EXHAUSTIVE CURRICULUM VITAE | CURRENT TO 29 JULY 2026

Llewellyn van der Merwe

Director | Senior Architectural Engineer of Intelligent Software and Systems

Senior Software & Systems Architect | PHP/Joomla Platform Engineer | Linux/DevOps & Applied AI

ENGINEERING
https://github.com/Llewellynvdm

PROFILE
https://vdm.io/llewellyn

CONTACT
https://t.me/llewellynvdm

WINDHOEK / NAMIBIA

Professional Profile

Llewellyn van der Merwe is a senior, hands-on software and systems architect whose work combines direct implementation with company leadership, client responsibility, infrastructure, operations, release ownership and long-term technical stewardship. He has worked in software engineering since 2008, in Linux systems administration since 2010, and as Founder and Director of Vast Development Method since 2013.

His strongest technical centre is PHP and Joomla platform engineering, but the defining feature of his work is broader: he takes responsibility for the complete engineering lifecycle. That responsibility regularly spans discovery, domain modelling, architecture, implementation, data, integration, testing, packaging, deployment, infrastructure, security, release, maintenance and technical leadership.

He builds platforms, generators, reusable frameworks, data systems, deployment tools and engineering ecosystems rather than limiting his work to individual end-user features. Joomla Component Builder is the clearest long-term expression of that approach: a compiler-like development platform that models and generates complete Joomla extensions across multiple CMS generations. The surrounding JoomEngine, OctoLeo and GetBible programmes show the same pattern in registries, reusable services, deterministic build pipelines, container images, APIs, native integrations, public-service applications and operational automation.

The chronology is professionally significant. Much of the public source estate was conceived, hand-coded and placed into long-term support before modern generative coding systems. Dated repository histories show original implementation and continued maintenance across changing Joomla, PHP, Linux, container, browser and release environments.

Current work extends this mature engineering foundation into Model Context Protocol systems, bounded agent actions, hardened agent environments and local-model infrastructure. These systems apply AI to established architecture, security and automation practice; they do not treat AI as a replacement for engineering discipline.

Llewellyn is also a teacher, lecturer, mentor and developer coach. He deliberately transfers architectural thinking, design patterns, Linux and infrastructure practice, and responsible AI-assisted engineering to interns and developing engineers. This combination of implementation depth, operational judgment, leadership and knowledge transfer is especially valuable where a senior engineer must understand and improve the whole production chain.

Professional Scope at a Glance

- **Software engineering:** 2008–present
- **Systems administration and production operations:** 2010–present
- **Vast Development Method:** 2013–present
- **Current role:** Founder, Director, Senior Architect and hands-on engineer
- **Primary technical centre:** PHP, Joomla architecture, developer tooling and code generation
- **Complementary engineering depth:** C/C++, Python, Bash, JavaScript/TypeScript, HTML/CSS, SQL, APIs, Docker, Linux, networking, infrastructure, security, local inference, MCP and agent workflows
- **Public engineering record:** 109 canonical, attributable, non-fork repository lineages as reconciled through 29 July 2026
- **Primary company engineering estate:** git.vdm.dev, containing a broader body of public, private, internal, client and mirrored work than the selective public GitHub record

The public repositories provide the detailed, shareable evidence of architecture, implementation and maintenance. They are a selective lower bound rather than the complete professional estate. Vast Development Method has also delivered substantial large-scale proprietary and institutional work under contractual confidentiality and non-disclosure obligations. Those responsibilities are described at a safe professional level; client repositories, system names, internal architecture, infrastructure, data and workflows are not itemised.

Professional Value

Architecture Grounded in Implementation

Llewellyn works at architecture level while remaining directly involved in code, data structures, integration boundaries, test design, packaging, deployment and failure recovery. He decomposes complex systems into stable contracts and reusable services, then carries those decisions through implementation and maintenance. His public work includes component-generation systems, semantic action layers, immutable domain objects, cache and integrity controls, cross-platform deployment tools, native ABI boundaries and release automation.

Platforms and Developer Tools

Much of his work enables other developers and systems:

- Joomla Component Builder turns domain and interface models into installable Joomla applications.
- JoomEngine registries distribute reusable classes, Joomla namespace mappings, package blueprints, field definitions and integration services.
- OctoLeo tools automate deployment, packaging, synchronisation and image publication.
- GetBible libraries and pipelines expose stable scripture reference, search, extraction and application contracts.
- Joomla MCP presents bounded semantic administration actions to AI and agent clients.

This work requires durable abstractions, compatibility management, schema and query modelling, deterministic generation, documented extension points and disciplined release engineering.

Production and Operational Judgment

Continuous systems-administration responsibility since 2010 informs Llewellyn's software decisions. His scope includes Linux servers, web and database services, SSH, scheduled automation, backups, access control, networking, firewall administration, OPNsense, containerised services, self-hosted systems and operational continuity. Current infrastructure work also includes Gitea, local inference services, Incus-based agent environments and Open WebUI integration.

Because he operates what he builds, architecture decisions account for upgrade paths, capacity, permissions, observability, recovery, storage, network boundaries and long-term maintainability.

Long-Term Stewardship

Llewellyn has maintained major systems across Joomla and PHP generations, evolving public platforms while continuing commercial, institutional and community responsibilities. His engineering record is characterised by compatibility work, migration planning, documentation, reproducible releases and sustained maintenance rather than short-lived demonstrations.

Technical Leadership and Knowledge Transfer

His leadership combines company direction, architectural responsibility and direct engineering. He reviews work, explains system decomposition, guides developers toward stronger design patterns, supervises interns and teaches infrastructure and operational practice. He has contributed to Joomla teams concerned with architecture, production, operations, automated testing, Docker, extensions, web infrastructure, language support and mentoring.

Technical Capability

Software and Systems Architecture

- Domain modelling, service boundaries and dependency management
- Modular, object-oriented and SOLID design
- Reusable frameworks, registries, plugins and extension points
- Compiler-like code-generation and packaging pipelines
- Immutable and lazy domain models
- Contract validation, schema evolution and compatibility strategy
- API, transport and integration architecture
- Security boundaries, permissions, auditability and guarded write operations
- Upgrade, release, maintenance and operational design

PHP and Joomla Platform Engineering

- Long-term Joomla extension architecture across Joomla 3, 4, 5 and 6
- Joomla Component Builder architecture, generation stages and reusable Powers
- Components, modules, plugins, templates, layouts, fields, services and scheduled tasks
- Joomla Web Services API and Joomla-native integration
- Composer packaging and dependency injection
- PHP application layers over native extensions
- Relational data modelling, queries, migrations, access control and multilingual systems
- Backward compatibility, generated delivery, packaging and upgrade paths

C and C++

- Certified C++ Programmer
- C++20 command-line and extraction systems
- C ABI design and consumption
- Zend/PHP extension integration
- Streamed native APIs and process-boundary elimination
- CMake, Autotools and architecture-aware packaging
- Pinned native dependencies, golden evidence and reproducible builds

Python

- Structured libraries and service clients
- Unicode-aware search and reference resolution
- Deterministic data-transformation pipelines
- API and cache integration
- Checksum and integrity validation
- Asynchronous public-service applications
- Packaging and PyPI distribution
- Contract testing and publication automation

Bash, Linux and Operational Automation

- Cross-platform command-line tooling
- Server provisioning and lifecycle automation
- Docker and Compose orchestration
- systemd service installation and management
- Backup, update, synchronisation and release utilities
- Hash and checksum verification
- Capacity and resource profiling
- Multi-device storage workflows
- Operator-focused diagnostics, recovery and uninstall paths

JavaScript, TypeScript and Browser Systems

- Browser interaction, responsive interfaces and AJAX
- Typed API clients and data models
- Telegram Mini App interaction
- React and TypeScript application surfaces
- Model Context Protocol implementation
- Accessible naming, reduced-motion behaviour and browser regression testing
- Supporting HTML, CSS, SVG and UI integration

Data, APIs and Integration

- Relational database design and SQL
- Static and dynamic JSON API design
- REST and Joomla Web Services
- NDJSON streaming contracts
- Integrity hashes, snapshots and immutable data objects
- Search, reference-resolution and caching layers
- External-service integrations
- Data normalisation, annotation and deterministic publication

Containers, Infrastructure and Delivery

- Dockerfile and image-matrix generation
- Official Joomla container maintenance
- Docker Compose deployments
- OCI image publication and multi-architecture releases
- Incus virtual-machine images and policy
- Provenance, manifests and software bills of materials
- Linux services, web servers, databases, networks, firewalls and backups
- Self-hosted Git and supporting operational services

Applied AI, MCP and Intelligent Systems

- Embeddable and self-hosted MCP servers
- Bounded semantic actions over administrative APIs
- Principal-bound grants, plans, idempotency, locks and audit events
- Authenticated HTTP and stdio transports
- Hardened agent virtual machines
- Local CPU/NVMe inference services
- OpenAI-compatible APIs and Open WebUI integration
- Multi-agent and tool-oriented workflows
- Responsible use of AI within source control, testing, review and operational boundaries

Quality, Security and Release Engineering

- Unit, integration, contract, browser and live-fixture validation
- PHPT coverage for native PHP extensions
- Deterministic outputs and golden test evidence
- CI/CD and release automation
- Dependency and hash verification
- Bounded inputs, concurrency control and lock management
- Permission-aware and auditable operations
- Reproducible packaging and versioned delivery
- Explicit readiness gates rather than unqualified production claims

Professional Chronology

Vast Development Method | 2013–present

Founder, Director, Senior Architect and Hands-on Engineer

Vast Development Method is the central context for Llewellyn's commercial, institutional, product and open-source work. He established the company in 2013 and remains responsible for both company direction and practical engineering delivery.

His role includes:

- defining technical and product direction;
- designing software, data, integration and infrastructure architecture;
- implementing and reviewing production code;
- leading client and institutional delivery;
- maintaining public products and open-source ecosystems;
- operating Linux, network and application infrastructure;
- owning release, upgrade and maintenance processes;
- managing continuity across long-running programmes;
- supervising interns and developing engineers;
- providing technical instruction and architectural coaching;
- integrating modern AI and agent tooling into established systems.

