Архитектура и Проектирование Цифровых Экосистем для Визуальных Творцов: Всесторонний Анализ Трендов 2025 Года и Техническая Спецификация для AI-Генерации

1. Введение: Смена Парадигмы в Веб-Присутствии Фотографа

К 2025 году концепция веб-портфолио фотографа претерпела фундаментальные изменения. Традиционный подход, рассматривающий сайт исключительно как статичную цифровую визитку или пассивную галерею изображений, окончательно устарел. В современной цифровой экономике сайт фотографа трансформировался в сложную, многокомпонентную платформу, объединяющую функции маркетингового инструмента, системы управления взаимоотношениями с клиентами (CRM), механизма электронной коммерции и иммерсивного сторителлинга.

Анализ текущего состояния индустрии веб-дизайна и технологий разработки показывает, что успешные цифровые продукты для фотографов требуют глубокой интеграции визуальной эстетики с жесткой технической эффективностью. Пользовательский опыт (UX) больше не ограничивается удобной навигацией; он охватывает скорость загрузки контента (Core Web Vitals), эмоциональный отклик от микро-взаимодействий и безопасность передачи цифровых активов.[1] Кроме того, демократизация технологий разработки, в частности использование нейросетей (LLM) для генерации кода, открывает перед индивидуальными творцами возможности создания систем уровня Enterprise, ранее доступных только крупным агентствам.[3]

Данный отчет представляет собой исчерпывающее руководство по созданию такой платформы. Первая часть посвящена анализу визуальных и поведенческих трендов 2025 года, определяющих ожидания аудитории. Вторая часть детально описывает

бизнес-логику и рабочие процессы (workflow) фотографа, которые должны быть оцифрованы. Третья и четвертая части содержат глубокую техническую документацию — от архитектуры базы данных до системных промптов для AI, позволяющих сгенерировать программный код описанной системы.

---

2. Визуальная Эстетика и UX-Паттерны 2025 Года

Дизайн 2025 года характеризуется отходом от стерильного минимализма начала 2020-х годов в сторону более насыщенных, тактильных и эмоционально резонирующих интерфейсов. Этот сдвиг обусловлен "цифровой усталостью" пользователей и стремлением к аутентичности.

2.1. Эра "Тактильного Максимализма" и "Nature Distilled"

Долгое время стандартом де-факто для портфолио считался "белый куб" — интерфейс, лишенный декоративных элементов, чтобы не отвлекать внимание от фотографий. Однако современные данные указывают на то, что в условиях перенасыщения контентом нейтральность становится синонимом невидимости. Тренды "Tactile Maximalism" (Тактильный максимализм) и "Nature Distilled" (Дистиллированная природа) выходят на первый план.[1]

"Тактильный максимализм" в контексте веб-дизайна подразумевает использование визуальных приемов, имитирующих физические свойства объектов: зернистость пленки, текстуры бумаги, глубокие тени и слои, создающие ощущение глубины экрана. Это не возврат к скевоморфизму 2010-х, а переосмысление материальности в цифровой среде. Для фотографа это означает, что интерфейс сайта должен дополнять атмосферу работ, а не просто служить контейнером.

Тренд "Nature Distilled" отражает стремление к органике. Палитры смещаются от чистых RGB-цветов к сложным, землистым оттенкам ("Dopamine colors", "Museumcore").[1] Типографика становится более выразительной и крупной, часто выполняя роль графического элемента, а не просто носителя информации ("Expressive Typography").[5] Шрифтовые пары подбираются таким образом, чтобы создавать контраст между массивными заголовками и функциональным наборным текстом, что улучшает сканируемость страниц.

2.2. Структурная Эволюция: От Masonry к Bento Grids

В организации контента происходит революция сеток. Классическая "кирпичная кладка" (Masonry Layout), долгое время доминировавшая в Pinterest и портфолио, уступает место более структурированным и информационно насыщенным "Бенто-сеткам" (Bento Grids).[6]

Феномен Bento Grid

Названная в честь японских коробок для ланча, Bento Grid представляет собой модульную систему, где контент распределен по прямоугольным блокам разного размера, но строго выровненным по сетке. Главное преимущество этого подхода для фотографа — возможность мультиформатного повествования. В отличие от Masonry, где все элементы равнозначны (обычно это просто изображения), Bento Grid позволяет смешивать в одном пространстве:

- Ключевое изображение (Hero shot).
- Карту с локацией съемки.
- Текстовый блок с описанием концепции.
- Технические данные (EXIF).
- Короткие видео-лупы (Reels/Shorts).
- Кнопки призыва к действию (CTA), например, "Забронировать дату".

