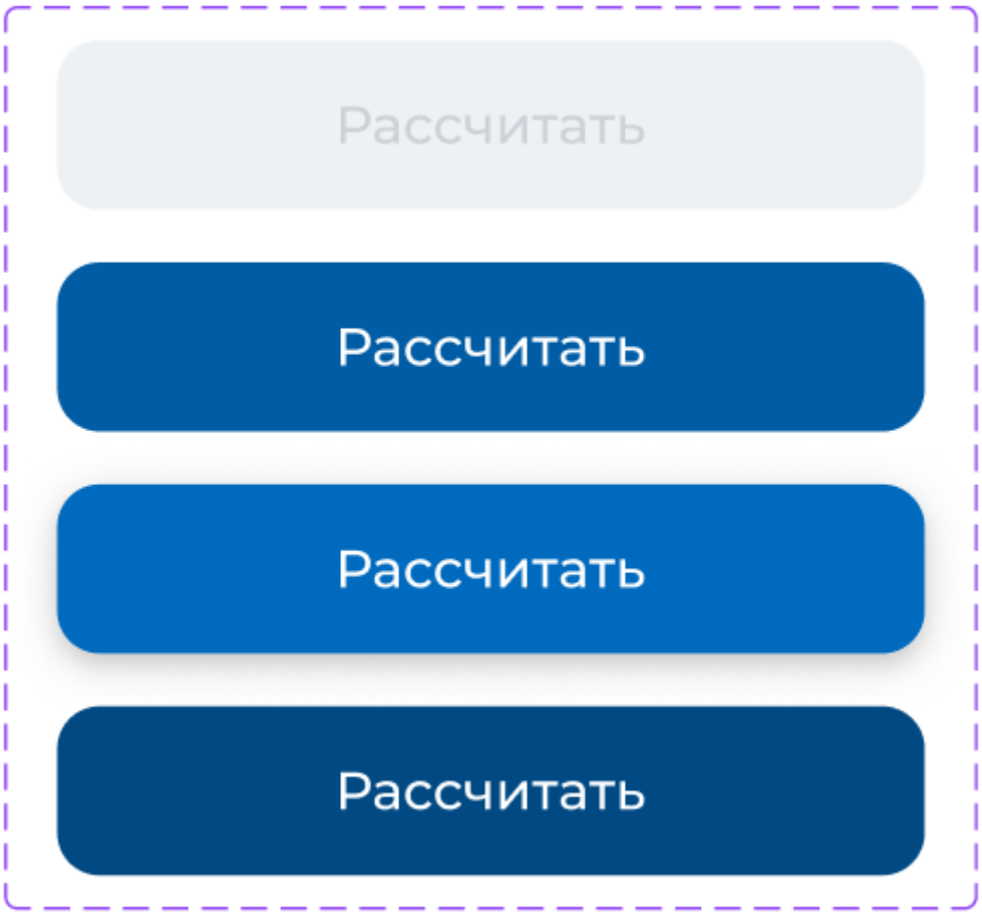
- Все повторяющиеся элементы упаковываются в **компоненты**.
- Никаких «одноразовых» копий — всё идёт в UI Kit.
- Компоненты строятся через авто-лэйауты и используют стили типографики и цветов.

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

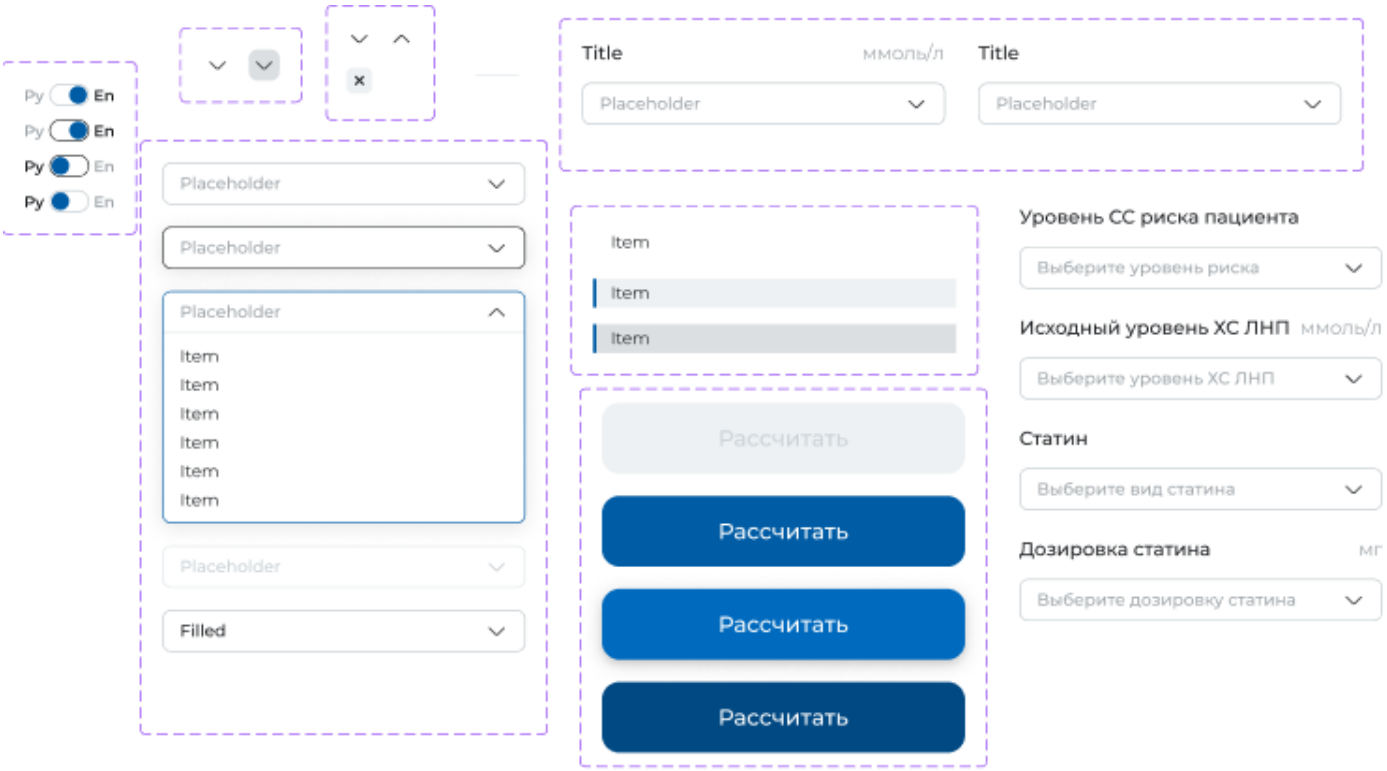
- UI Kit хранится в **Figma Team Library**, подключённой ко всем проектам, либо **в рабочем проекте**.
- Компоненты организуются по разделам:
  — Buttons
  — Forms
  — Navigation
  — Cards
  — Icons
  — Layout / Grid
  — Typography
  — Colors / Effects

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.

Версия #4
Artem Garashko создал 2025-11-23 10:10:38 UTC
Artem Garashko обновил 2025-11-25 21:03:02 UTC