Vast Development Method supports long-running programmes as well as shorter and medium-duration consulting engagements across different organisations and sectors. GetBible, OctoLeo, JoomEngine and related public ecosystems are company-led, community, product or multi-stakeholder engineering programmes; they are not separate employers.

Company Engineering Estate

git.vdm.dev is the company-owned Git forge and primary engineering estate. It contains a substantial body of public, private, internal, client and mirrored engineering work, including large-scale proprietary engagements that cannot be itemised publicly. Public GitHub organisations—including JoomEngine, GetBible, OctoLeo, Vast Development Method, True Christian and Llewellyn-vdm—provide a selective public view. Protecting client repositories, system names, internal structures, infrastructure and operational methods is part of the professional trust attached to the company's long-running technical responsibility.

Systems Administration and Production Operations | 2010–present

Llewellyn's systems work grew from operating web and database hosting into broader responsibility for Linux services, deployment, security, backups, networking and self-hosted infrastructure. He has maintained services used by company, client and public projects, built automation to make operations repeatable, and used production experience to improve application architecture.

Demonstrated areas include:

- Linux server and service administration;
- SSH and access management;
- web and relational database hosting;
- scheduled jobs and automation;
- backups and recovery planning;
- domain and application operations;
- networking, firewalls and OPNsense;
- Dockerised and self-hosted services;
- Gitea-based engineering operations;
- Incus-based development and agent environments;
- systemd-managed local inference services.

Software Engineering | 2008–present

Llewellyn began building websites in 2008 and progressed from interface and web-delivery work into PHP applications, Joomla extensions, relational data systems, platform architecture, code generation, container delivery, infrastructure automation and intelligent systems.

This progression established a broad engineering foundation:

1. web presentation, client delivery and PHP application work;
2. Joomla extension development and reusable application architecture;
3. relational database design and domain-specific systems;
4. Linux hosting, deployment and operational responsibility;
5. developer tooling, generation, packaging and release engineering;
6. public APIs, data pipelines and multi-platform applications;
7. native C/C++ integration and modern PHP application layers;
8. MCP, agent environments and local-model infrastructure.

Commercial Foundation | Powercell Namibia | 1996–1999

Before entering software engineering, Llewellyn worked in sales for a family business, travelling extensively across Namibia and carrying significant customer and budget responsibility. This early experience developed commercial awareness, independent planning, communication and accountability—capabilities that later supported client leadership and company direction.

Mission, Church and Public Teaching | 1998–present

Following his Christian conversion in 1998, Llewellyn entered full-time mission work in 2000. The Windhoek church was established in 2004, and he began serving as an elder in 2026. Mission, church leadership, preaching, long-form teaching and public communication remain a sustained part of his life and leadership foundation.

Engineering Ecosystems and Case Studies

1. Joomla Component Builder and JoomEngine

Joomla Component Builder

Repository: joomengine/Joomla-Component-Builder
Role: Founder, principal architect, lead developer and long-term maintainer
Technologies: PHP, JavaScript, CSS, HTML, SQL, Joomla

Joomla Component Builder (JCB) is a professional Joomla extension-development engine and compiler-like build platform. It enables developers to model domain structures, administration and site interfaces, permissions, fields, database schemas, language material, reusable code and packaging rules, then compile installable Joomla extensions.

Llewellyn's architectural responsibility includes the platform model, generation stages, service boundaries, reusable Powers, version-aware behaviour, packaging, Git-backed distribution and long-term compatibility. The system has evolved across Joomla 3 through Joomla 6 while continuing to support production application delivery.

JCB demonstrates:

- complex domain and schema modelling;
- generator and compiler-style architecture;
- repeatable creation of components, modules and plugins;
- reusable service and class injection;
- Joomla-version and PHP-version compatibility;
- install, update and migration packaging;
- access-control and multilingual application generation;
- documentation and ecosystem support;
- sustained maintenance and release ownership.

The dated public history reviewed in July 2026 contained 959 attributable commits and approximately 7.75 million recorded changed lines. That change volume includes generated delivered output and is not presented as manually typed code or an effort estimate. The professional significance is the duration, architecture and ecosystem enabled by the platform.

JoomEngine Reusable Registries and Distribution

The JoomEngine organisation is the public distribution and collaboration surface for a much wider JCB engineering system. Its 22 substantive public repositories resolve to 21 JoomEngine-owned canonical lineages plus the official JoomEngine Docker build lineage maintained under OctoLeo; the `joomengine/docker` fork is not counted again.

The ecosystem's generated indices, reviewed on 29 July 2026, expose:

- 180 core Super Powers across 46 namespaces;
- 510 version-aware Joomla class mappings;
- 49 field-type blueprints;
- 353 reusable UI snippets;
- seven component-package blueprints;
- 12 GitHub, Gitea and Codeberg repository-target definitions;
- 114 reusable Gitea powers across 20 namespaces;
- a JCB 6.1.6 installable bundle containing 18 extensions;
- a Service Directory 6.0.3 bundle containing two extensions.

These figures describe indexed definitions, mappings and packaged outputs. They are evidence of a model-driven reuse and delivery architecture, not claims that every generated item was manually typed or that every upstream dependency was authored by Llewellyn.

The complete JoomEngine public system is organised into five functional layers:

1. **Core platforms and applications**

- Joomla Component Builder — the primary compiler-like development platform, with public evidence from January 2016 to May 2026 and a manifest recording development from April 2015.
- Joomla MCP — bounded Joomla administration for MCP and agent systems.
- Joomla Service Directory — a complete Joomla 6 application whose repository credits Lemuel van der Merwe as author and Llewellyn as developer, builder and release contributor.

2. **Reusable Power and service registries**
   - super-powers, gitea, search, openai, phpseclib, fof, minify and psr provide reusable interfaces, services and integration classes through stable JCB identifiers.
   - These repositories demonstrate integration architecture and maintained compatibility surfaces. They do not imply authorship of OpenAI, phpseclib, FOF, PSR or other upstream technologies.
3. **Version-aware models and reusable definitions**
   - joomla-powers resolves Joomla classes and namespaces across CMS generations.
   - joomla-fieldtypes distributes field blueprints.
   - snippets supplies reusable layout, view and template material.
   - packages provides machine-readable component blueprints.
   - repoindex defines remote repository targets used by JCB INIT, RESET and PUSH workflows.
4. **Packaging and release distribution**
   - pkg-component-builder publishes the installable JCB release bundle; release 6.1.6 packages 18 extensions for Joomla 6.
   - pkg-servicedirectory publishes the Service Directory bundle.
   - octoleo/joomengine supplies the authoritative deterministic container-build and publication engine described in the OctoLeo section.
5. **Documentation, external integration and browser delivery**
   - jcb-documentation has an eleven-calendar-year public history and covers architecture, setup, first builds, data modelling, extensibility, packaging, CLI operation and maintenance.
   - jcb-external supports JCB's external-code compilation protocol; contained upstream code is not presented as Llewellyn's authorship.
   - uikit provides a maintained, collision-aware upload and delete integration; it is not the UIkit framework itself.

The organisation therefore documents a complete engineering model: platform source, reusable service registries, version-aware metadata, application blueprints, documentation, packaging, distribution and container delivery. This is materially broader than one flagship repository and demonstrates how Llewellyn turns architecture into an ecosystem that other projects can consume and maintain.

2. Joomla MCP and AI-Assisted Joomla Tooling

Joomla MCP

Repository: joomengine/joomla-mcp
Role: Project architect, principal implementer and maintainer
Technologies: TypeScript, PHP, JavaScript, JSON, Bash, Docker, YAML/CI

Joomla MCP is an embeddable library and self-hosted Model Context Protocol implementation for Joomla administration. It exposes bounded, auditable semantic actions instead of giving agents unrestricted access to raw administrative interfaces.

Its architecture includes:

- Joomla Web Services API and Joomla-native companion integration paths;
- a source-backed catalogue of all 236 Joomla 6.1 core Web Services route templates;
- semantic CRUD coverage across 36 core resource bases;
- guarded writes and permission-aware operations;
- principal-bound grants, plans, idempotency and locks;
- structured audit events and post-action verification;
- stdio and authenticated Streamable HTTP transports;
- typed packaging and public npm distribution;
- OCI, Compose and systemd deployment foundations;
- live validation and explicit production-certification gates.

The complete default-branch history contained 83 commits at the 29 July 2026 review, and the public package had reached `@joomengine/joomla-mcp` 0.7.0. The project does not claim that every semantic action is already production-certified; explicit gates preserve that distinction.

Joomla MCP is a direct bridge between Llewellyn's mature Joomla architecture and modern AI/agent systems. It demonstrates how agent tooling can be added to established platforms with typed contracts, permission boundaries, auditability and operational control.

Controlled Integration into Agent Environments

Joomla MCP is also integrated in the PHP and full images of the OpenCode Platform. This places the MCP server inside a version-controlled, reproducible agent environment rather than treating it as an ad hoc local utility.

3. OctoLeo Engineering and Automation

OctoLeo is a coherent public systems-engineering estate, not a collection of incidental scripts. Ten authored Octo* repositories cover platform operation, packaging, dependency resolution, release integrity, synchronisation, network-edge control, mail diagnostics and service lifecycle coordination. Their reviewed working trees contain approximately 14,600 lines of Bash, Docker, HTML and workflow implementation, excluding README documentation. The engineering value lies in how those tools encode operational procedures, constraints, recovery paths and publication rules into maintainable software.

OctoJoom — Containerised Joomla Operations

Repository: octoleo/octojoom
Role: Creator, principal Bash engineer and long-term maintainer
Dated public evidence: 3 May 2021 to 26 February 2026
Technologies: Bash, Docker, Docker Compose, Traefik, Portainer, OpenSSH, ACME, Cloudflare DNS

OctoJoom is the flagship OctoLeo platform utility. Its reviewed implementation contains approximately 6,573 lines of Bash across 175 functions and 71 authored commits. It installs and manages Docker and Compose across Linux, macOS and Windows, then generates and operates multi-user Joomla and OpenSSH environments.

The system includes:

- Traefik and Portainer integration;
- ACME and Cloudflare DNS certificate automation;
- network and security-header generation;
- host-file integration and operator-facing configuration;
- persistent-volume cloning, reset and permissions repair;
- bulk environment creation;
- update, repair and uninstall lifecycle operations;
- SSH-based push/pull migration and backup flows.