Такая структура превращает просмотр портфолио из пассивного листания в исследование контекста.[6] Это коррелирует с трендом на "Scrollytelling" — повествование через скролл, где пользователь, прокручивая страницу, погружается в историю съемки, а не просто видит набор разрозненных кадров.[9]

2.3. Темный режим и Адаптивность

Поддержка "Dark Mode" в 2025 году перестала быть опцией и стала обязательным требованием.[5] Для фотосайтов это особенно критично, так как темный фон (зачастую не чистый черный #000000, а глубокий серый или темно-синий) усиливает восприятие цвета и контраста изображений, снижая нагрузку на глаза зрителя. Реализация темного режима требует переосмысления цветовой теории сайта: просто инвертировать цвета

недостаточно, необходимо балансировать контрастность текста и яркость изображений, чтобы избежать эффекта "гало".

Адаптивность также вышла на новый уровень. Речь идет не просто о том, чтобы сайт "влезал" в экран смартфона (Mobile-Friendly), а о полноценном Mobile-First опыте. Учитывая, что до 70% трафика на сайты фотографов приходит из социальных сетей через мобильные устройства, навигация должна быть оптимизирована под управление большим пальцем, а жесты (свайпы, тапы) должны быть естественными и интуитивными.[10]

---

3. Бизнес-Логика и Workflow Фотографа: Оцифровка Процессов

Эффективная CMS для фотографа должна быть построена вокруг его реальных рабочих процессов. Понимание жизненного цикла заказа — от первого контакта до сдачи готового материала — является ключом к проектированию архитектуры системы.[11]

3.1. Жизненный Цикл Проекта

Этап Workflow	Офлайн/Ручные действия	Цифровая реализация в CMS
1. Привлечение (Pre-Production)	Портфолио в Instagram, переписка в мессенджерах.	SEO-оптимизированные лендинги проектов, контактные формы с интеграцией в CRM, автоматический календарь занятости.
2. Съемка и Отбор (Production)	Съемка, копирование на жесткий диск, ручной отбор в Lightroom.	Загрузка превью в "облако" для чернового отбора клиентом (Proofing Gallery).

3. Согласование (Post-Production)	Отправка PDF-файлов или ссылок на Google Drive, переписка с номерами фото ("обработай 5678.jpg").	Интерактивная галерея с возможностью ставить "лайки" (Favorites), оставлять комментарии к конкретным точкам на фото.
4. Сдача и Продажа (Delivery)	Передача флешки или ссылки на облако.	Защищенная галерея с возможностью скачивания (Web-quality/High-Res). Upsell: продажа доп. фото и принтов через встроенный магазин.

3.2. Клиентский Пруфинг (Client Proofing)

Функционал пруфинга является критическим отличием профессиональной CMS от обычного конструктора сайтов. Он решает проблему коммуникации на этапе отбора. Система должна позволять создавать приватные коллекции, доступные по уникальной ссылке или паролю, где клиент может самостоятельно формировать списки "Избранного".13
Технически это требует реализации сложной модели прав доступа: клиент должен видеть только свои фото, иметь возможность взаимодействовать с ними (выбор, комментарий), но не иметь возможности скачать оригиналы до оплаты или завершения ретуши. Для защиты контента на этом этапе необходимо автоматическое наложение водяных знаков (Watermarking) на лету.15

3.3. E-commerce Специфика: Цифровые и Физические Товары

Монетизация архивов требует гибкой системы электронной коммерции. В отличие от продажи футболок, продажа фотографий имеет свои нюансы:

1. **Вариативность лицензий:** Одно изображение может продаваться как "Лицензия для соцсетей" (низкое разрешение, низкая цена), "Коммерческая лицензия" (высокое разрешение, RAW), или как физический объект (принт на холсте).[17]

2. **Безопасность доставки:** После оплаты цифрового товара система должна генерировать временную, подписанную ссылку (Signed URL) для скачивания, которая истекает через заданное время. Это предотвращает распространение прямой ссылки на файл на форумах и в пабликах.[18]
3. **Интеграция с лабораториями:** В идеальном сценарии заказ на печать должен автоматически перенаправляться в принт-лабораторию через API, минуя ручную обработку фотографом.

