Christopher Wilt

Systems & Full-Stack Software Engineer

Mifflinburg, PA — **remote, or hybrid in central PA** | 570-541-9807 | ctwilt23@gmail.com
linkedin.com/in/christopher-t-wilt | christopherwilt.github.io | github.com/ChristopherWilt

PROFESSIONAL SUMMARY

Systems and full-stack engineer who ships production software end to end. Designed, built and launched **The Veil**, a live multi-surface SaaS platform (iOS, Android, web and a vendor CRM) on a single PostgreSQL instance where row-level security — not the client — enforces access for five distinct user roles. Equally comfortable below the abstraction line: bare-metal Vulkan 1.3 rendering pipelines, thread-safe C++ schedulers, and server-authoritative multiplayer networking in Unreal Engine 5. Focused on secure architecture, verifiable permission models and data consistency that holds under load.

TECHNICAL SKILLS

Languages: C++, TypeScript/JavaScript, SQL (PostgreSQL, T-SQL, MySQL), Python, C#, Java

Full-Stack: React, Next.js 15, React Native (Expo), Node.js, Deno, REST APIs, Tailwind, monorepo architecture

Data & Security: PostgreSQL row-level security, SECURITY DEFINER functions, schema normalization (3NF), query plan analysis & indexing, migration discipline, Stripe / Stripe Connect payment infrastructure

Cloud & DevOps: Terraform, Docker, GitHub Actions CI/CD, AWS, Azure (Key Vault, VNet, ACR), Vercel, Supabase, Nginx, Perforce (Helix Core), Git

Systems & Graphics: Vulkan 1.3, HLSL, PBR & IBL pipelines, ECS / data-oriented design, multithreading & concurrency, memory management, TCP/UDP sockets, Unreal Engine 5, Unity

EXPERIENCE

Founder & Lead Systems Architect — The Veil (theveilwedding.com) **April 2026 – Present**

Next.js 15 · React Native (Expo) · PostgreSQL 17 · Deno edge functions · Stripe Connect · Vercel

- **Shipped a three-surface product solo** — an iOS, Android and web client from one React Native codebase, plus a Next.js marketing site, vendor CRM and staff operations console — totaling ~300,000 lines across 550 ordered SQL migrations and 75 Deno edge functions.
- **Designed a derived permission model** in which two SQL predicates and seven module-scoped grants authorize five roles (host, secondary host, co-planner, day-of coordinator, guest) across ~27 shared tables, eliminating a roles table that could drift from the access it implied.
- **Closed a privilege-escalation path** by moving per-module delegation out of the UI and into RLS, where a collaborator granted one module could previously write every other through the API; verified with impersonated, rolled-back transactions judged on affected row count rather than absence of an error.
- **Isolated untrusted callers from shared tables entirely** — every guest action runs through a SECURITY DEFINER function scoped to the caller's claimed seat, and the partner-linking table carries no UPDATE policy at all.
- **Built payment infrastructure across five flows** on Stripe, Stripe Connect and RevenueCat; the client submits a plan name and never an amount, entitlement expiry is resolved server-side from authoritative data, and the webhook is the single idempotent writer keyed on checkout session ID against at-least-once delivery.
- **Engineered AI subsystems against real product constraints**: a curation engine separating explicit instruction from inferred taste across thirteen event-specific briefs, and a planning generator emitting dependency-aware tasks as date offsets, so changing the event date re-derives every deadline instead of invalidating the plan.
- **Built the marketplace cold-start engine** — an enquiry materializes an unclaimed storefront from Google Places data and triggers verified outreach; the system refuses to fabricate contact addresses and gates live sending behind an opt-in that fails closed.
- **Opened the platform to other software**: a key-scoped public REST API with signed webhooks, a Zapier app, QuickBooks Online accounting sync over OAuth, and an MCP server with OAuth 2.1 sign-in so a business can work its leads from Claude or ChatGPT.
- **Sell it myself, on tooling I built**: demos and walk-ins with venue owners, backed by paced Gmail outreach batches, a crawler that finds each business's published address, and self-planning call lists that drop anyone who opts out.
- **Tuned performance at the policy layer**: wrapped authorization lookups in scalar subqueries to stop per-row re-evaluation and added covering indexes to every foreign key the predicates filter on.

