# ZNotice — Changelog

Build history of the ZNotice SaaS. Dates are phase completion dates; the full stack was deployed to production by **2026-09-10**. Version follows the `package.json` `0.1.0` line.

## Phase 14 — IntouchPay mobile money payments (MTN MoMo / Airtel Money) — 2026-09-16

Adds a live **IntouchPay** mobile-money gateway (Rwanda: MTN MoMo and Airtel Money), the first real (non-manual) payment provider. Customers pay instantly from the Billing page and the transaction is auto-confirmed by cross-checking the gateway — no admin approval and no proof upload required for credit top-ups. Live mode, with keys supplied by the operator.

### Added

- **Provider** `intouch` registered in `server/services/payments/index.js` alongside `manual`. The shared `settlePaid()` helper gained an optional credit-ledger **note override** so Intouch settlements can tag the ledger entry distinctly.
- **New gateway service** `server/services/payments/intouch.js` (uses the global `fetch`, no new dependencies). Auth is a **SHA256 hex hash** over `username + account_number + partner_password + timestamp` (raw concatenation, UTC `YYYYMMDDHHMMSS`); the account number and partner password never leave the hash. Requests hit `POST /requestpayment/` (async, `Pending` `1000`) and `POST /gettransactionstatus/` (no amount) against the live `https://www.intouchpay.co.rw/api` or sandbox `https://developer.intouchpay.co.rw/api/v1/sandbox`, plus `POST /getbalance/` for ops. Rwandan numbers are normalized to `2507XXXXXXXX`; provider refs are `ZNI-<id>-<8hex>`; every request carries an `X-Trace-Id` header; 30s timeout.
- **Dashboard Intouch checkout.** A "Pay instantly with MTN MoMo / Airtel Money" card on the Billing page (amount + phone, with an RWF hint at the operator rate). Routes `POST /dashboard/billing/intouch/checkout`, `POST /dashboard/billing/intouch/:id/start` (re-request a USSD prompt) and `POST /dashboard/billing/intouch/:id/status` (manual status check). The pay page gained an Intouch branch with no manual instructions / proof UI.
- **Currency conversion.** USD prices are converted to whole RWF at checkout using the admin-editable `intouch_rwf_rate` (float, default `1477.29`, migration `013_intouch.sql`; `settings.getFloat()` added). Transaction `meta` carries `{ currency:'RWF', amount_rwf, rate, intouch:{ phone, tx_id, provider, statusdesc } }`; the `transactions` row stays in USD (`amount_cents`).
- **Webhook** `POST /payments/intouch/webhook` (mounted **before** session/CSRF, no auth — the callback is **unsigned**). The payload is wrapped in a `jsonpayload` field (object or JSON string). See Security below for the cross-check/ACK contract.
- **Recovery sweep** `server/services/intouchSweep.js`: every **5 minutes** cross-checks pending `intouch` transactions older than **2 minutes** and confirms settled ones (heals lost/early callbacks).
- **Admin UI.** The pending-payments table shows a provider chip; `intouch` rows show a muted "auto-confirmed by InTouch" hint instead of Approve/Reject.

### Changed

- `server/services/payments/index.js` now resolves `providers = { manual, intouch }`.

- Credit top-ups via Intouch settle through the same `settlePaid()` path as manual payments, but with the RWF amount and provider metadata attached.
- The sender-name add-on and the integration service remain **manual only** (no Intouch checkout), so their flows are unchanged. **No API v1 changes.**

**Security**

- The webhook is unsigned, so every callback is **cross-checked** against `gettransactionstatus` (3 bounded retries) before any settlement; the response is always a pure ACK `200 {"message":"success","success":true, "request_id":"<uuid>"}`, and malformed/unknown/mismatched payloads are logged and ACKed `200` rather than leaking validation detail or retrying. Terminal failures are recorded in `meta.failure_reason`.
- Only the account number + partner password are hashed into the auth digest — neither is ever transmitted in cleartext.
- Whole-RWF amounts are strictly matched (currency + amount) at confirmation, so an off-by-one or mismatched amount can never settle.
- See SECURITY.md §3.7 for the full control list.

**Tests:** `npm test` → **136 unit tests**; `npm run test:integration` → **149 DB-backed tests**. New coverage: `test/unit/intouch.test.js` (auth digest, RWF amount/currency math, phone normalization, webhook payload unwrapping/ACK) and `test/integration/intouch.test.js` (checkout → webhook → cross-check → settlement, sweep healing, `failure_reason` recording).

## Phase 15 — KitovuPay mobile money (MTN MoMo) — 2026-09-18

Adds a second live mobile-money gateway, **KitovuPay** (Rwanda, MTN MoMo), alongside IntouchPay. Customers pay instantly from the Billing page and the transaction is auto-confirmed by cross-checking the gateway — no admin approval and no proof upload for credit top-ups. The wire protocol is ported from the reference Laravel integration (zeron-xs `KitovuService`); the UAT merchant account is shared with zeron-xs, so the 6-digit external IDs must be unique.

**Added**

- **Provider `kitovu`** registered in `server/services/payments/index.js` alongside `manual` and `intouch`. `providers = { manual, intouch, kitovu }`.
- **New gateway service `server/services/payments/kitovu.js`** (global `fetch`, no new dependencies). Auth is an **HMAC-SHA256 signature** (lowercase hex) over the compact JSON envelope `{"method":"POST","path":"<path>","body":{...}}` keyed with the API secret; `x-api-timestamp` is Unix **milliseconds as a string** (`Date.now().toString()`). GET status uses only `x-api-key`. Collection hits `POST /merchant-transaction/merchant/collection/mobile_money` (`amount`, `payment_rail:'mtn'`, `sender_account_issuer:'mtn'`, `sender_account_name`, `sender_account_number`, `description`, `callbackUrl`, `externalId`); status is `GET /merchant-transaction/payment/:externalId`. 429 responses are retried with bounded exponential backoff (3 attempts, 250ms/500ms); 30s timeout. The `description` is sanitized to plain text (symbols break the USSD push).
- **6-digit `externalId` allocation.** The reference is a unique 6-digit string (the gateway's lookup key). It is checked for uniqueness in `transactions.provider_ref` before sending, and a gateway rejection whose message mentions `DUPLICATE_EXTERNAL_ID` allocates a fresh ref and retries (bounded). `provider_ref` stays `NULL` until a collection request is sent.