---

4. Техническая Архитектура: Современный Стек (2025)

Для реализации описанных требований к производительности, SEO и интерактивности, монолитные CMS (WordPress) уже не являются оптимальным выбором. Архитектура 2025 года базируется на принципе **Composable Web** (Компонуемый веб) с использованием Headless CMS и современных фронтенд-фреймворков.[20]

4.1. Обзор Технологического Стека

В качестве основы предлагается следующая конфигурация, обеспечивающая баланс между скоростью разработки (DX) и производительностью пользователя (UX):

Компонент	Технология	Обоснование выбора
Frontend Framework	**Next.js 15 (App Router)**	Лидер рынка. Поддержка React Server Components (RSC) для максимальной скорости загрузки и SEO. Встроенная оптимизация изображений. [22]
CMS (Content Backend)	**Sanity.io (v3)**	Headless CMS с концепцией "Content Lake". Гибкое моделирование данных,

		язык запросов GROQ, Real-time коллаборация. [24]
Database (User Data)	**PostgreSQL + Drizzle ORM**	Хранение реляционных данных (пользователи, заказы, лайки), которые сложно структурировать в документо-ориентированной CMS. [26]
Media Assets (DAM)	**Cloudinary**	Профессиональное управление медиа: авто-кроп, форматирование (AVIF/WebP), динамические водяные знаки, AI-ресайз. [15]
Payments	**Stripe**	Глобальный стандарт платежей. Удобный Checkout, мощные вебхуки для обработки успешных транзакций. [28]
Deployment	**Vercel**	Нативная поддержка Next.js, Edge Functions для быстрой доставки контента по всему миру.

4.2. Frontend: Next.js 15 и React Server Components

Использование Next.js 15 с App Router является стратегическим решением. Технология React Server Components (RSC) позволяет рендерить тяжелые компоненты (например, галереи с сотнями изображений) на сервере, отправляя на клиент только легкий HTML и минимально необходимый JavaScript. Это решает извечную проблему JS-тяжелых сайтов — долгую инициализацию (Hydration).[29]

Оптимизация Изображений (next/image)

Компонент <Image> в Next.js 15 берет на себя самую сложную часть работы фронтенда портфолио:

- **Prevention of CLS (Cumulative Layout Shift):** Резервирует место под картинку до её загрузки, предотвращая "прыжки" контента, что критично для UX и ранжирования в Google.
- **Responsive Sizing:** Автоматически генерирует srcSet, отдавая мобильному телефону маленькую картинку, а Retina-дисплею — большую.
- **Lazy Loading:** Изображения вне видимой области экрана не загружаются, экономя трафик.[22]

4.3. Backend Content: Философия Sanity.io

Выбор Sanity обусловлен тем, что это не просто CMS, а платформа структурированного контента. В отличие от традиционных CMS, где контент часто "заперт" в HTML-блоках, Sanity хранит данные в чистом JSON-формате. Это позволяет использовать один и тот же контент для веб-сайта, генерации PDF-презентаций или мобильного приложения.[31] Язык запросов GROQ позволяет извлекать именно те данные, которые нужны, трансформируя их на лету (например, подсчитывая количество фото в альбоме прямо в запросе).[33]

---

5. Моделирование Данных и Схемы

Разработка правильной структуры данных — фундамент системы. Мы используем гибридный подход: контент (проекты, фото, тексты) хранится в Sanity (NoSQL), а транзакционные данные (пользователи, заказы, избранное) — в PostgreSQL (SQL).

5.1. Схемы Sanity (Content Layer)

Ниже приведены концептуальные схемы для Sanity, которые необходимо будет передать

AI для генерации кода.

Схема Project (Проект/Съемка)

Основная сущность портфолио.