This is advanced shell-based platform engineering: substantial state management, cross-platform compatibility, service orchestration, network and certificate concerns, operator ergonomics and safe lifecycle handling in a single maintained system.

JoomEngine — Deterministic Container Build and Publication

Repository: octoleo/joomengine
Role: Creator, lead architect and maintainer
Latest reviewed activity: 9 July 2026
Technologies: Bash, Docker, GitHub Actions, jq, XMLStarlet, awk

JoomEngine supplies the authoritative build system for official Joomla Component Builder images. A 703-line Bash build engine, a 456-line strict and idempotent runtime entrypoint and a focused publication workflow:

- discover authoritative JCB releases and SHA-512 values;
- expand Joomla, PHP and Apache/FPM/FPM-Alpine build matrices;
- generate deterministic Dockerfiles, contexts and entrypoints;
- track processed combinations and avoid duplicate work;
- calculate stable, prerelease and global-latest tag leadership;
- emit machine-readable NDJSON manifests;
- check registry state, build images and publish them.

The reviewed repository contains 26 JCB version families and 156 manifest-defined image contexts. Generated image trees are release artefacts, not represented as manually authored source volume. The authored build logic demonstrates reproducibility, supply-chain verification, version policy and multi-variant publication.

OctoPower, OctoJpack, OctoZipo and OctoShoom — Release Engineering Chain

- **OctoPower** — approximately 2,091 lines and 66 functions. Traverses JCB Power relationships, resolves child and linked dependencies, validates identifiers, derives namespaces and imports, calculates Composer autoloading and requirements, calls Gitea APIs, generates package metadata and publishes repositories. Public evidence runs from June 2023 to March 2025.
- **OctoJpack** — approximately 1,588 lines and 40 functions. Uses JSON and environment configuration to retrieve tagged or keyed artefacts, generate installation XML, language, README and licence material, and create or update repositories. Public evidence runs from March 2022 to September 2025.
- **OctoZipo** — approximately 1,346 lines and 22 functions. Converts ZIP artefacts into versioned Git repositories, parses package metadata, maps names and versions, manages GPG/SSH concerns, and supports dry-run and push-create operation. Public evidence runs from June 2022 to December 2025.
- **OctoShoom** — processes repository definitions, downloads artefacts referenced by Joomla update XML, calculates SHA-512 values, inserts or replaces integrity elements, and commits controlled metadata updates. Public evidence runs from January to September 2025.

Together these repositories show an end-to-end release system: source and dependency discovery, package generation, artefact ingestion, repository creation, versioning, integrity metadata and publication.

OctoMailTest — Mail, DNS and TLS Diagnostics

Repository: octoleo/octomailtest
Latest reviewed activity: 31 January 2026
Technologies: Bash, Docker, OpenSSL, DNS, SMTP, IMAP, ManageSieve

OctoMailTest provides Docker and standalone operational diagnostics across MX, SPF, DKIM, DMARC, BIMI, MTA-STS, TLS-RPT, SMTP ports 25/465/587, IMAPS 993, ManageSieve 4190, certificates, protocol and cipher posture, authentication, reverse DNS, block lists and SMTP capability discovery. Its approximately 1,187 lines of operational source support quiet CI operation, JSON output and file logging.

The project is strong public evidence of practical network protocols, certificate analysis, mail deliverability, observability and operator-oriented Bash design.

OctoSync — Cross-Repository Synchronisation

OctoSync selectively transfers files and directories between repositories, clones source, target and fork branches, rebases upstream changes, merges directly or creates pull requests, and supports configuration, testing and dry-run modes. Its 874-line, 18-function public implementation records authored work from 2020 to 2021 and remains part of the historical foundation of the later release toolchain.

OctoFlare, Octodinator and OctoLanding — Operational Safety

- **OctoFlare** uses the Cloudflare API to resolve zones and enable, disable or inspect security posture through a token-scoped CLI with quiet and debug modes.
- **Octodinator** uses shared lock state to run initialisation once when the first process starts and cleanup once when the final process exits. Its documented integration coordinates OctoFlare security transitions.
- **OctoLanding** publishes a deliberately minimal BusyBox-backed static endpoint for gateways, DMZs, reverse proxies and infrastructure fallback pages, avoiding an unnecessary executable server-side surface.

These tools demonstrate the ability to choose proportionate designs: automate a network-edge control, coordinate process lifecycle safely, or deliberately minimise an exposed service.

Git Identity and Documentation Integration

- **git-user** is a Bash and GitHub Action bootstrap for GPG-signed Git operations, SSH key validation, protected permissions, managed `known_hosts`, idempotent configuration and signed commit/tag verification. The reviewed implementation contains approximately 657 lines of Bash and an 859-line offline test suite with 89 passing tests using generated keys and mocked network discovery.
- **plantuml-gitea** is an adaptation and integration that renders PlantUML content in Gitea HTML blocks. Upstream components remain credited; Llewellyn's contribution is the maintained Gitea integration, not authorship of PlantUML.

All explicit GitHub forks and imported histories are excluded from the canonical achievement count. Each authored or adapted code lineage is presented once under its clearest current public custodian.

4. GetBible Scripture Platform

GetBible is a long-running scripture data, application and distribution ecosystem. Llewellyn's role spans architecture, data pipelines, APIs, Python services, PHP and Joomla applications, C/C++ extraction, native PHP integration, browser and mobile clients, Telegram delivery, packaging and operations.

GetBible Librarian

Repository: getbible/librarian
Role: Original author, principal architect and maintainer
Technologies: Python, PyPI packaging, Unicode search, HTTP APIs, caching, YAML/CI

Librarian is a reusable Python library for resolving scripture references, retrieving verses and searching GetBible translations. Its architecture includes:

- reference parsing and resolution;
- Unicode-aware search;
- separate Query and Search service integration;
- bounded caches;
- checksum verification;
- multi-worker operational behaviour;
- release gates and PyPI distribution.

The initial-to-reviewed-head history contained 33 commits through 20 July 2026. Older line totals were deliberately retired instead of mixing incompatible snapshots.

GetBible Robot and Telegram Mini App

Repository: getbible/robot
Role: Project architect, principal implementer and maintainer
Technologies: Python, TypeScript/JavaScript, HTML/CSS, Telegram Mini Apps, Docker/Compose, systemd, Caddy, YAML/CI

GetBible Robot is a hardened Telegram scripture service with immediate reference delivery and an authenticated private Mini App. It combines public bot interaction with private browsing, search and selection workflows.

Current engineering includes:

- signed Mini App authorisation;
- bounded hostile-input handling;
- asynchronous Python service behaviour;
- private navigation and multi-selection flows;
- responsive and fullscreen Bible reading;
- persistent translation preferences and translation synchronisation;
- accessible naming and reduced-motion behaviour;
- polling and webhook delivery modes;
- observability and service hardening;
- host-native systemd deployment;
- Docker Compose deployment;

- browser regression testing;
- published versioned Linux AMD64 and ARM64 images.

Robot 2.1.0 images and release-pinned Compose delivery were confirmed in the 29 July 2026 review. No current commit total is claimed because the older historical figure is obsolete and a complete replacement recount was not available.

Native Sword Extraction and PHP Integration

getBibleSword **Repository:** getbible/getbiblesword
Role: Project owner and lead engineering contributor
Technologies: C++20, Bash, CrossWire SWORD

getBibleSword is a command-line extractor that exports SWORD module families through a deterministic, versioned, loss-preserving NDJSON contract. It creates a stable native boundary for Bibles, commentaries, dictionaries and related resources.

GetBible Sword PHP Extension **Repository:** getbible/sword
Role: Principal architect, native-integration engineer and maintainer
Technologies: C, C++, Zend/PHP extension APIs, Bash, Autotools, CMake, YAML/CI

The `getbible/sword` extension exposes the released getBibleSword C ABI directly to PHP. It avoids per-request subprocess startup and does not depend on an ambient SWORD shared library.

Engineering evidence includes:

- C and Zend extension integration;
- streamed PHP APIs;
- ABI-aware boundaries;
- position-independent executable packaging;
- PHPT coverage;
- pinned native dependencies;
- architecture documentation;
- reproducible golden release evidence.

The exact comparison reviewed in July 2026 contained 41 commits. The extension exposes the released ABI and does not claim unimplemented network or module-management operations.

GetBible Scripture Application Layer **Repository:** getbible/scripture
Role: Principal application-layer architect and maintainer
Technologies: PHP, Composer, Joomla Framework, native extension integration, Bash, YAML/CI

GetBible Scripture provides a modern object-oriented application layer over the native extension. It validates the NDJSON contract and exposes immutable, lazy `Translation`, `Book`, `Chapter` and `Verse` objects without loading an entire Bible graph into every PHP process.

The architecture includes:

- contract validation;
- immutable snapshots;
- lazy domain objects;
- bounded locking;
- durable refresh and maintenance state;
- dependency injection;
- console commands;
- Joomla scheduled-task integration.

The exact July 2026 comparison contained 23 commits. The application layer correctly retains capability gates around native extension and module provisioning.

Deterministic Scripture Data Pipelines

GetBible v3 Builder **Repository:** getbible/v3_builder
Role: Principal pipeline architect and maintainer
Technologies: Python 3.12, Bash

The v3 Builder converts SWORD modules into deterministic API v3 artefacts. It applies structured token and span annotations, integrity hashes, editorial projection and publication automation under a contract-tested Python architecture.

Current work includes:

- deterministic corpus transformation;
- token/span annotations;
- CrossWire editorial projection;
- chapter editorial schema handling;
- integrity hashing;
- contract tests;
- native publication safeguards;
- reproducible data publication.

The July 2026 identity-aware snapshot recorded 307 attributable commits. Generated data is distinguished from hand-authored source.

GetBible v2 and v2 Builder GetBible v2 provides second-generation queryable JSON data for translations, books, chapters and verses, with integrity material. v2_builder converts CrossWire SWORD modules into the static API and hash artefacts. Together they demonstrate versioned API design, data publication and reproducible build pipelines.

Study and Content Pipelines v1_study_builder validates and normalises commentary and dictionary exports into independently deployable static JSON APIs. Earlier v1, browser loader and Joomla 3 integrations remain part of the historical foundation without being counted as duplicates of current systems.