- **Dashboard Kitovu checkout.** A "Pay instantly with MTN MoMo (KitovuPay)" card on the Billing page (amount + phone + optional account name, RWF hint). Routes `POST /dashboard/billing/kitovu/checkout`, `POST /dashboard/billing/kitovu/:id/start` (re-request) and `POST /dashboard/billing/kitovu/:id/status` (manual status check). The pay page gained a Kitovu branch with no manual instructions / proof UI.
- **Currency conversion.** USD prices convert to whole RWF at the operator rate `kitovu_rwf_rate` (float, default `1477.29`, migration `014_kitovu.sql`; `settings.getFloat()`). `meta` carries `{ currency:'RWF', amount_rwf, rate, kitovu:{ phone, tx_id, status, requested_at } }`; the `transactions` row stays in USD.
- **Webhook** `POST /payments/kitovu/webhook` (mounted **before** session/CSRF, no auth — the callback is **unsigned**). See Security below.
- **Recovery sweep** `server/services/kitovuSweep.js`: every **5 minutes** cross-checks pending `kitovu` transactions older than **2 minutes** and confirms settled ones.
- **Admin UI.** The pending-payments table shows a provider chip; `kitovu` rows show a muted "auto-confirmed by KitovuPay" hint instead of Approve/Reject.

**Changed**

- `server/services/payments/index.js` now resolves `providers = { manual, intouch, kitovu }`.
- Credit top-ups via Kitovu settle through the same `settlePaid()` path as manual/intouch payments, with the ledger note `kitovu top-up verified`.
- The sender-name add-on and the integration service remain **manual only**. **No API v1 changes.**

**Security**

- The webhook is unsigned, so every callback is **cross-checked** against `GET /merchant-transaction/payment/:externalId` (3 bounded retries) before any settlement; the response is always a pure ACK `200 {"message":"success","success":true,"request_id":"<uuid>"}`. Terminal failures and amount mismatches are recorded in `meta.failure_reason`.
- The API secret only feeds the HMAC signature and is never transmitted; only ids/references are logged.
- Whole-RWF amounts are strictly matched at confirmation, so an off-by-one amount can never settle. See SECURITY.md §3.7/§5.20.

**Tests:** `npm test` → **164 unit tests**; `npm run test:integration` → **174 DB-backed tests**. New coverage: `test/unit/kitovu.test.js` (signature vector cross-validated against the PHP `hash_hmac` reference, compact JSON, timestamp-ms string, phone normalization, 6-digit externalId, webhook payload parse, status mappings, RWF math, description sanitization) and `test/integration/kitovu.test.js` (checkout → signed request → webhook → cross-check → settle, amount mismatch, sweep healing, duplicate-externalId retry, ACK contract).

## Client-facing PDF documentation — 2026-09-16

Client-facing PDF exports of this documentation set: a combined 94-page volume (cover + table of contents + 7 chapters) plus seven individual chapter PDFs, built from the markdown sources by an idempotent pandoc → weasyprint pipeline and served as static downloads at `/docs/pdf/`.

**Added**

- **Build pipeline** `docs/pdf-src/`. `build.sh` (idempotent — wipes and regenerates `pdf-src/build/` and all of `docs/pdf/`), `print.css` (print stylesheet, copied next to the generated HTML so the

pandoc `--css` link resolves inside `build/`), and `front-matter.md` (cover, document-control table, chapter map, and confidentiality note). Pipeline: pandoc `gfm → html5` (`--standalone`, `--syntax-highlighting=none`), weasyprint for PDF rendering, and poppler `pdfinfo` for the page-count summary.

- **Deliverables in `docs/pdf/`.** Combined `ZNotice-Complete-Documentation-<date>.pdf` (94 pages: cover + TOC + 7 chapters) plus 7 individual chapter PDFs: Overview 2p, Business Model 6p, API Reference 12p, Architecture 21p, Security 11p, Operations Runbook 20p, Changelog 14p (page counts from the 2026-09-14 build).
- **Static serving of `GET /docs/pdf/*.pdf`.** `express.static` route rooted at `config.paths.docsPdf` (overridable via `DOCS_PDF_DIR`), index disabled, dotfiles denied, and `Cache-Control: public, max-age=1d`. A "Documentation downloads (PDF)" section on the `/docs` page lists the files; names are derived by a strict filename whitelist in `server/services/pdfDocs.js`. Directory traversal is blocked by `express.static` / `send`. Plain links and downloads only — no CSP or security-policy changes required.
- **Test coverage.** New `test/unit/pdfDocs.test.js` (node:test) and `test/integration/docsPdf.test.js` (real HTTP over an ephemeral port using a `DOCS_PDF_DIR` fixture; asserts `200` + `application/pdf` for whitelisted files and `404` for non-matching names, dotfiles and encoded traversal).

**Changed**

- **`docs/pdf-src/build.sh` hardening.** `DATE` is now overridable via `$1` (defaults to today); the cover version line now interpolates `VERSION_LABEL` / `VERSION_DATE` instead of a hardcoded string.
- **`docs/README.md`.** The quick-facts table gained a PDF-docs row, and the intro now mentions the downloadable PDF set.
- **`docs/pdf-src/front-matter.md`.** Document-control dates moved to 2026-09-16 (`v0.1.0 (Phase 13, incl. follow-ups through 2026-09-16)`).