ENGINEERING PROJECTS

Multiplayer FPS Framework — Unreal Engine 5, C++ **Sept 2025 – Present**

- Architected a modular, **server-authoritative** framework with strict client-prediction and server-validation logic to manage real-time replication while awaiting authoritative responses.
- Engineered a **custom lag compensation system** for hit-scan weapons, rewinding server-side hitboxes to match client timestamps for fair competitive play.
- Built a hybrid, data-driven recoil solver decoupled from the weapon class via **UDataAssets**, supporting hot-swappable pattern-based and procedural modes plus a spring-damper ADS system.
- Integrated **Epic Online Services** for matchmaking tickets, lobby creation and peer-to-peer traversal, including a log writer that scrubs portal credentials before output.

Graphics & Game Engine Systems — C++, Vulkan 1.3, HLSL **June 2025 – Aug 2025**

- Developed a production-grade, bare-metal **Vulkan PBR rendering pipeline** from the ground up, implementing Cook-Torrance microfacet shading, image-based lighting from HDR cube maps, and a pre-computed BRDF lookup table.
- Built a data-driven **ECS level loader** on EnTT, replacing object-oriented game loops with cache-friendly component arrays and a flyweight instancing factory that shares pooled geometry across replicated entities.
- Managed the lifecycle of low-level graphics structures — pipelines, descriptor pools, uniform buffers and storage views — alongside a CPU-only software rasterizer with Z-buffering and perspective-correct UV mapping.

Cloud Security & Infrastructure — Nexus Platform (Terraform, Azure, Docker) **Jan 2026 – March 2026**

- Authored modular **Terraform** configurations provisioning an isolated Azure architecture: virtual network subnets, Linux Web Apps and an Azure Container Registry.
- Implemented **zero-trust backend isolation**, wrapping Azure SQL in virtual-network firewall rules that reject all inbound traffic originating outside the application subnet.
- Managed pipeline secrets natively through **Azure Key Vault**, generating, applying and rotating administrator credentials at runtime rather than hardcoding them, and shipped GitHub Actions workflows that build and push signed Nginx container images on every push to main.

Concurrency & Operating Systems Engineering — C++ **April 2025 – July 2025**

- Engineered thread-safe pipelines demonstrating shared-memory corruption hazards, applying strict synchronization primitives (**std::unique_lock**, condition variables) to override OS scheduling non-determinism.
- Replaced array-based allocation by tracking concurrent hardware contexts inside a single 32-bit unsigned integer, using **atomic bitwise operations** to throttle and dispatch execution threads.

Database Architecture & Security Analysis — SQL, Python **Feb 2025 – March 2025**

- Designed a **3NF relational schema** to parse and store raw web server logs, tracking user IPs, session durations and access patterns, with stored procedures and views auto-generating daily traffic reports.
- Wrote SQL detecting security anomalies — bot traffic and unauthorized access attempts — from user-agent strings and request frequency, and a Python regex utility that flags 401 responses during parsing, before database commitment.
- Optimized high-volume search queries with clustered and non-clustered indexes across a **30,000+ record dataset**, cutting complex-join execution time and reducing data redundancy by 60% while enforcing referential integrity through foreign key constraints and ACID transactions.

Technical Lead & Infrastructure Admin — Delivery Detonation (UE5 capstone team) **Jan 2026**

- Administered a remote **Perforce (Helix Core)** depot for a multi-developer team, defining branching strategy and manually resolving deep binary merge conflicts.
- Optimized network bandwidth with a customized replication graph, using raw UDP for high-frequency actor translation while reserving reliable RPC nodes for critical game state.

EDUCATION

Bachelor of Science in Game Development — Full Sail University, Winter Park, FL **Graduated April 2026**

Core focus: Systems Architecture & Software Engineering

- Coursework: Advanced Computer Networks (TCP/UDP sockets, blocking and non-blocking execution), Operating Systems & Systems Programming (race conditions, process scheduling, raw file I/O, binary serialization), Advanced Database Management, Linux Systems Administration, Computer Graphics.