Application Surfaces and Integration

- getbible/joomla-component is a multilingual Bible study extension with notes, tags, search, GUID-linked sessions and AI integration.
- getbible/mcp exposes GetBible through Model Context Protocol over authenticated streamable HTTP and stdio with validation, caching and integrity rules.
- app.getbible.life is a React/TypeScript reader and behavioural source for native implementations.
- getbible/flutter is a typed Flutter/Dart application for Android, iOS, web, Windows, macOS and Linux with local persistence.
- getbible/desktop provides a configurable native C++ WebView shell and packaging pipeline for Linux, Windows and macOS.
- joomla-scripture-loader integrates scripture loading into Joomla content.

These repositories show an ecosystem that spans source acquisition, normalisation, integrity, APIs, libraries, native boundaries, web and mobile clients, Joomla integration, public-service interaction and multi-platform delivery.

5. Official Joomla Container Engineering

Official Joomla Docker Images

Repository: joomla-docker/docker-joomla
Role: Official Joomla Docker maintainer and senior contributor
Technologies: Bash, PHP, Docker

Llewellyn contributes to the official Joomla image source used to distribute Joomla across supported PHP and web-server variants. His verified work includes:

- versioned Dockerfiles;
- container entrypoints;
- PHP configuration;
- build automation;
- multi-version compatibility;
- image maintenance and release responsibility.

The July 2026 identity-aware snapshot recorded 185 attributable commits and public activity through 7 July 2026. This is independent, official-project evidence of sustained container engineering. The role is accurately presented as maintainer and senior contributor, not sole ownership.

6. Vast Development Method Platforms and Applied AI

OpenCode Platform

Repository: vast-development-method/opencode-platform
Role: Platform architect, principal implementer and maintainer
Technologies: Bash, Python, YAML, Incus, systemd, Docker/OCI, TypeScript tooling

OpenCode Platform is a version-controlled authority for hardened agent virtual machines on Incus. It defines six Ubuntu 24.04 image variants through manifest-owned composition, resource and policy rules.

The platform addresses:

- repeatable agent VM image generation;
- capacity-safe builds;
- manifest-owned resource and policy authority;
- credentials and network boundaries;
- reproducible artefacts;
- provenance and software bills of materials;
- controlled release promotion;
- integration of `@joomengine/joomla-mcp` 0.7.0 in PHP and full images.

The complete default-branch history contained 41 commits through 29 July 2026. The project preserves explicit production gates and does not claim that every target environment is already certified.

Colibri Setup

Repository: vast-development-method/colibri-setup
Role: Systems architect, principal implementer and maintainer
Technologies: Bash, C/C++ build tooling, systemd, Node.js, Docker networking, OpenAI-compatible APIs

Colibri Setup is an operator-focused toolkit for deploying and maintaining a persistent CPU/NVMe-backed local inference service while leaving the GPU available for other work.

Its operational design includes:

- resumable large-model acquisition;
- checksum verification and surgical repair;
- host-derived resource profiles;
- an 80% ceiling on available RAM allocation;
- CPU-only startup enforcement;
- validated physical multi-NVMe reads;
- custom-profile hardening;
- systemd installation and lifecycle control;
- OpenAI-compatible service access;
- Open WebUI integration;
- deterministic tests;
- preservation of model weights during ordinary lifecycle operations.

The complete default-branch history contained 53 commits through 29 July 2026. No performance or deployment-scale claims are made because results depend on host resources and upstream model/runtime behaviour.

7. Developer Tooling, Packaging, Release Automation and CI/CD

Across JCB, JoomEngine, OctoLeo, GetBible and VDM platforms, Llewellyn repeatedly converts manual release and operational work into version-controlled systems.

Demonstrated patterns include:

- model-driven code generation;
- source-of-truth package registries;
- Git-backed reusable class distribution;
- deterministic data builders;
- Dockerfile and image-matrix generation;
- checksum and upstream hash gates;
- package, image and multi-architecture publication;
- CI validation and release gates;
- machine-readable manifests;
- browser, contract and native extension tests;
- reproducible golden artefacts;
- synchronisation, update and uninstall tooling.

This work reflects a preference for inspectable build logic, stable contracts and repeatable delivery over undocumented manual procedures.

8. Linux, Self-Hosting, Networking and Operational Tooling

Llewellyn's operational work supports both commercial delivery and public engineering. It includes:

- Linux host and service administration;
- Dockerised application environments;
- web, database and SSH services;
- self-hosted Git infrastructure;
- backups and restoration planning;
- network and firewall administration;
- OPNsense;
- systemd services;
- scheduled automation;
- TLS, DNS, SMTP, IMAP and service diagnostics;
- incident diagnosis and safe maintenance;
- local model storage and inference operations;
- Incus VM images and network policy.

Current public evidence includes:

- OctoJoom for cross-platform Docker, Traefik, OpenSSH, ACME, persistent storage, multi-user environments, backup and remote migration;
- OctoMailTest for DNS, SMTP, IMAP, ManageSieve, TLS, authentication and mail-deliverability diagnostics;
- JoomEngine for deterministic, hash-gated multi-variant container generation and publication;
- Joomla MCP for authenticated HTTP, OIDC/JWKS, least-privilege operation, Compose, systemd and live validation;
- OpenCode Platform for Incus image engineering, credentials and network boundaries, reproducible artefacts and controlled agent environments;
- Colibri Setup for CPU/NVMe-backed local inference, checksum verification and repair, multi-device resource planning and systemd lifecycle;
- GetBible Robot for a hardened public service with observability, polling/webhook operation, systemd deployment, Compose and versioned multi-architecture images;
- Backup System for historical operational and recovery automation.

These repositories make the systems claim concrete without exposing client infrastructure. They show software architecture informed by actual responsibility for the services, networks, security boundaries and recovery paths on which applications depend.

9. Historical Applications and Professional Foundations

The historical public record shows how the current platform architecture developed from real applications and domain systems.

Joomla Applications

- Joomla Cost Benefit Projection demonstrates substantial domain, database and presentation logic for a cost-benefit modelling application.
- Joomla Sermon Distributor is a complete sermon-library and media-distribution application.
- Joomla Members Manager, Support Groups, Questions and Answers, Location Data, IP Data, demo and Hello World components document reusable patterns and earlier Joomla application delivery.
- Historical GetBible components, modules and content plugins show the progression from Joomla-specific scripture delivery to the present multi-language, multi-platform architecture.

Christian Publishing Systems

- trueChristian/daily-light and daily-scripture are maintained publishing systems that generate and distribute recurring scripture content.
- hymnbook-creator is a Python utility for producing hymn booklets.
- witnesses supports automated online publication.
- sermonindex/audio_api organises a sermon corpus into static JSON APIs by Scripture, preacher and topic.

C++ and Formal-Study Foundations

Public academic and demonstration repositories include a C++ implementation of Conway's Game of Life, a Go Fish card game and coursework covering software development, networking, databases and web systems. These projects support the documented C++ certification and formal-study record without being presented as equivalent to current production platforms.

Selected Institutional and Long-Term Engagements Delivered Through Vast Development Method

The following engagements are presented at professional-objective and responsibility level. Private system names, repository details, infrastructure, security controls, internal workflows and confidential implementation information are intentionally excluded.

NamPharm | 2016–present

Long-running delivery and operational relationship in the pharmaceutical-distribution context. Llewellyn has provided senior application, integration, maintenance and continuity responsibility through Vast Development Method. The engagement demonstrates sustained trust, domain understanding and stewardship of business-critical digital services.

East African Community | 2017–present

Institutional systems delivery and continuing operational support for a regional intergovernmental organisation. Llewellyn's work through Vast Development Method has required senior architecture, information-management, integration, maintenance and continuity capability across a multi-stakeholder environment.

EU–EAC MARKUP | 2019–present

Programme-context application delivery and maintenance through Vast Development Method. The work supports the programme's information and collaboration objectives and is accurately described as contracted programme delivery rather than employment by the EU, United Nations, International Trade Centre or partner institutions.

GIZ-Linked Engagements

Selected GIZ-linked work has involved health, welfare and organisational information objectives. Llewellyn contributed application architecture and delivery through Vast Development Method. Public presentation remains at capability and objective level to protect stakeholder and system confidentiality.

Continuing Consulting Portfolio

Vast Development Method also handles shorter and medium-duration engagements across commercial, health, welfare, publishing, professional-services and community contexts. These engagements range from application delivery and integration to maintenance, hosting, operational support and modernisation. They are grouped as a consulting portfolio rather than presented as separate employment positions.

This broader record includes earlier work in biokinetics and evaluation systems, production workflow, social-impact applications, organisational websites and maintained digital services. Present scope varies by engagement and may consist primarily of maintenance, hosting, domain or operational support.

Open-Source Leadership and Joomla Community Service

Long-Term Platform Stewardship

Joomla Component Builder is Llewellyn's central open-source stewardship programme. His contribution extends beyond feature implementation to architecture, compatibility, packaging, release maintenance, documentation, user support, reusable ecosystem design and the development of other contributors.

The wider GetBible, JoomEngine, OctoLeo and True Christian estates show similar responsibility across:

- project architecture;
- direct implementation;
- issue diagnosis and maintenance;
- compatibility across platform generations;
- documentation and onboarding;
- release and distribution;
- infrastructure and public-service operations;
- review and technical guidance;
- integration between repositories and products.

Joomla Community Roles

The official Joomla community record documents service from 2016 onward across architecture, production, operations, testing, containers, web infrastructure, extensions, mentoring, language support and community organisation.

Recorded service includes:

- Software Architecture & Strategy participation and PHP leadership support;
- Production department and operations work;
- assistant-to-coordinator responsibilities;
- Joomla Extensions Directory team leadership;
- Automated Testing Team participation;
- Google Summer of Code mentoring, project leadership and code review;
- CMS maintenance and official Docker container maintenance;
- web and community-site infrastructure work;
- Framework Working Group participation;
- Windhoek Joomla user-group organisation;
- Afrikaans CMS localisation;
- security-related community service;
- service-provider and community-site work.

This breadth places Llewellyn in collaborative environments where technical changes affect other developers, extensions, infrastructure and production users. It also requires consensus, compatibility judgment, review discipline and responsible release practices.