**Tests:** `npm test` → **113 unit tests**; `npm run test:integration` → **127 DB-backed tests**.

## Register QR share link — 2026-09-14

Adds a "scan to register" QR code that encodes a short link redirecting to the registration page, so the owner can print or share it anywhere ("scan to sign up"). Pure JS, no native dependencies and no NGINX/systemd changes.

**Added**

- **Dependency `qrcode`** (`^1.5.4`) as a runtime dependency. Installed as `www-data` so `node_modules/` / `package.json` / `package-lock.json` ownership stays consistent.
- **New service `server/services/qr.js`.** `renderPng(url)` returns a `Promise<Buffer>` (1024px PNG, margin 2, error-correction level M, black on white) and `renderSvg(url)` returns a `Promise<string>` (SVG, margin 2, level M). Both share the same quiet-zone/EC options.
- **Public routes in `server/routes/pages.js`** (same-origin, no auth):
  - `GET /join` → `302` redirect to `/register`.
  - `GET /join/qr` → renders the new `server/views/join-qr.ejs` share page (dark theme, `partials/head` + `partials/foot`, no inline scripts/styles).
  - `GET /join/qr.png` → `200 image/png` of `${APP_URL}/join` with `Cache-Control: public, max-age=86400`.

  - `GET /join/qr.svg` → `200 image/svg+xml; charset=utf-8` with the same cache header. Both image routes funnel render failures to `next(err)`.

- **New view `server/views/join-qr.ejs`.** 256px white/rounded QR image (`/join/qr.png`), the absolute short link (`joinUrl` from the route, derived from `config.appUrl` with trailing slashes stripped), an "Open registration" button linking to `/join`, and a print/share note. CSP-safe (no inline scripts or styles).
- **Landing CTA QR card** in `server/views/landing.ejs`: a compact, responsive card with a 128px QR image (`/join/qr.png`), the caption "Scan to create an account" and the link text `znotice.consultbofa.com/join`, using the existing Tailwind style language. `npm run build:css` recompiled `public/css/tailwind.css` so the new utility classes are present.

**Tests**

- `test/unit/qr.test.js` (node:test): `renderPng` resolves to a Buffer with the PNG magic bytes (`89 50 4E 47`); `renderSvg` resolves to a string containing `<svg`.
- `test/integration/joinQr.test.js`: real-HTTP over an ephemeral loopback port (`fetch`), following the `paymentProofs` pattern (`./helpers/db` first). Covers `/join` (302 + `Location: /register`), `/join/qr.png` (200, `image/png`, non-trivial body), `/join/qr.svg` (200, contains `<svg`), `/join/qr` (200 HTML referencing `/join/qr.png`) and `/` (200 landing HTML referencing `/join/qr.png`). No DB rows are created.

**Tests:** `npm test` → **106 unit tests**; `npm run test:integration` → **122 DB-backed tests** (both fully passing).

## Security hardening — Phase 12/13 follow-up — 2026-09-13

Targeted fixes from the Phase 12 (proof upload) / Phase 13 (billing lock) security review. No auth/CSRF/traversal flaws were found; these close resource and UX gaps.

- **H1 — strict multer limits.** `server/middleware/uploadProof.js` now sets `{ fileSize: MAX_PROOF_BYTES, files: 1, fields: 1, fieldSize: 1024, parts: 3 }` instead of only `fileSize`/`files`, so busboy can no longer buffer unbounded text fields/parts in memory before CSRF/auth. The upload form sends exactly one `_csrf` field + one file. The friendly multer error mapper moved next to the instance (`uploadProof.proofUploadErrorMessage`) and now covers `LIMIT_PART_COUNT`, `LIMIT_FIELD_COUNT`, `LIMIT_FIELD_KEY` and `LIMIT_FIELD_VALUE`; `server/index.js` still flashes + redirects for limit errors.
- **H2 — per-user proof storage quota.** `server/services/proofs.js` adds `MAX_PROOFS_PER_USER = 25` and `MAX_USER_PROOF_BYTES = 50 MB`, plus `getUserProofUsage(userId, executor?)` (single query joining `payment_proofs` to `transactions` on `user_id`) and `assertUserProofQuota(usage, incomingBytes)` (typed `user_limit_reached`, user-safe message). Enforced in `POST /dashboard/billing/proof/:id` inside the locked section before the file write; the per-transaction cap stays at 5.
- **H3 — locked users can order integration.** `/integration` was added to `ALLOWED_PREFIXES` in `server/services/billingAccess.js`; the integration request form is part of the payment flow (it creates a pending $20 payment) and is shown on the Billing page, so locked users may reach it.
- **H4 — row lock no longer held across disk I/O.** The proof route now does begin → `SELECT ... FOR UPDATE` → per-tx cap → per-user quota → `INSERT` → commit → release, then writes the file outside the transaction. A write failure best-effort deletes the committed row and any partial file. No code path deletes a file after a successful commit (the old catch-all compensation could fire on a release error). A

brief window where the row exists before the file is written is acceptable — serving returns 404 until the write completes.

**Tests:** `npm test` → **104 unit tests**; `npm run test:integration` → **117 DB-backed tests**. New coverage: strict multer limits + error mapping, per-user quota (count + bytes), write-failure compensation, the admin proof route auth guard over real HTTP (non-admin → 403, anonymous → login redirect), and a real-multipart CSRF composition test (session stub → multer → real `verifyCsrf`; valid token passes, missing/wrong → 403, extra fields → friendly rejection).