- title (String): Название съемки.
- slug (Slug): URL-адрес (напр., /projects/wedding-anna-igor).
- category (Reference -> Category): Свадьба, Портрет, Fashion.
- coverImage (Image): Обложка для списка проектов.
- gallery (Array of Images): Основной массив фотографий. Каждое изображение в Sanity — это объект, который может содержать дополнительные поля:
  - caption (String): Подпись.
  - exif (Object): ISO, выдержка, камера (заполняется автоматически через вебхук при загрузке).
- content (Portable Text): Текстовое описание с возможностью вставки видео, цитат и карт.
- parameters (Object): Настройки отображения (тип сетки: Bento/Masonry, цвет фона для этой страницы).

Схема Settings (Глобальные настройки)

- siteTitle (String).
- seo (Object): Глобальные мета-теги, OpenGraph картинка.
- navigation (Array): Пункты меню.
- watermark (Image): Файл водяного знака для наложения через Cloudinary.

5.2. Схема PostgreSQL (User & Transaction Layer)

Для реализации функций CRM и магазина необходима строгая реляционная структура.[26]

Таблица users

Хранит данные как администратора, так и клиентов.

SQL

```sql
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  role VARCHAR(50) DEFAULT 'client', -- 'admin', 'client'
  created_at TIMESTAMP DEFAULT NOW()
);
```

Таблица favorites (Связь многие-ко-многим)

Ключевая таблица для пруфинга. Связывает пользователя и конкретное фото. Обратите внимание: так как фото хранятся в Sanity, мы храним sanity_asset_id как строку.

SQL

```sql
CREATE TABLE favorites (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  project_slug VARCHAR(255) NOT NULL, -- Для быстрой фильтрации по проекту
  sanity_photo_id VARCHAR(255) NOT NULL, -- ID ассета из Sanity
  notes TEXT, -- Комментарий клиента к фото
  created_at TIMESTAMP DEFAULT NOW(),
  UNIQUE(user_id, sanity_photo_id) -- Защита от дублей
);
```

Таблица orders

SQL

```
CREATE TABLE orders (
 id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
 user_id UUID REFERENCES users(id),
 stripe_session_id VARCHAR(255) UNIQUE,
 amount DECIMAL(10, 2) NOT NULL,
 currency VARCHAR(3) DEFAULT 'USD',
 status VARCHAR(50) DEFAULT 'pending', -- 'paid', 'failed', 'delivered'
 items JSONB NOT NULL, -- Snapshot купленных товаров (чтобы история не менялась при смене цен)
 created_at TIMESTAMP DEFAULT NOW()
);
```

---

6. E-commerce и Безопасность

6.1. Реализация безопасного доступа (Secure Access)

Для защиты приватных галерей используется комбинированный подход.

1. **Middleware Protection:** В Next.js создается middleware.ts, который перехватывает запросы к путям /client/*. Он проверяет наличие валидной сессии (JWT) или специального токена доступа в URL (Magic Link). Если проверка не пройдена, происходит редирект на страницу входа.[36]
2. **Signed URLs для загрузок:** Прямые ссылки на файлы в облаке (например, на S3 или Cloudinary) никогда не должны быть публичными. Когда клиент нажимает "Скачать", сервер (через Server Action) генерирует подписанную ссылку с временем жизни (TTL) 15-60 минут. Это гарантирует, что ссылка, пересланная третьему лицу, перестанет работать в ближайшее время.[18]

6.2. Интеграция Stripe Checkout

Stripe Checkout предоставляет готовый, оптимизированный для конверсии интерфейс оплаты, который снимает с разработчика бремя соответствия требованиям PCI-DSS.
Процесс покупки:

1. Пользователь собирает корзину (React Context / LocalStorage).
2. При нажатии "Оплатить" данные корзины отправляются на API Route /api/checkout.
3. Сервер создает сессию Stripe (stripe.checkout.sessions.create), передавая массив товаров (line_items), success_url и cancel_url.
4. Важно: в metadata сессии Stripe передается userId и список photoIds, чтобы после оплаты можно было идентифицировать, что именно куплено.
5. После оплаты Stripe отправляет событие checkout.session.completed на вебхук нашего сервера.
6. Обработчик вебхука проверяет подпись события, находит пользователя и отправляет ему email с вечными ссылками на скачивание купленных фото.[28]

---

7. Документация для AI-Генерации (The "Engine" Prompt)

Этот раздел содержит ключевой актив отчета — системную документацию, предназначенную для "скармливания" современной большой языковой модели (LLM), такой как Claude 3.5 Sonnet или GPT-4o. Эта документация позволит сгенерировать до 90% кодовой базы проекта.