Official Joomla Docker Maintenance

Maintenance of joomla-docker/docker-joomla provides independent evidence of practical responsibility inside an official Joomla distribution path. The work combines Docker, Bash, PHP configuration, version compatibility and release maintenance.

Localisation, Documentation and Support

Llewellyn has contributed to Afrikaans localisation and maintains extensive public documentation and educational material around Joomla Component Builder. He has also supported developers through issue analysis, architectural explanation, code review and reusable examples.

Teaching, Speaking, Mentoring and Knowledge Transfer

Developing the next generation of engineers is a sustained part of Llewellyn’s professional practice.

Formal and Technical Lectures

His teaching record includes lectures and talks on Joomla, algorithms, design patterns, software construction and architectural thinking. He is able to translate complex technical ideas into practical models that developers can apply.

Conference and Community Speaking

Verified public markers include:

- contribution to the Joomla Forum for the Future in Marbella in 2020, including reporting from the Technology Stream in official Open Source Matters minutes;
- a Joomla Component Builder session at JoomlaDay USA in 2021;
- speaking and presentation at the Anabaptist Code Blocks Summit in Pennsylvania;
- practitioner discussions on AI-assisted engineering during a recent United States visit;
- listing as a Joomla World Conference 2026 speaker at the July 2026 review date.

The 2026 conference item remains described as a listing rather than as a completed appearance.

Internship Supervision

Through Vast Development Method, Llewellyn provides internship opportunities and practical supervision. Interns are exposed to real project structure, source control, decomposition, implementation discipline, testing, deployment and maintenance rather than isolated exercises.

Developer Mentoring and Architecture Coaching

His coaching covers:

- object-oriented design and design patterns;
- decomposition of large systems into clear responsibilities;
- interfaces, contracts and dependency management;
- PHP and Joomla architecture;
- data modelling and integration;
- Linux, services and infrastructure;
- Docker and deployment;
- source control, review and release practice;
- debugging and operational reasoning;
- responsible AI-assisted software development;
- progression from task implementation toward architectural thinking.

Documentation and Public Instruction

JCB documentation, repository guidance, tutorials, conference material and public technical discussion form a continuing knowledge-transfer record. Llewellyn treats documentation and explanation as part of engineering delivery rather than as optional work after implementation.

Education, Certification and Continuing Development

Certified C++ Programmer

Concentrated Studies in C++ Programming, 2020

- Certified C++ Programmer
- C++ Programming I: **A**
- C++ Programming II: **A**

The certification is supported by formal C++ study and public C++ foundation projects. Current native work in GetBible also demonstrates applied C/C++ integration, while the certification is not used to overstate the breadth of the public C++ portfolio.

Champlain College | 2019–2022

Bachelor of Science in Web Design and Development studies — degree not completed

The supplied academic audit dated 24 August 2020 records:

- 30 earned institutional credits;
- 4.000 institutional and overall GPA;
- A grades in every listed completed credit-bearing course.

Completed study recorded in the supplied material includes:

- networking fundamentals;
- computer systems;
- operating systems;
- relational database design and SQL;
- project management;
- critical reading;
- expository writing;
- discrete mathematics;
- C++ Programming I;
- C++ Programming II.

The programme was studied from 2019 to 2022. The degree was not completed and is not presented as awarded.

Self-Directed Computer Science and Engineering Study | 2008–present

Llewellyn’s professional development has combined production work, source study, formal coursework, open-source contribution, technical reading and deliberate experimentation. Continued learning is an established working habit supporting adaptation across Joomla and PHP generations, container platforms, native integration, Python, intelligent systems and operational tooling.

Recorded Continuing-Development Courses | 2011–2015

The historical professional record contains thirty-four named technical and design courses. They are grouped below by subject while retaining their recorded titles and years.

Software Architecture, PHP, Data and Backend Systems

- **PHP with MySQL Essential Training (2007)** — Kevin Skoglund, 2011
- **PHP with MySQL Beyond the Basics** — Kevin Skoglund, 2012
- **Object-Oriented Programming with PHP** — Jon Peck, 2013
- **Foundations of Programming: Object-Oriented Design** — Simon Allardice, 2013
- **MVC Frameworks for Building PHP Web Applications** — Drew Falkman, 2013
- **Up and Running with NoSQL Databases** — Joseph LeBlanc, 2013
- **Using Regular Expressions** — Kevin Skoglund, 2013
- **Up and Running with Linux for PHP Developers** — Jon Peck, 2013

JavaScript, Browser and Web Development

- **JavaScript Essential Training** — Simon Allardice, 2011
- **AJAX Essential Training** — Dori Smith, 2011
- **jQuery Essential Training** — Joe Marini, 2011
- **Create an Interactive Map with jQuery and Dreamweaver** — Chris Converse, 2011
- **HTML5: Web Forms in Depth** — Joe Marini, 2011
- **XHTML and HTML Essential Training** — Bill Weinman, 2011
- **CSS for Developers** — Bill Weinman, 2011
- **Using Perl/CGI Scripts** — Bill Weinman, 2011
- **Building Facebook Applications with HTML and JavaScript** — Ray Villalobos, 2012
- **JavaScript and AJAX** — Ray Villalobos, 2013
- **JavaScript and JSON** — Ray Villalobos, 2013
- **Node.js First Look** — Joseph LeBlanc, 2013
- **Design the Web: SVG Rollovers with CSS** — Chris Converse, 2014

Joomla

- **Joomla! 1.5 Essential Training** — Joseph LeBlanc, 2011
- **Joomla! 1.7: Programming and Packaging Extensions** — Joseph LeBlanc, 2011

C/C++ and Java

- **C/C++ Essential Training** — Bill Weinman, 2014
- **Java Essential Training** — David Gassner, 2014
- **Java Advanced Training** — David Gassner, 2014
- **Up and Running with Java Applications** — Todd Perkins, 2014

Visual Design and Production

- **Photoshop CS6 Essential Training** — Julieanne Kost, 2012
- **Photoshop CS6 for Web Designers** — Justin Seeley, 2012
- **Photoshop CS6 One-on-One: Intermediate** — Deke McClelland, 2012
- **Design the Web: Layer Comps** — Chris Converse, 2013
- **Introduction to Graphic Design** — Justin Seeley, 2014
- **Illustrator CC 2013 Essential Training** — Justin Seeley, 2014–2015

Business Operations

- **QuickBooks Pro 2012 Essential Training** — Bonnie Biafore, 2012

Mission, Communication and Service

Faith, church responsibility, mission and teaching are a long-term foundation in Llewellyn's life. They are presented here for their direct relevance to leadership, preparation, public communication, service and continuity.

Mission and Church Leadership

- Christian conversion in 1998
- Full-time mission involvement from 2000
- Windhoek church established in 2004
- Missionary and teaching work in Namibia
- Long-running online Christian publishing
- Elder service from 2026

Preaching and Long-Form Teaching

The supplied ministry archive contains 444 distinct audio recordings from 2021 to 2026, approximately 249.67 hours, of which 437 are at least twenty minutes. A separate public Brother Llewellyn playlist on the church channel on YouTube contained 387 videos and approximately 438.9 displayed hours across visible dates from 2018 to 2023. This gives us approximately 831 sermons of approximately 689 hours over 8 years.

The record demonstrates sustained preparation, expository teaching, public communication and service. Recurrent themes include Scripture, Christ, the gospel, prayer, repentance, grace, holiness, church leadership, marriage and family, mission, forgiveness, suffering, endurance and practical wisdom.

Engineering in Service of Publishing and Access

GetBible, True Christian publishing systems, sermon distribution applications, daily scripture automation and static data APIs connect Llewellyn's engineering work with his commitment to making Scripture and teaching freely accessible.

Complete Canonical Public Engineering Catalogue

This catalogue names 101 attributable, canonical, non-fork public repository lineages selected for presentation. It is intentionally broader than a flagship list: major platforms, supporting utilities, application repositories, educational foundations and historical systems are all named. Forks, duplicate code histories, superseded delivery copies and generated-package duplicates are omitted so the same engineering work is not counted twice.

The dates matter. The long-running platforms, Joomla applications and Bash automation suites were established before modern generative coding systems and reflect original hands-on engineering. Their dated histories also show continued ownership, adaptation and maintenance across changing Joomla, PHP, Linux, container and delivery environments. Newer AI and MCP projects are presented as an extension of that established engineering base, not as its origin.

The detailed public record remains a lower bound. Vast Development Method has also delivered substantial commercial and institutional systems under confidentiality and non-disclosure obligations; those responsibilities are described elsewhere at an appropriate professional level without exposing client architecture, infrastructure or internal methods.

JoomEngine

- **joomengine/Joomla-Component-Builder** — Professional-grade Joomla extension development engine and full-stack build pipeline supporting Joomla 3 through 6, reusable powers, blueprint modelling, multi-version compilation, packaging, and Git-backed distribution. **Technologies:** PHP, JavaScript, CSS, HTML, SQL. **Role:** Founder, principal architect, lead developer and long-term maintainer. **Dated public evidence:** 2016-01-30 to 2026-05-21 (11 calendar years represented). Enables developers to model, compile, maintain, and distribute complete Joomla extensions across several CMS generations; this is the central platform behind much of the surrounding portfolio.
- **joomengine/joomla-mcp** — Embeddable library and self-hosted Model Context Protocol server for administering Joomla 6.x through bounded, auditable semantic actions. **Technologies:** TypeScript, PHP, JavaScript, JSON, Shell/Bash, Docker, YAML/CI. **Role:** Project architect, principal implementer and maintainer. Extends Joomla into modern AI and agent workflows through a source-backed semantic catalogue, guarded writes, Joomla API and native-companion transports, live validation, packaging and deployment foundations. The repository catalogues all 236 Joomla 6.1 core Web Services route templates while explicitly retaining production-certification gates.
- **joomengine/gitea** — Reusable JCB Super Power registry for typed Gitea API and repository-integration services. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-04-15 to 2026-03-24 (4 calendar years represented). Provides versioned Git-forge integration capabilities that can be injected into generated Joomla systems through stable JCB keys.
- **joomengine/joomla-powers** — Version-aware registry that lets JCB resolve Joomla classes and namespaces across CMS generations without hard-coded imports. **Technologies:** PHP, JSON, Joomla 3–6, generated registries. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2024-07-16 to 2025-12-04 (2 calendar years represented). Encodes cross-version compatibility as reusable data and services, reducing migration friction across generated applications.
- **joomengine/Joomla-Service-Directory** — Generated Joomla 6 service-provider directory component. The repository README names Lemuel van der Merwe as component author; Llewellyn's matched commits evidence build/import/release contribution rather than sole authorship. **Technologies:** PHP, HTML, CSS, JavaScript, SQL. **Role:** Build, repository and release contributor; not represented as sole component author. Demonstrates complete production application delivery, including data model, backend, frontend, access control, assets, packaging, and upgrade paths.
- **joomengine/packages** — Machine-readable JCB blueprint registry from which complete Joomla extensions can be compiled, installed and evolved. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2025-06-18 to 2026-06-24 (2 calendar years represented). Separates reusable application models from generated delivery and supports repeatable, maintainable extension production.