## Phase 13 — Billing lock for unpaid users + role policy — 2026-09-13

### Added

- **Billing lock for unpaid users.** A non-admin account is confined to the **Billing** and **Account** pages until an admin confirms its first payment. New middleware `server/middleware/paymentLock.js` (`requirePayment`) is mounted at `/dashboard` in `server/index.js` immediately after `verifyCsrf` and before every dashboard router. Anonymous requests pass through (the downstream `requireAuth` owns the login redirect); admins always pass. Locked users hitting any other `/dashboard/*` page get an error flash and a `302` to the pending transaction's pay page (`/dashboard/billing/pay/<id>`) or, when there is no pending transaction, to `/dashboard/billing`.
- **New service `server/services/billingAccess.js`.** `getBillingAccess(userId)` resolves `{ unlocked, pendingTransactionId }` with one query (`EXISTS(paid transaction)` + `COALESCE(SUM(credit_ledger.delta), 0)`) and only looks up the most recent `pending` transaction when locked. Pure helpers `isAllowedBillingPath` (Billing/Account prefixes; non-strings rejected) and `billingRedirectTarget` are exported for unit tests.
- **Locked-state UX.** `server/views/partials/dashnav.ejs` renders only the Billing and Account tabs (plus a "Locked — payment required" indicator) when `res.locals.billingLocked` is true; `server/views/dashboard/billing.ejs` shows an info banner explaining that the account unlocks after an admin confirms the first payment.
- **Admin proof visibility.** The Admin overview pending-payments query (`server/routes/admin.js`) now returns `(SELECT COUNT(*) FROM payment_proofs pp WHERE pp.transaction_id = t.id) AS proof_count` and `server/views/admin/index.ejs` renders an "N proof(s)" badge (or "No proof") so admins can spot transactions with uploaded proof before confirming.

### Changed

- `server/index.js` mounts `app.use('/dashboard', requirePayment)` immediately after `app.use(verifyCsrf)` and before the first dashboard router. The Phase 12 proof-upload multer mount stays before CSRF, so multipart uploads still parse `_csrf`; the lock allows every `/billing/*` path (including proof upload/view/delete, checkout, cancel and sender-name) plus `/account/*` and `/integration/*` (the integration request form starts a pending payment).
- Registration is unchanged: `POST /register` hardcodes `role = 'user'` and a `role` field in the request body is ignored. A regression test now pins this.

### Security

- The lock is enforced **before** every dashboard router, so a locked user cannot reach Overview, WhatsApp, API keys, Messages, Scheduled, Templates, Analytics or Webhooks — even by direct URL. Billing/Account paths always pass through, so there is no redirect loop.
- `isAllowedBillingPath` only ever returns `true` for string paths starting with `/billing` or `/account`; non-string input can only lock down.

- The lock is dashboard-only: `/api/v1/*` is unchanged (API sends already fail with `402 insufficient_credits` at a zero balance) and `/admin/*` is unaffected.
- Existing accounts are promoted to `admin` by the deploy agent; production promotion is a one-time SQL step outside this change set.

### Tests

- `npm test` → **97 unit tests** (`test/unit/billingAccess.test.js` added).
- `npm run test:integration` → **110 DB-backed tests** (`test/integration/billingAccess.test.js`, `test/integration/paymentLock.test.js` added; the admin overview proof-count query is covered in `test/integration/paymentProofs.test.js`).
- `paymentLock.test.js` drives the middleware with fake req/res (locked → `302` + flash; Billing/Account → `next()` + `billingLocked`; paid/positive balance/admin/anonymous → `next()`; DB error → `next(err)`) and verifies a client-supplied `role: 'admin'` at registration is ignored.

## Phase 12 — Payment proof uploads — 2026-09-13

### Added

- **Payment proof uploads.** After choosing an amount/method and confirming a manual payment, the customer now sees a **Payment proof** card on the pay page (`server/views/dashboard/pay.ejs`) with instructions and an upload form (`POST /dashboard/billing/proof/:id`, multipart). Files are stored under `<uploads>/payment-proofs/<transaction_id>/<random16hex>.<ext>`; a row is written to the new `payment_proofs` table (migration `server/migrations/012_payment_proofs.sql`). Up to **5 proofs per transaction**, **5 MB** each, types **JPG/PNG/WebP/PDF**.
- **New service `server/services/proofs.js`:** `sniffMime` (magic bytes), `safeOriginalName` (control chars stripped, path dropped, 255-char cap, `proof` fallback), `makeStoredName` (`<16 hex>.<ext>`), `validateProofBuffer`, `assertProofLimit`, and filesystem helpers (`writeProof`, `proofPath`, `proofExists`, `deleteProofFile`) rooted at `config.paths.uploads` (read at call time). Typed errors (`invalid_type`, `too_large`, `limit_reached`) carry user-safe messages.
- **New middleware `server/middleware/uploadProof.js`:** a `multer@^2` instance with `memoryStorage()` and `limits { fileSize: 5 MB, files: 1, fields: 1, fieldSize: 1024, parts: 3 }` (dependency added: `multer`; the stricter field/part limits were added in the Phase 12/13 security hardening above).
- **Authenticated proof serving.** Owners stream their own proofs at `GET /dashboard/billing/proof/:id/:proofId`; admins stream any proof at `GET /admin/transactions/:id/proof/:proofId`. Both send `Content-Disposition: inline; filename="proof-<id>.<ext>"`, `Cache-Control: private, no-store` and `Content-Security-Policy: default-src 'none'; sandbox`; the MIME type comes from the DB row (byte-sniffed). A missing or foreign proof is always a 404.
- **Owner delete while pending** (`POST /dashboard/billing/proof/:id/:proofId/delete`) removes the row and best-effort unlinks the file. Proofs remain readable by owner + admin after `paid` / `failed` for audit.
- **Admin visibility.** `GET /admin/transactions` now attaches each transaction's proofs (single `WHERE transaction_id IN (...)` query) and the table renders a "Proofs (N)" column linking each proof in a new tab.
- **Config:** `UPLOADS_DIR` env override for `config.paths.uploads`.