7.1. Концепция .cursorrules

Файл .cursorrules (или системный промпт) задает контекст и ограничения для AI-агента. Он предотвращает галлюцинации, навязывает архитектурные паттерны и гарантирует соблюдение лучших практик.[40]

7.2. Системный Промпт (Мастер-инструкция)

Ниже приведен текст инструкции, которую необходимо использовать при старте разработки.

SYSTEM PROMPT BEGIN

You are an expert Principal Software Architect specializing in **Next.js 15**, **Sanity.io**, and **Digital Asset Management**. You are tasked with building a high-performance photography portfolio and proofing platform.

CORE TECHNOLOGY STACK:

- **Framework:** Next.js 15 (App Router) with React 19.
- **Language:** TypeScript (Strict Mode).
- **Styling:** Tailwind CSS v4, Shadcn/UI, Framer Motion (for complex animations).
- **CMS:** Sanity.io (v3). Use next-sanity for fetching.
- **Database:** PostgreSQL (via Neon or Supabase) managed by **Drizzle ORM**.
- **Auth:** Clerk or NextAuth (v5).
- **Media:** Cloudinary (for auto-watermarking and transformations).

ARCHITECTURAL GUIDELINES:

1. **Server-First Mental Model:**
    - Default to React Server Components (RSC). Use use client ONLY for leaf components requiring interactivity (buttons, inputs, frame-motion gestures).
    - Use **Server Actions** for all mutations (form submissions, adding to favorites, checkout initiation). DO NOT create REST API routes unless absolutely necessary (e.g., for external Webhooks).
2. **Image Optimization Strategy:**
    - ALWAYS use next/image.
    - For Sanity images, use the @sanity/image-url builder.
    - Implement a robust blurDataURL strategy using the LQIP (Low Quality Image Placeholder) metadata from Sanity.
    - Enforce explicit sizes prop to ensure responsive image delivery.
    - **Watermarking:** Create a helper utility getWatermarkedUrl(sanityId) that constructs a Cloudinary URL with an overlay transformation layer applied on-the-fly.
3. **Data Fetching & Caching:**
    - Use sanityFetch utility with next: { tags: ['collection'] } for Revalidation.
    - Implement GROQ queries that are precise. Do not fetch *. Fetch { _id, title, slug, "aspectRatio": mainImage.asset->metadata.dimensions.aspectRatio }.
    - For Bento Grids: Your GROQ query should pre-calculate layout hints or fetch

dimensions so the CSS Grid can be generated server-side to avoid Layout Shift.

4. **Directory Structure (Feature-Sliced approach adapted for Next.js):**
    - /app: Routes.
    - /components/ui: Shadcn primitives.
    - /features/gallery: Components specific to gallery logic (Masonry, Lightbox).
    - /features/proofing: Logic for favorites, client authentication.
    - /lib/sanity: Schemas, client config.
    - /lib/db: Drizzle schema and connection.
5. **Coding Rules:**
    - **Type Safety:** Generate TypeScript types from Sanity Schemas automatically. Do not use any.
    - **Accessibility:** Ensure all images have alt text (fallback to empty string for decorative). Ensure Lightbox is keyboard navigable (Esc to close, Arrows to navigate).
    - **Error Handling:** Use error.tsx boundaries in Next.js. Wrap Server Actions in try/catch and return structured error objects { success: boolean, error?: string }.

SPECIFIC MODULE INSTRUCTIONS:

Module A: The "Bento Grid" Component
Create a BentoGrid component that accepts a list of items. Use CSS Grid.

- Logic: The first item spans 2 columns and 2 rows. Every 5th item spans 2 columns.
- Use :nth-child selectors or pass a className prop calculated during the map loop.
- Wrap images in a motion.div that triggers a fade-in animation when the element enters the viewport.

Module B: Client Proofing Favorites

- Schema: Users <-> Favorites <-> Photos.
- Action: toggleFavorite(photoId, projectId).
- Optimistic UI: Use useOptimistic hook to instantly toggle the heart icon before the server response comes back.

Module C: Stripe Integration

- Create a checkout session that accepts mode: 'payment'.
- Important: Pass metadata: { userId, projectSlug } to the session.
- Handle checkout.session.completed webhook to update the orders table in Drizzle and send an email via Resend/SendGrid.