- **joomengine/super-powers** — Central registry of reusable namespaced PHP classes, interfaces, traits and abstractions managed by JCB and injected through stable identifiers. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-03-22 to 2026-02-18 (4 calendar years represented). Creates a shared architectural layer across generated projects and demonstrates ecosystem-scale dependency design.
- **joomengine/pkg-component-builder** — Installable Joomla 6 Joomla Component Builder release bundle that assembles the JCB component and its supported plugin set. **Technologies:** PHP, XML, Joomla packaging, release engineering. **Role:** Author, packager and release maintainer. Shows release composition across eighteen bundled extensions and maintained Joomla 6.0–6.3 compatibility without double-counting the JCB application itself.
- **joomengine/pkg-servicedirectory** — Installable Joomla 6 package for the Service Directory application, bundling two related extensions for controlled installation and update. **Technologies:** PHP, XML, Joomla packaging, release engineering. **Role:** Build, packaging and release contributor.
- **joomengine/fof** — Maintained compatibility surface for FOF framework classes used by Joomla extension systems. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-10-09 to 2025-06-17 (3 calendar years represented).
- **joomengine/jcb-documentation** — Long-running technical documentation and onboarding corpus for Joomla Component Builder. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2016-10-20 to 2026-03-24 (11 calendar years represented).
- **joomengine/jcb-external** — Registry of external source and dependency definitions used by JCB compilation and package workflows. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-10-02 to 2025-06-05 (3 calendar years represented).
- **joomengine/joomla-fieldtypes** — Machine-readable catalogue and documentation for Joomla and JCB field types across platform generations. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2024-08-23 to 2026-04-09 (3 calendar years represented).
- **joomengine/minify** — Maintained minification-service integration for generated Joomla assets and packages. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-04-15 to 2025-06-17 (3 calendar years represented).
- **joomengine/openai** — Reusable JCB Super Power registry for OpenAI service integration within generated Joomla systems. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-05-20 to 2026-03-24 (4 calendar years represented).
- **joomengine/phpseclib** — Maintained phpseclib integration surface for reusable cryptographic and secure-transport capabilities in JCB systems. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-04-15 to 2026-03-24 (4 calendar years represented).
- **joomengine/psr** — Maintained PSR interface and compatibility surface for JCB-generated PHP applications. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-04-15 to 2026-03-24 (4 calendar years represented).
- **joomengine/repoindex** — Repository-index data used to discover and coordinate JCB ecosystem sources and delivery surfaces. **Role:** JCB ecosystem architect, registry maintainer and release contributor.
- **joomengine/search** — Reusable JCB Super Power registry for search abstractions and service integrations. **Technologies:** PHP. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2023-04-15 to 2026-03-24 (4 calendar years represented).
- **joomengine/snippets** — Reusable UI, HTML and layout-snippet registry for JCB-built Joomla views and templates. **Technologies:** HTML, CSS, JavaScript, Joomla, generated registries. **Role:** JCB ecosystem architect, registry maintainer and release contributor.
- **joomengine/uikit** — Maintained UIkit integration assets and compatibility surface for JCB-generated application interfaces. **Technologies:** JavaScript. **Role:** JCB ecosystem architect, registry maintainer and release contributor. **Dated public evidence:** 2020-09-04 to 2025-12-29 (6 calendar years represented).

OctoLeo

- **octoleo/octojoom** — Bash-based utility for deploying and managing Dockerised Joomla and OpenSSH environments through interactive and direct CLI workflows across Linux, macOS and Windows. **Technologies:** Shell/Bash, Docker, Docker Compose, HTML, cross-platform deployment tooling. **Role:** Creator, principal Bash engineer and long-term maintainer. **Dated public evidence:** 2021-05-03 to 2026-02-26 (6 calendar years represented). Demonstrates cross-platform shell engineering, environment/configuration management, Docker orchestration, self-update/uninstall flows and operator-focused developer tooling.
- **octoleo/joomengine** — Official Docker image build system for Joomla Component Builder, generating deterministic image matrices across Joomla, PHP and runtime variants. **Technologies:** Docker, Shell/Bash, YAML/CI. **Role:** Creator / lead architect and maintainer. Automates release discovery, hash verification, Dockerfile generation, tag leadership, machine-readable manifests and image publication without hiding the build logic in CI YAML.
- **octoleo/octojpack** — JSON- and environment-driven Bash packaging engine for Joomla extensions, including source collection, manifest assembly, archive creation and release-oriented output. **Technologies:** Shell/Bash, JSON, Joomla packaging, CI/CD. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-06-08 to 2025-09-20 (5 calendar years represented). Turns a complex Joomla release procedure into a reusable, deterministic packaging system used across a wider extension ecosystem.
- **octoleo/octopower** — Bash packaging system that extracts and assembles Joomla Component Builder Power classes into versioned Composer-ready packages. **Technologies:** Shell/Bash, PHP, Composer, JSON, CI/CD. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-06-08 to 2025-03-24 (5 calendar years represented). Bridges JCB's reusable class model with conventional PHP distribution and demonstrates source transformation, dependency packaging and release automation.
- **octoleo/octoshoom** — Release-integrity utility that calculates SHA-512 hashes for Joomla update archives, injects checksum elements into update XML and commits the resulting release metadata. **Technologies:** Shell/Bash, SHA-512, XML, Git, CI/CD. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-08-23 to 2025-09-20 (5 calendar years represented). Adds repeatable integrity controls to extension publication and connects archive production, metadata mutation and source-controlled release evidence.
- **octoleo/octozipo** — Bash utility for unpacking release archives, transforming their contents and updating target repositories as part of a repeatable delivery workflow. **Technologies:** Shell/Bash, Git, archives, release automation. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2020-11-11 to 2025-12-01 (6 calendar years represented). Demonstrates practical release engineering, source transformation and reliable propagation of packaged deliverables.
- **octoleo/octomailtest** — Deep mail-system diagnostic toolkit covering DNS, connectivity, SMTP, IMAP, TLS, authentication and deliverability, with structured output suitable for operators and automation. **Technologies:** Shell/Bash, Docker, DNS, SMTP, IMAP, TLS, JSON, CI/CD. **Role:** Creator, lead Bash engineer and maintainer. Demonstrates advanced systems diagnosis, secure protocol handling, failure classification and operator-focused network tooling.
- **octoleo/octodinator** — First-start/last-stop lifecycle coordinator that uses shared lock state to run initialization once when the first process starts and cleanup once when the last process exits. **Technologies:** Shell/Bash, JSON, process locks. **Role:** Original author and maintainer.
- **octoleo/octoflare** — Focused Bash command-line utility for checking, enabling and disabling Cloudflare Under Attack Mode through the Cloudflare v4 API. **Technologies:** Shell/Bash, Cloudflare API, JSON. **Role:** Originator/maintainer of mirrored deliverable.
- **octoleo/octolanding** — Minimal container-ready static landing surface for infrastructure endpoints, reverse proxies and default service routes. **Technologies:** YAML/CI, HTML, Docker. **Role:** Principal or sole recorded author.
- **octoleo/plantuml-gitea** — JavaScript integration that renders PlantUML embedded in Gitea-hosted Markdown and HTML code blocks. **Technologies:** JavaScript, PlantUML, Gitea, HTML. **Role:** Adapter and maintainer of the Gitea integration. **Dated public evidence:** 2022-08-22 to 2025-11-14 (4 calendar years represented).

- **octoleo/octosync** — Repository synchronisation bot that turns repeatable cross-repository update operations into a version-controlled Bash workflow. **Technologies:** Shell/Bash, Git, GitHub Actions, CI/CD. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-07-13 to 2021-08-31 (1 calendar year represented).
- **octoleo/git-user** — Security-conscious Bash and GitHub Action bootstrap for GPG-signed Git operations, SSH identity validation and idempotent CI configuration on Ubuntu. **Technologies:** Shell/Bash, GitHub Actions, GPG, SSH, Git. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-04-30 to 2021-04-30 (1 calendar year represented).