### Security

- File type is decided by **magic bytes only** — the client MIME/filename is never trusted. JPEG (`FF D8 FF`), PNG (`89 50 4E 47 0D 0A 1A 0A`), WebP (`RIFF....WEBP`) and PDF (`%PDF-`) are accepted; everything else is rejected.
- Stored filenames are generated (`crypto.randomBytes(8)` hex) and validated against a strict pattern before any path join — no user-controlled paths, no traversal. Directories are created `0o750`, files written `0o640`.
- Uploads are `memoryStorage` only: nothing hits disk until CSRF, ownership, status, size and bytes have been validated. The DB row is committed first and the file is written afterwards (outside the row lock, H4); if the write fails the committed row (and any partial file) is best-effort removed.
- **CSRF preserved for multipart:** `server/index.js` mounts the multer upload (and a rate limiter) for `/dashboard/billing/proof` **immediately before** `app.use(verifyCsrf)`, so the hidden `_csrf` field is parsed and the global CSRF check still applies. Non-multipart requests pass straight through.
- Upload POSTs are rate-limited (~30 per 15 minutes, keyed by `req.user.id` else IP). Size (5 MB), per-transaction count (5) and per-user storage (25 proofs / 50 MB) caps are enforced in the service inside a transaction that locks the transaction row (`FOR UPDATE`).
- Proofs are served only through authenticated Node routes (`uploads/` is outside `public/`); foreign/missing proofs 404 without leaking existence.

### Tests

- `npm test` → **91 unit tests** (`test/unit/proofs.test.js` added).
- `npm run test:integration` → **89 DB-backed tests** (`test/integration/paymentProofs.test.js` added; `test/integration/helpers/db.js` now points `UPLOADS_DIR` at a wiped temp dir and truncates `payment_proofs`).

## Phase 11 — Admin sender-name override & message templates — 2026-09-12

### Added

- **Admin sender-name override.** Platform admins can view and set/override a tenant's WhatsApp sender (profile) name from **Admin → Sessions** — a new "Sender name" column shows `wa_sessions.profile_name` plus an inline form. `POST /admin/sessions/:id/sender-name` (`server/routes/admin.js`, `server/views/admin/sessions.ejs`) is admin-only, CSRF-protected, validates with the shared `validateSenderName` (1–25 chars, control chars/line breaks rejected), requires the session to be `connected`, calls the engine `setProfileName`, and persists `wa_sessions.profile_name`. Raw engine error detail is logged server-side only and never surfaced to the admin UI.
- **Deliberate entitlement bypass (support tooling).** The customer route (`server/routes/whatsapp.js`) requires both a connected session **and** the paid add-on (`users.sender_name_until`); the admin route intentionally does **not** require the add-on. This is the **only** bypass.
- **Message templates.** New table `message_templates` (migration `server/migrations/011_message_templates.sql`): `id`, `user_id` FK `ON DELETE CASCADE`, `name` VARCHAR(80), `body` TEXT, `media_url` VARCHAR(500), timestamps, `UNIQUE(user_id, name)`.
- New service `server/services/templates.js`: `validateTemplate` (name ≤80, body ≤4096, `http(s)` media URL ≤500, control characters rejected), user-scoped CRUD, strict `parseTemplateId`, `resolvePrefill` (ownership-checked; returns `null` for foreign/invalid/missing ids), `listForUser`

(`LIMIT 500`), and an **atomic per-user cap** (`templates_max_per_user`, default 200) using a user-row lock; typed `duplicate_name` / `limit_reached` errors.
- New dashboard page `/dashboard/templates` (`server/routes/templates.js`, `server/views/dashboard/templates.ejs`): create, inline `<details>` edit (no JS) and delete; a "Templates" tab was added to the dashboard nav (`server/views/partials/dashnav.ejs`).
- Server-side prefill (no JS): `GET /dashboard/whatsapp?template=<id>` and `GET /dashboard/scheduled?template=<id>` pre-fill the message body + media URL from an owned template (`server/routes/dashboard.js`, `server/routes/scheduled.js`).
- Admin setting `templates_max_per_user` (int 1–10000, default 200) in **Admin → Settings** (`server/routes/admin.js`).

### Changed

- `validateSenderName` moved to `server/services/senderName.js` (alongside the add-on entitlement helper) and re-exported from `server/routes/whatsapp.js`; the existing unit test is unchanged.
- Templates are dashboard-only — **no API v1 changes**. `{{name}}` / `{{phone}}` placeholders in a prefilled message are still resolved at send time by the existing scheduler/bulk pipeline (unchanged).

### Fixed

- Security-review findings fixed before deploy: **P11-01** unbounded template creation → atomic per-user cap (Medium); **P11-02** unbounded template list → `LIMIT 500` (Low); **P11-03** strict route id parsing; **P11-04** admin route safe-integer id; **P11-05** `mediaUrl` control-character rejection.

### Tests

- `npm test` → **76 unit tests** (`test/unit/templates.test.js` added).
- `npm run test:integration` → **75 DB-backed tests** (`test/integration/templates.test.js`, `test/integration/adminSenderName.test.js` added).