SYSTEM PROMPT END

---

8. Заключение и Перспективы

Представленная архитектура определяет "золотой стандарт" для веб-присутствия фотографа в 2025 году. Она решает ключевое противоречие индустрии: необходимость демонстрации высококачественного (и "тяжелого") визуального контента при сохранении мгновенной скорости загрузки сайта.

Использование Bento-сеток и эстетики тактильного максимализма позволяет фотографу выделиться на фоне конкурентов, использующих шаблонные решения. Интеграция глубокой бизнес-логики (пруфинг, многовариантная продажа) превращает сайт из пассива в актив, генерирующий доход.

Критически важным является предложенный подход к разработке через AI. Используя детализированную документацию и архитектурные ограничения (Constraints), описанные в разделе 7, разработчик может делегировать нейросети написание рутинного кода (boilerplate), сосредоточившись на тонкой настройке UX и визуальной полировке. Это снижает время выхода на рынок (Time-to-Market) с месяцев до недель, делая кастомную разработку экономически рентабельной для индивидуальных профессионалов.

В будущем данная система может быть расширена за счет внедрения AI-поиска по содержимому изображений (на базе CLIP-моделей) и персонализированных предложений для клиентов на основе их истории просмотров, что станет следующим логическим шагом в эволюции цифровых платформ для творцов.

Works cited

1. The 11 biggest web design trends of 2025 - Wix.com, accessed November 21, 2025, https://www.wix.com/blog/web-design-trends
2. The 10 Most Inspirational UX Design Portfolio Examples in 2025 | IxDF, accessed November 21, 2025, https://www.interaction-design.org/literature/article/the-10-most-inspirational-ux-design-portfolio-examples
3. Build a Document Generation System by Using AI - Azure Architecture Center, accessed November 21, 2025, https://learn.microsoft.com/en-us/azure/architecture/ai-ml/idea/generate-documents-from-your-data
4. AI Code Generation Explained: A Developer's Guide - GitLab, accessed November 21, 2025, https://about.gitlab.com/topics/devops/ai-code-generation-guide/
5. 25 Web Design Trends to Watch in 2025 - DEV Community, accessed November 21, 2025, https://dev.to/watzon/25-web-design-trends-to-watch-in-2025-e83
6. Bento Grid | React Components & Templates - Magic UI, accessed November 21,

2025, https://magicui.design/docs/components/bento-grid
7. Tailwind CSS Bento Grids - Official Tailwind UI Components, accessed November 21, 2025, https://tailwindcss.com/plus/ui-blocks/marketing/sections/bento-grids
8. Bento Grids, accessed November 21, 2025, https://bentogrids.com/
9. 25 Top Web Design Trends 2025 | TheeDigital, accessed November 21, 2025, https://www.theedigital.com/blog/web-design-trends
10. Online marketing for photographers in 2025 - picu, accessed November 21, 2025, https://picu.io/blog/online-marketing-for-photographers-in-2025/
11. Photography Workflow: The Ultimate Action Plan from Pre to Post - Imagely, accessed November 21, 2025, https://www.imagely.com/photography-workflow/
12. The Ultimate Guide to Creating a Workflow for Photographers - Iris Works, accessed November 21, 2025, https://www.iris-works.com/the-ultimate-guide-to-creating-a-workflow-for-photographers
13. 10 Must-Have Features of Great Websites for Photographers in 2025 - EuroDNS, accessed November 21, 2025, https://www.eurodns.com/blog/10-must-have-features-of-great-websites-for-photographers
14. Favorite Proofing Workflow Guide - Shootstack, accessed November 21, 2025, https://shootstack.io/docs/favorite-proofing-workflow
15. Automatically Watermark an Image Gallery on Upload - Cloudinary, accessed November 21, 2025, https://cloudinary.com/blog/guest_post/automatically-watermark-image-gallery-on-upload
16. Add Watermark to Image, Add Text Overlay to Image, and More - Cloudinary, accessed November 21, 2025, https://cloudinary.com/blog/adding_watermarks_credits_badges_and_text_overlays_to_images
17. commercelayer/sanity-commerce: Sanity Commerce by Commerce Layer - GitHub, accessed November 21, 2025, https://github.com/commercelayer/sanity-commerce
18. Signed URLs vs Signed Cookies for Secure File Access in Web Apps - CloudThat, accessed November 21, 2025, https://www.cloudthat.com/resources/blog/signed-urls-vs-signed-cookies-for-secure-file-access-in-web-apps
19. Delivering digital goods using Next.js and Stripe Checkout? - Reddit, accessed November 21, 2025, https://www.reddit.com/r/stripe/comments/1c1xljr/delivering_digital_goods_using_nextjs_and_stripe/
20. 10 Best CMS Platforms to Build With in 2025 (Compared) - Strapi, accessed November 21, 2025, https://strapi.io/blog/best-cms-2025
21. Best Headless CMS Platforms: Top Picks & Comparison 2025 | Kontent.ai, accessed November 21, 2025, https://kontent.ai/blog/best-headless-cms-complete-buyers-guide/
22. Image Component | Next.js, accessed November 21, 2025,