GetBible

- **getbible/librarian** — Python library for resolving scripture references, retrieving verses and performing Unicode-aware searches against GetBible translations. **Technologies:** Python, packaging/PyPI, Unicode search, HTTP API integration, caching, YAML/CI. **Role:** Original author, principal architect and maintainer. **Dated public evidence:** 2023-11-11 to 2026-07-20 (4 calendar years represented). Provides reusable reference and search contracts, checksum-verified bounded caches, multi-worker operations, PyPI distribution, and separate Query/Search HTTP service integration.
- **getbible/robot** — Hardened Telegram Scripture service with immediate reference delivery, authenticated private Mini App navigation and search, bounded multi-selection, responsive/fullscreen reader UX, persistent translations, polling/webhook operation, and container or systemd deployment. **Technologies:** Python, TypeScript/JavaScript, HTML/CSS, Docker/Compose, systemd, Caddy, Telegram Mini Apps, YAML/CI. **Role:** Project architect, principal implementer and maintainer. **Dated public evidence:** 2020-02-02 to 2026-07-29 (7 calendar years represented). Demonstrates public-service reliability, hostile-input boundaries, asynchronous Python, signed web-app authorization, accessible responsive interaction, browser regression testing, Docker/Compose distribution, and operational hardening.
- **getbible/sword** — Native PHP extension exposing the released getBibleSword C ABI directly to PHP without a subprocess or ambient SWORD shared-library dependency. **Technologies:** C, C++, PHP extension APIs, Shell/Bash, Autotools, CMake, YAML/CI. **Role:** Principal architect, native-integration engineer and maintainer. Demonstrates C/C++ and Zend/PHP extension work, deterministic dependency pinning, PIE packaging, streamed APIs, PHPT coverage, architecture documentation and reproducible golden release evidence.
- **getbible/scripture** — Object-oriented Bible application layer for the native getbible/sword extension, providing validated immutable and lazy Translation, Book, Chapter and Verse objects. **Technologies:** PHP, Composer, Joomla Framework, native extension integration, Shell/Bash, YAML/CI. **Role:** Principal application-layer architect and maintainer. Adds contract validation, immutable snapshots, bounded locking, durable refresh state, console commands and Joomla scheduled-task integration without loading an entire Bible graph into each PHP process.
- **getbible/v3_builder** — SOLID Python 3.12 build pipeline converting SWORD modules into API v3 with token/span annotations, deterministic hashing, publication automation, and a large contract test suite. **Technologies:** Python, Shell/Bash. **Role:** Principal pipeline architect and maintainer. **Dated public evidence:** 2021-04-20 to 2026-07-24 (6 calendar years represented). Turns upstream SWORD modules into deterministic, testable and distributable scripture APIs instead of hand-maintained datasets.
- **getbible/desktop** — Configurable cross-platform native C++ WebView shell and packaging pipeline for Linux, Windows, and macOS GetBible desktop applications. **Technologies:** Shell/Bash, Python. **Role:** Project owner and lead engineering contributor. Extends the GetBible platform into maintained end-user application surfaces and cross-platform distribution.
- **getbible/flutter** — Native Flutter/Dart implementation of GetBible for Android, iOS, web, Windows, macOS, and Linux with typed API models and local persistence. **Technologies:** Dart, C/C++ headers, Swift, C++. **Role:** Project owner and lead engineering contributor. Extends the GetBible platform into maintained end-user application surfaces and cross-platform distribution.
- **getbible/getbiblesword** — C++20 command-line extractor built on CrossWire SWORD, exporting every module family through a deterministic, versioned, loss-preserving NDJSON contract. **Technologies:** C++, Shell/Bash. **Role:** Project owner and lead engineering contributor. Establishes a loss-preserving C++ extraction boundary for Bibles, commentaries, dictionaries and other SWORD resources.

- **getbible/joomla-component** — Feature-rich Bible study extension for Joomla with multilingual scripture, notes, tags, search, GUID-linked sessions, and OpenAI integration. **Technologies:** PHP, HTML, SQL, JavaScript, CSS. **Role:** Original author, principal architect and maintainer. **Dated public evidence:** 2023-07-26 to 2025-11-18 (3 calendar years represented). Makes multilingual Bible reading and study features deployable on Joomla sites, with personal notes, tags, search and session sharing.
- **getbible/mcp** — Production Model Context Protocol server exposing GetBible over streamable HTTP and stdio with validation, caching, and integrity rules. **Technologies:** Python, HTML. **Role:** Project owner and lead engineering contributor. Makes the GetBible data platform directly usable by AI clients through standard MCP transports.
- **getbible/v2** — Second-generation GetBible JSON API and queryable scripture data distribution, including translation, book, chapter, verse, and integrity/hash material. **Role:** Project founder, data/API architect and publisher. **Dated public evidence:** 2019-12-29 to 2026-04-01 (8 calendar years represented). Publishes a globally reusable, registration-free scripture API/data layer and supports the live grouped-reference query service.
- **getbible/v2_builder** — Python and shell build pipeline converting CrossWire SWORD modules into GetBible API v2 static JSON and hash artefacts. **Technologies:** Shell/Bash, Python. **Role:** Principal pipeline architect and maintainer. **Dated public evidence:** 2021-04-20 to 2026-07-16 (6 calendar years represented). Turns upstream SWORD modules into deterministic, testable and distributable scripture APIs instead of hand-maintained datasets.
- **getbible/joomla-pkg** — Joomla 5 package and release assembler bundling the GetBible component, Daily Light module, Daily Scripture module and loader plugin. **Technologies:** XML, Joomla packaging, PHP, release engineering. **Role:** Author, packager and maintainer. **Dated public evidence:** 2023-07-26 to 2025-11-18 (3 calendar years represented). Demonstrates multi-extension package composition, version compatibility and sustained distribution responsibility across the GetBible Joomla surface.
- **getbible/app.getbible.life** — React/TypeScript GetBible reader application and behavioural source of truth for the native Flutter implementation. **Technologies:** TypeScript, CSS, TypeScript/React. **Role:** Project owner and lead engineering contributor.
- **getbible/joomla-scripture-loader** — Joomla content plugin mirror for loading scripture in articles and other content. **Technologies:** PHP, HTML. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2023-11-09 to 2025-11-18 (3 calendar years represented).
- **getbible/v1_study_builder** — Python pipeline that validates and normalizes SWORD commentary and dictionary exports into independently deployable static JSON APIs. **Technologies:** Python. **Role:** Project owner and lead engineering contributor.
- **getbible/getverse** — Small shell client for retrieving verses from the GetBible API. **Technologies:** Shell/Bash. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2021-05-02 to 2023-06-01 (3 calendar years represented).
- **getbible/Joomla-3-Activity-Cron-Plugin** — Home of the getBible Activity Cron Job for Joomla 3.x **Technologies:** PHP, HTML, SQL. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2015-01-05 to 2016-02-21 (2 calendar years represented).
- **getbible/Joomla-3-Component** — Historical Joomla 3 GetBible component. **Technologies:** PHP, CSS, JavaScript, HTML, SQL. **Role:** Original author, principal architect and maintainer. **Dated public evidence:** 2014-03-14 to 2017-01-16 (4 calendar years represented).
- **getbible/Joomla-3-Load-Scripture-Plugin** — Home of the getBible Load Scripture in Content for Joomla 3.x **Technologies:** PHP, HTML. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2014-11-09 to 2016-04-11 (3 calendar years represented).
- **getbible/Joomla-3-Search-Module** — Home of the getBible search module for Joomla 3.x **Technologies:** PHP, HTML, JavaScript, CSS. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2014-11-08 to 2016-02-21 (3 calendar years represented).
- **getbible/loader** — Browser-side JavaScript loader that inserts scripture from GetBible into ordinary HTML elements. **Technologies:** JavaScript, HTML. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2023-11-08 to 2024-06-15 (2 calendar years represented).
- **getbible/txtFactory** — Historical Bash data-acquisition utility for collecting Bible translations from the Unbound Biola source for downstream GetBible processing. **Technologies:** Shell/Bash. **Role:** Project owner and lead engineering contributor. **Dated public evidence:** 2016-07-18 to 2016-07-24 (1 calendar year represented).

- **getbible/v1** — First-generation public static JSON Bible API dataset. **Role:** Project founder, data/API architect and publisher. **Dated public evidence:** 2019-12-29 to 2019-12-29 (1 calendar year represented).

Vast Development Method

- **vast-development-method/opencode-platform** — Version-controlled platform authority for hardened OpenCode agent VMs on Incus, six Ubuntu 24.04 image variants, manifest-owned composition and capacity rules, credentials/network boundaries, reproducible artifacts, and controlled Joomla MCP integration. **Technologies:** Shell/Bash, Python, YAML, Incus, systemd, Docker/OCI, TypeScript tooling. **Role:** Platform architect, principal implementer and maintainer. Demonstrates secure agent-platform architecture across image generation, network policy, runtime credentials, SBOM/provenance, release promotion and Joomla MCP integration in PHP/full images.
- **vast-development-method/colibri-setup** — Operator-focused CPU/NVMe Colibri deployment toolkit with GPU exclusion, host-derived resource profiles capped at 80% of available RAM, checksum verification and surgical repair, validated physical multi-NVMe reads, systemd lifecycle, and Open WebUI integration. **Technologies:** Shell/Bash, C/C++ build tooling, systemd, Node.js, Docker networking, OpenAI-compatible APIs. **Role:** Systems architect, principal implementer and maintainer. Shows resource-aware local-model deployment, resumable large-model acquisition, checksum verification and repair, multi-NVMe operation, systemd lifecycle management and Open WebUI integration.

True Christian Publishing and Ministry Systems

- **trueChristian/daily-light** — A maintained Daily Light publishing system that generates and distributes devotional Scripture content. **Technologies:** HTML, Shell/Bash, YAML/CI. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-04-30 to 2026-07-18 (6 calendar years represented). Demonstrates the design and sustained operation of daily publication/data pipelines over more than five years.
- **trueChristian/daily-scripture** — A maintained daily Scripture publication system with static web/data outputs and automation. **Technologies:** HTML, Shell/Bash, YAML/CI. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-05-02 to 2026-07-17 (6 calendar years represented). Demonstrates the design and sustained operation of daily publication/data pipelines over more than five years.
- **trueChristian/hymnbook-creator** — A Python hymn-booklet generation utility. **Technologies:** Make, Python. **Role:** Principal or sole recorded author.
- **trueChristian/Joomla-Daily-Light** — Joomla module delivering the Daily Light scripture-reading sequence from the maintained True Christian publishing data. **Technologies:** XML, PHP, HTML. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2023-04-09 to 2025-11-18 (3 calendar years represented).
- **trueChristian/Joomla-Daily-Scripture** — Joomla module for publishing the maintained Daily Scripture sequence within Joomla sites. **Technologies:** XML, PHP, HTML. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2022-01-04 to 2025-11-18 (4 calendar years represented).
- **trueChristian/witnesses** — An automated online witnessing/publication system. **Technologies:** YAML/CI, Shell/Bash. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-05-03 to 2026-07-16 (6 calendar years represented).