## Phase 10 — Dedicated sender name add-on ($10 / 30 days) — 2026-09-11

### Added

- Paid add-on gating the Phase 9 sender-name feature: **$10.00 / 30 days** by default, bought through the existing **manual payment + admin approval** flow. Entitlement is stored on `users.sender_name_until` (UTC) and resolved by `server/services/senderName.js` (`isSenderNameActive`).
- Dashboard Billing card and `POST /dashboard/billing/sender-name` (`server/routes/dashboard.js`, `server/views/dashboard/billing.ejs`): creates a `pending` transaction with `credits = 0` and `meta = { type: 'sender_name', days }`, then redirects to the standard payment page. `server/views/dashboard/pay.ejs` labels the transaction and period.
- Settlement extension (`server/services/payments/index.js`): approving a `meta.type='sender_name'` transaction sets `sender_name_until = DATE_ADD(GREATEST(COALESCE(sender_name_until, NOW()), NOW()), INTERVAL ? DAY)`, so a **renewal extends from the current expiry** and paid time is never lost. `meta.days` is clamped to **1..365**; an unset/unparseable value falls back to the configured `sender_name_days`.
- Migration `server/migrations/010_sender_name_addon.sql`: `users.sender_name_until DATETIME NULL` plus settings seeds `sender_name_price_cents = 1000` and `sender_name_days = 30`.
- Admin settings (`server/routes/admin.js`): `sender_name_price_cents` (int, 1–100000000) and `sender_name_days` (int, 1–365) are editable.

- Tests: `test/integration/senderNameAddon.test.js` — settlement/renewal, `days` clamping, entitlement-update rollback + retry, the connected/entitled gate, and concurrent checkout (exactly one pending transaction).

### Changed

- `payments.manual.createCheckout` accepts an optional `conn` so the dashboard sender-name route can insert the pending transaction inside its own transaction. Existing callers are unchanged (defaults to the shared pool).
- The WhatsApp sender-name card and the Billing card show the add-on state and a Billing CTA when inactive; the WhatsApp form is disabled until the add-on is active and the session is `connected`.
- `/pricing` advertises the add-on next to message credits (published default list price **$10.00 / 30 days**).
- **Message pricing is unchanged: 10 = 1,000*credits*(0.01/message).** Attachments/media remain available to all users and are **not** gated by the add-on.

### Fixed

- **Settlement rollback safety:** if the entitlement `UPDATE` throws after the `pending → paid` transition, the transaction is reverted to `pending` (`paid_at = NULL`) and the error is rethrown, so an admin can retry and no payment is lost (mirrors the existing credit-add compensation).
- **Atomic duplicate-checkout guard:** `POST /dashboard/billing/sender-name` now locks the user row (`SELECT ... FOR UPDATE`), re-checks for an existing pending `sender_name` transaction, and inserts in the **same** transaction. The previous non-atomic SELECT-then-INSERT allowed N concurrent requests to each create a pending transaction (TOCTOU).
- **`days` clamp correctness:** the entitlement extension previously used `parseInt(meta.days) || fallback`, which treated a legitimate `days: 0` as unset. The code now uses an explicit finite check and clamps to `1..365`.
- Concurrent admin approvals cannot double-extend: the atomic `pending → paid` gate means exactly one winner extends the entitlement.

## Phase 9 — WhatsApp sender name — 2026-09-11

### Added

- Dashboard sender-name card on `/dashboard/whatsapp` and `POST /dashboard/whatsapp/sender-name` (`server/routes/whatsapp.js`, `server/views/dashboard/whatsapp.ejs`): customers can set the WhatsApp profile/push name recipients see. The form is enabled only while the session is `connected` (and, since Phase 10, while the add-on is active).
- Engine call `wpp.setProfileName(sessionName, name)` (`server/services/wppconnect.js`) → `POST /api/{session}/change-username` with `{ name }`, which the engine maps to `client.setProfileName(name)`.
- Pure `validateSenderName()` helper (exported for tests): control characters and line breaks rejected; whitespace runs collapsed; leading/trailing whitespace trimmed; result must be **1–25 characters**.
- Migration `server/migrations/009_sender_name.sql`: `wa_sessions.profile_name VARCHAR(60) NULL` — the last name pushed through ZNotice (`NULL` = never set here). `waSessions.update()` persists `profileName`.
- Tests: `test/unit/senderName.test.js` (validation) and `test/integration/senderName.test.js` (engine call + persistence and every failure path leaving `profile_name` untouched).

### Changed

- The sender-name card states the WhatsApp profile-name caveat: changing the name **also changes it on the customer's phone**, and WhatsApp limits how often a profile name can be changed.

### Notes

- Dashboard-only; **no API v1 changes**.
- The engine is called only after the session-connected check (and the Phase 10 entitlement gate), and engine error detail is logged server-side, never shown to the customer.