https://nextjs.org/docs/app/api-reference/components/image
23. How To Optimize Images in Next.js with Next/Image Component | Blazity, accessed November 21, 2025, https://blazity.com/blog/next-js-image-component
24. Portfolio - made with Sanity.io, accessed November 21, 2025, https://www.sanity.io/projects/saragrayenportfolio
25. Collected – Portfolio template - Sanity, accessed November 21, 2025, https://www.sanity.io/templates/collected-portfolio-template
26. Top 10 Database Schema Design Best Practices - Bytebase, accessed November 21, 2025, https://www.bytebase.com/blog/top-database-schema-design-best-practices/
27. Enhancing Product Images With Cloudinary AI in Your Next.js Store, accessed November 21, 2025, https://cloudinary.com/blog/product-images-cloudinary-ai-next-js-store
28. Integrating Stripe Checkout with Next.js Is Easier Than You Think | HackerNoon, accessed November 21, 2025, https://hackernoon.com/integrating-stripe-checkout-with-nextjs-is-easier-than-you-think
29. How to Think About Security in Next.js, accessed November 21, 2025, https://nextjs.org/blog/security-nextjs-server-components-actions
30. Guides: Data Security - Next.js, accessed November 21, 2025, https://nextjs.org/docs/app/guides/data-security
31. Using Sanity CMS to model a product catalog - Commerce Layer, accessed November 21, 2025, https://commercelayer.io/blog/using-sanity-cms-to-model-a-product-catalog-for-the-commerce-layer-demo-store
32. The engineer's guide to content operations [E-commerce edition] - Sanity, accessed November 21, 2025, https://www.sanity.io/blog/e-commerce-content-operations-guide
33. Query Cheat Sheet - GROQ | Sanity Docs, accessed November 21, 2025, https://www.sanity.io/docs/content-lake/query-cheat-sheet
34. How Queries Work – GROQ | Sanity Docs, accessed November 21, 2025, https://www.sanity.io/docs/content-lake/how-queries-work
35. Best Database Design for Favorites - FlutterFlow Community, accessed November 21, 2025, https://community.flutterflow.io/database-and-apis/post/best-database-design-for-favorites-ZV8WjFZMuMNRG5w
36. How to Configure Secure Routes Using Next.js Middleware— the right way!, accessed November 21, 2025, https://trillionclues.medium.com/how-to-configure-secure-routes-using-next-js-middleware-the-best-way-320fe1f7236d
37. How to Secure Routes in Next.js 13 – Client-Side, Server-Side, and Middleware-Based Protection - freeCodeCamp, accessed November 21, 2025, https://www.freecodecamp.org/news/secure-routes-in-next-js/
38. Is this implementation of signed URLs reasonably secure? - Stack Overflow, accessed November 21, 2025,

https://stackoverflow.com/questions/61203451/is-this-implementation-of-signed-urls-reasonably-secure
39. Stripe Checkout and Webhook in a Next.js 15 (2025) | by John Gragson | Medium, accessed November 21, 2025, https://medium.com/@gragson.john/stripe-checkout-and-webhook-in-a-next-js-15-2025-925d7529855e
40. How Cursor project rules can improve Next.js app development - LogRocket Blog, accessed November 21, 2025, https://blog.logrocket.com/cursor-project-rules-improve-next-js-app-development/
41. Rules | Cursor Docs, accessed November 21, 2025, https://cursor.com/docs/context/rules