Personal Engineering and Joomla Applications

- **Llewellynvdm/Joomla-Cost-Benefit-Projection** — A Joomla cost-benefit projection application with substantial domain, database and presentation logic. **Technologies:** PHP, CSS/SCSS, JavaScript, XML, HTML, SQL, Shell/Bash. **Role:** Principal or sole recorded author. **Dated public evidence:** 2015-12-01 to 2022-05-27 (8 calendar years represented). Delivered attributable changes across 65 commits and 2004 paths in Llewellynvdm/Joomla-Cost-Benefit-Projection; the visible history demonstrates hands-on implementation and maintenance of a joomla cost-benefit projection application with substantial domain, database and presentation logic.
- **Llewellynvdm/Joomla-Sermon-Distributor** — A complete Joomla sermon-library and distribution application. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL, YAML/CI. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2015-11-30 to 2024-05-03 (10 calendar years represented). Demonstrates full-stack Joomla application architecture for cataloguing, managing and distributing media-rich sermon content.
- **Llewellynvdm/game-of-life** — C++ implementation of Conway's Game of Life used to demonstrate object-oriented modelling, state transitions and algorithmic reasoning. **Technologies:** C++, C. **Role:** Principal or sole recorded author. **Dated public evidence:** 2020-05-04 to 2024-01-13 (5 calendar years represented).
- **Llewellynvdm/Joomla-Daily-Scripture** — Historical public distribution of the Joomla Daily Scripture module, preserving the earlier application lineage before current True Christian custody. **Technologies:** XML, PHP, HTML. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2022-01-04 to 2023-11-09 (2 calendar years represented).
- **Llewellynvdm/Llewellynvdm** — Personal GitHub profile source and curated public navigation for Llewellyn's engineering identity, current work and professional contact routes. **Role:** Principal or sole recorded author. **Dated public evidence:** 2020-11-03 to 2026-02-01 (7 calendar years represented).
- **Llewellynvdm/mastodon-secure-feed** — Python service for publishing or consuming a security-conscious Mastodon feed through a self-hosted workflow. **Technologies:** Python. **Role:** Originator/maintainer of mirrored deliverable.
- **Llewellynvdm/mod_version_calendar_svg** — Version Calendar in SVG **Technologies:** PHP, HTML, XML. **Role:** Principal or sole recorded author. **Dated public evidence:** 2022-09-21 to 2023-08-11 (2 calendar years represented).
- **Llewellynvdm/PreUpVer** — The easy pre tag updater for include statments. **Technologies:** JavaScript, HTML. **Role:** Principal or sole recorded author.
- **Llewellynvdm/PreUpVer-Plugin** — PreUpVer JS loader for Joomla. **Technologies:** HTML, PHP, XML. **Role:** Principal or sole recorded author.
- **Llewellynvdm/Backup-System** — The Bash scripts used to backup database and folders on a server **Technologies:** Shell/Bash. **Role:** Lead maintainer / major contributor. **Dated public evidence:** 2017-06-05 to 2018-07-16 (2 calendar years represented).
- **Llewellynvdm/CoinRate** — Bash command-line utility for retrieving and presenting cryptocurrency exchange-rate data. **Technologies:** Shell/Bash. **Role:** Principal or sole recorded author. **Dated public evidence:** 2017-12-27 to 2018-01-17 (2 calendar years represented).
- **Llewellynvdm/GoFish** — "Go Fish" is a classic children's card game I wrote in C++ **Technologies:** C++. **Role:** Author / implementation contributor.
- **Llewellynvdm/Joomla-Demo-Component** — Generated demonstration Joomla component used to show JCB structure, installation, interfaces and extension-delivery conventions. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2015-08-24 to 2022-05-27 (8 calendar years represented).
- **Llewellynvdm/Joomla-Hello-World-Component** — Introductory Joomla component demonstrating the minimum generated JCB application structure and packaging workflow. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2018-05-05 to 2022-05-27 (5 calendar years represented).
- **Llewellynvdm/Joomla-Members-Manager** — Joomla application for managing member records, relationships and administrative workflows. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2018-07-11 to 2022-05-27 (5 calendar years represented).

- **Llewellynvdm/Joomla-Questions-and-Answers** — Joomla question-and-answer application with structured content, administration and user-facing interaction. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2018-04-24 to 2022-05-27 (5 calendar years represented).
- **Llewellynvdm/Joomla-Support-Groups** — Joomla application supporting structured group, participant and programme administration. **Technologies:** PHP, CSS/SCSS, JavaScript, HTML, XML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2016-03-06 to 2022-03-03 (7 calendar years represented).
- **Llewellynvdm/Joomla-XML-Lang-stream** — XML and Bash automation for synchronising Joomla language material through a source-controlled workflow. **Technologies:** XML, Shell/Bash, YAML/CI. **Role:** Principal or sole recorded author. **Dated public evidence:** 2021-06-01 to 2021-06-01 (1 calendar year represented).
- **Llewellynvdm/mod_pergroup** — Joomla module for presenting content or functionality according to configured user-group context. **Technologies:** PHP, HTML, XML, JavaScript, CSS/SCSS. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2014-06-27 to 2019-08-05 (6 calendar years represented).

Official Joomla Project Contribution

- **joomla-docker/docker-joomla** — Official Docker image source for Joomla; matched work covers multi-version Dockerfiles, entrypoints, PHP configuration, build automation, and maintenance. **Technologies:** Shell/Bash, PHP. **Role:** Official Joomla Docker maintainer and senior contributor. **Dated public evidence:** 2021-08-24 to 2026-07-07 (6 calendar years represented). Contributes directly to the official container distribution path used to run Joomla across supported PHP and web-server variants.

SermonIndex Data Systems

- **sermonindex/audio_api** — A static JSON API organizing the SermonIndex corpus by Scripture, preacher and topic. **Role:** Lead maintainer / major contributor. **Dated public evidence:** 2019-01-02 to 2020-05-19 (2 calendar years represented). Turns a large sermon catalogue into queryable, deployment-friendly static data grouped across multiple retrieval dimensions.

Formal-Study Software Repositories

- **mychamplain/SDEV-240-81** — Formal-study C++ repository covering intermediate software-development concepts and implementation exercises. **Technologies:** C++. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2020-02-02 to 2020-05-02 (1 calendar year represented).
- **mychamplain/SDEV-324-81** — Formal-study C++ repository for advanced programming exercises and software-construction practice. **Technologies:** C++. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2020-09-12 to 2020-12-18 (1 calendar year represented).
- **mychamplain/SDEV-340-81** — Formal-study C and C++ systems-programming repository. **Technologies:** C++, C. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2020-05-16 to 2020-08-17 (1 calendar year represented).
- **mychamplain/SDEV-415-81** — Formal-study C and Bash repository focused on software-development and systems-oriented implementation. **Technologies:** Shell/Bash, C. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2020-09-19 to 2020-12-17 (1 calendar year represented).
- **mychamplain/WEBD-125-40** — Formal-study HTML and CSS repository covering foundational standards-based web presentation. **Technologies:** HTML, CSS/SCSS. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-09-04 to 2021-10-14 (1 calendar year represented).
- **mychamplain/WEBD-225-40** — Formal-study front-end repository combining HTML, CSS and JavaScript application work. **Technologies:** HTML, CSS/SCSS, JavaScript. **Role:** Originator/maintainer of mirrored deliverable.
- **mychamplain/WEBD-310-45** — Formal-study PHP, SQL and HTML repository covering server-side web application development. **Technologies:** PHP, HTML, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-10-30 to 2021-12-18 (1 calendar year represented).
- **mychamplain/WEBD-325-45** — Formal-study full-stack web repository using PHP, SQL, HTML, CSS and JavaScript. **Technologies:** PHP, HTML, CSS/SCSS, JavaScript, SQL. **Role:** Originator/maintainer of mirrored deliverable.

Historical Namibian Joomla Applications

- **namibia/eHealth-Portal** — A Joomla-based eHealth portal delivered as an application artefact. **Technologies:** PHP, HTML, XML, CSS/SCSS, JavaScript, SQL. **Role:** Originator/maintainer of mirrored deliverable. **Dated public evidence:** 2021-04-24 to 2024-01-19 (4 calendar years represented).
- **namibia/CBP-Joomla-2-Component** — Historical Joomla 2 cost-benefit projection component implementing structured domain, data and presentation logic. **Technologies:** PHP, HTML, JavaScript, CSS/SCSS, XML, SQL. **Role:** Principal or sole recorded author. **Dated public evidence:** 2014-03-31 to 2014-05-21 (1 calendar year represented).
- **namibia/dropboxlinks** — Historical PHP utility for generating shareable links from an organised Dropbox audio collection. **Technologies:** PHP. **Role:** Principal or sole recorded author. **Dated public evidence:** 2015-07-01 to 2015-07-01 (1 calendar year represented).
- **namibia/ipdata-joomla-3-component** — Historical Joomla 3 component for maintaining and presenting IP-related reference data; retained as a superseded application lineage. **Technologies:** PHP, HTML, JavaScript, XML, CSS/SCSS, SQL. **Role:** Principal or sole recorded author. **Dated public evidence:** 2015-01-03 to 2017-02-02 (3 calendar years represented).

Framework Experimentation

- **Kumwe/cms** — Little cms build with Joomla Framework **Technologies:** PHP, HTML, JavaScript, CSS/SCSS, SQL. **Role:** Principal or sole recorded author.

Sentinel Systems

- **sentinel-mx/server** — PHP-based server application combining administrative interfaces, structured data and deployment workflow within the Sentinel MX system. **Technologies:** PHP, CSS/SCSS, HTML, JavaScript, XML, SQL, YAML/CI. **Role:** Principal or sole recorded author. **Dated public evidence:** 2020-02-26 to 2020-02-26 (1 calendar year represented).

Role Fit

Llewellyn is well suited to senior work requiring substantial autonomy and end-to-end responsibility in:

- software and systems architecture;
- principal or senior engineering;
- technical direction in a focused engineering organisation;
- platform and framework engineering;
- developer tooling and code generation;
- PHP and Joomla ecosystems;
- data, API and integration systems;
- Linux, infrastructure, deployment and operations;
- Docker and reproducible delivery;
- open-source technical leadership;
- native C/C++ and PHP integration;
- applied AI, MCP and intelligent-system integration;
- legacy modernisation and long-term platform stewardship;
- technical teaching, mentoring and developer development.

His strongest contribution is made where architecture must remain connected to implementation, production operations, maintenance and people development.

Contact

Contact via Telegram

t.me/llewellynvdm

Professional website: vdm.io/llewellyn
GitHub: github.com/Llewellynvdm
Company engineering estate: git.vdm.dev
Location: Windhoek, Namibia