## Phase 8 — Account recovery, hardening & test suite — 2026-09-11

### Added

- Self-service password reset (web UI, dashboard only — **no API changes**): `GET/POST /forgot-password` and `GET/POST /reset-password` (`server/routes/auth.js`, `server/views/auth/forgot-password.ejs`, `server/views/auth/reset-password.ejs`). The forgot response is always the same (`302 /forgot-password?sent=1`) whether or not the account exists, so it cannot be used to enumerate users. Both endpoints are CSRF-protected and rate-limited per IP: **5 / 15 min** (forgot) and **10 / 15 min** (reset).
- Reset-token lifecycle: 32 random bytes (`crypto.randomBytes(32).toString('hex')`, 64 hex chars) with **only `sha256(token)` stored**; a 30-minute expiry; and single-use enforced by an **atomic conditional claim** inside the password-change transaction (`UPDATE password_resets SET used_at=NOW() WHERE id=? AND used_at IS NULL AND expires_at>NOW()`; `affectedRows!==1` → invalid link). Requesting a new link deletes any prior unused token. A successful reset changes the password, deletes every session for the user, and sends a password-changed notification (fire-and-forget).
- Outbound email service (`server/services/email.js`, nodemailer **10.0.8**): env-driven SMTP config (`SMTP_HOST`, `SMTP_PORT` default `587`, `SMTP_SECURE`, `SMTP_USER`, `SMTP_PASS`, `SMTP_FROM`), reused transport, `passwordResetEmail()` / `passwordChangedEmail()` templates, and a `send()` that **never throws** and never logs credentials, tokens or message bodies. With no `SMTP_HOST` it returns `{ok:false, error:'email_not_configured'}` and the UI still shows the generic "check your email" message.
- Migration `server/migrations/008_password_resets.sql`: `password_resets` (`token_hash CHAR(64)` unique, `expires_at`, `used_at`, FK `user_id` → `users` `ON DELETE CASCADE`). Defensive `CREATE TABLE IF NOT EXISTS`.
- Automated test suite (Node's built-in `node:test`, **no new dependencies**):
  - `npm test` → **57 unit tests** (`test/unit/`: `send`, `schedule`, `webhook`, `netguard`, `analytics`, `safeNext`), no database required.
  - `npm run test:integration` → **36 DB-backed tests** (`test/integration/`: `credits`, `schedule`, `scheduler`, `passwordReset`, `reaper`).
  - `npm run test:all` runs both.
  - `test/integration/helpers/db.js` refuses to run against production (`NODE_ENV!==production`, `TEST_DB_NAME` must end in `_test` and differ from the `.env` `DB_NAME`), forces the isolated test DB before the app loads, and serializes integration files with a MariaDB advisory lock (`GET_LOCK`).

### Changed

- **Lazy CSRF token** (`server/middleware/csrf.js`): `attachCsrf` now materializes `res.locals.csrfToken` only when a view actually reads it, and `needsCsrf(req)` mints a token only

for authenticated requests or the auth form pages (`/login`, `/register`, `/forgot-password`, `/reset-password`). Public pages no longer create `sessions` rows (previously an enumerable `res.locals` getter forced `express-mysql-session` to persist one on every render).
- **Session reaper pruning** (`server/services/reaper.js`): a new pass deletes expired `sessions` rows and anonymous sessions older than **7 days** (`sessionsDeleted` in the summary). `sessions.expires` is stored by `express-mysql-session` in UNIX **seconds**, not milliseconds.
- **No-store on private responses**: `Cache-Control: no-store` is set on authenticated HTML responses (`server/index.js`) and on **all** `/api/v1` responses including 401/403 (`server/routes/api/v1.js`).
- **Log redaction** (`server/index.js`): the global error log now redacts the `token`, `secret`, `key` and `password` query parameters (case-insensitive), so a reset token or secret can never reach the journal.
- **SMTP transport rotation** (`server/services/email.js`): the transport cache key includes a short non-reversible password fingerprint, so a rotated `SMTP_PASS` rebuilds the transport instead of silently reusing old credentials.
- **Host hardening (item E)** applied by `@devops`; full record and rollback in `docs/OPERATIONS.md` §14 and `/root/backups/host-hardening-2026-09-11/CHANGES-AND-ROLLBACK.md`:
  - UFW stale allows for **3001/tcp, 3005/tcp and 4000/tcp** (v4 + v6) removed; the GameBet `next-server` on 3001 stays reachable through NGINX on loopback, direct external access is denied. Rollback: `ufw allow <port>/tcp`.
  - Sudoers hygiene: duplicate `kali ALL=(ALL) NOPASSWD:ALL` and the empty-group `kali-grant-root` file removed; `kali` passwordless sudo retained; `visudo -c` clean.
  - NGINX `access_log off` for the exact `location = /reset-password` so reset tokens in the query string never reach `access.log`.

**Fixed**

- **F1 — reset-token race (single-use):** the token is claimed with a conditional `UPDATE` *inside* the password-change transaction, so two concurrent submissions of the same token yield exactly one winner and one `400` invalid-link response. Regression test: `test/integration/passwordReset.test.js` ("two parallel consumptions of one token yield exactly one winner").
- **F2 — `safeNext` open redirect:** the post-login/registration redirect target now blocks protocol-relative (`//host`), dot-segment (`/..//host`, `/%2e%2e//host`), raw/encoded backslash (`\`, `%5c`) and encoded-separator (`%2f` / `%5c`, including multi-encoded) variants, and only ever returns a same-origin path/search/hash. `safeNext` is exported for unit tests (`test/unit/safeNext.test.js`).
- **F3 — register 500 on long email:** an email longer than `VARCHAR(190)` is now rejected with a clean `400` (was a `500` "Data too long").
- **F5 — error-log redaction** (above).
- **F6 — session pruning** (above).
- **F7 — SMTP transport rebuild on password rotation** (above).
- **F8 — maximum password length:** passwords over **72 bytes** (the bcrypt truncation point) are rejected on registration, reset and account password change, including multi-byte input.
- One-time cleanup removed ~1,300 anonymous `sessions` rows created by the pre-Phase-8 CSRF behaviour.

**Open (documented, not fixed)**

- `/register` still returns `409` for an existing email — a pre-existing user enumeration vector (see `docs/SECURITY.md` §5.13).
- `POST /forgot-password` response time differs by ~15 ms for existing vs non-existing accounts; status and body are identical (see §5.14).

**Post-review configuration (2026-09-11):** production SMTP is configured using the existing Hostinger mailbox (`info@consultbofa.com`, `smtp.hostinger.com:465`, implicit TLS) and verified with a live test send; see `docs/OPERATIONS.md` §12 and `docs/SECURITY.md` §5.15.

## Phase 7 — Usage analytics — 2026-09-11

### Added

- Read-only analytics service (`server/services/analytics.js`): `customerDaily`, `customerSummary`, `adminDaily`, `adminSummary` and `topUsers`. Pure `SELECT` aggregations over `messages`, `credit_ledger`, `transactions` and `users`; no writes, no cache, no background worker.
- Customer analytics page `/dashboard/analytics` (`server/routes/analytics.js`, `server/views/dashboard/analytics.ejs`, fixed 30-day window): summary cards (messages today / last 7d / last 30d, 30-day success rate, 30-day credits spent), messages-per-day and credits-spent-per-day charts, a sent/delivered/read delivery funnel, and the top 5 failure reasons (last 30d, blank errors normalized to `unknown`), with a friendly empty state. New **Analytics** tab in the dashboard nav after Overview.
- Admin analytics page `/admin/analytics` (admin only; `server/routes/admin.js`, `server/views/admin/analytics.ejs`, fixed 30-day window): summary cards (revenue from paid transactions, credits sold, messages, active users, credits outstanding, pending payments), revenue-per-day and messages-per-day charts, and the top 10 users by message volume (name / email / messages / credits spent / last activity). New **Analytics** tab in the admin nav after Overview.
- Customer API `GET /api/v1/usage?days=N` (`server/routes/api/v1.js`): API-key auth; `days` default 30, clamped 1–90. Returns `{ok, days, data[], totals}` with exactly `days` zero-filled UTC day points ending today and a `success_rate` `0`–`1` ratio.
- Reusable server-rendered inline SVG bar chart (`server/views/partials/svg-bars.ejs`): single or grouped bars, accessible (`role="img"`, `<title>` tooltips), no chart library, presentation attributes only — CSP-safe (no inline scripts or style attributes).
- Migration `server/migrations/007_analytics_indexes.sql`: index-only (`messages(created_at)`, `transactions(status, paid_at)`), idempotent and `information_schema`-guarded so it is safe to re-run.
- Docs: analytics pages, the `/api/v1/usage` endpoint, metric semantics, the new indexes and a runbook section added to `docs/API.md`, `docs/ARCHITECTURE.md`, `docs/OPERATIONS.md`, `docs/SECURITY.md` and the customer `/docs` page.

### Changed

- Dashboard and admin navigation each add an **Analytics** tab immediately after Overview.
- `server/index.js` mounts the customer analytics router at `/dashboard/analytics` (before the `/dashboard` catch-all).

### Fixed

- **Queued-credit consistency:** every `credits_spent` aggregate sums `credits_charged` only for completed statuses (`sent` / `delivered` / `read` / `failed`). `queued` rows are excluded from customer counts and from all credit sums, so analytics never report credits that were not actually charged.
- **Defense-in-depth input clamp:** `days` is clamped to `1..90` both in the HTTP routes and inside the service (`clampDays`), and `topUsers.limit` is clamped to `1..50` in the service, so a direct service call can never produce an `INTERVAL NaN DAY` SQL error or allocate an unbounded day-key array.

- Security review outcome: **GO**. One low-severity follow-up recorded as **L2** — authenticated HTML/API responses lack `Cache-Control: no-store` (browser-local cache/bfcache only); see `docs/SECURITY.md` §5.12.

## Phase 6 — Scheduled & bulk sending — 2026-09-11

### Added

- Scheduled send: `POST /api/v1/send` accepts an optional `scheduled_at` (ISO-8601 with `Z`/offset, or naive `YYYY-MM-DDTHH:MM` / `YYYY-MM-DD HH:MM` interpreted as UTC). Omitted/blank keeps the immediate path; a valid future value enqueues and returns `202 {ok, scheduled_id, scheduled_at, credits_balance}`. Window: now+60s … now+`schedule_max_days_ahead` (default 90) days. Enqueue prechecks: active account (`403 account_suspended`), connected WhatsApp session (`409 whatsapp_not_connected`), balance ≥ item count (`402 insufficient_credits`) and the per-user pending cap (`429 queue_full`).
- Bulk send: `POST /api/v1/bulk` (`{message, media_url?, scheduled_at?, label?, recipients:[{to,name?}]}`), 1–`bulk_max_recipients` (default 1000) recipients, normalized + deduped (response `duplicates_removed`); invalid entries reject the request with `400 invalid_recipients` (up to 10 `{line,value}`). `{{name}}` / `{{phone}}` are resolved per recipient at enqueue; unknown tokens are left untouched. Returns `202 {ok, batch_id, total, duplicates_removed, scheduled_at, credits_balance}`.
- Queue management API: `GET /scheduled` (default pending+sending, whitelisted `?status=`, `?limit=` 1–200), `GET /scheduled/:id`, `DELETE /scheduled/:id` (pending only; `409 not_pending`), `GET /batches` (`?limit=` 1–100), `GET /batches/:id` (batch + items, `?limit=` 1–500) and `DELETE /batches/:id` (cancels remaining pending items; `409 not_cancellable` when terminal).
- Scheduler worker (`server/services/scheduler.js`): every 30s plus a kick on enqueue. Claims due `pending → sending` atomically (at-most-once), enforces a per-user rate limit shared with `send_rate_per_min`, and retries only retryable failures (409 not connected, engine unreachable, engine HTTP 5xx) with `attempts++` and +1 min/+5 min backoff, terminal after 3 attempts. Credits are charged per message at actual send time and auto-refunded on engine failure; there is no reservation at enqueue.
- Scheduled/bulk service (`server/services/schedule.js`): `parseScheduledAt`, `resolveTemplate`, `parseRecipientLines`, `normalizeRecipients`, create/list/ cancel helpers, and atomic pending-cap enforcement via a user-row lock.
- Batch completion webhook: `batch.completed` is emitted **once**, only by the scheduler, when a batch finishes with zero cancellations. Fully-cancelled batches become `cancelled` silently; partially-cancelled batches become `completed` silently.
- Dashboard scheduled queue (`server/routes/scheduled.js`, `server/views/dashboard/scheduled.ejs`, `server/views/dashboard/scheduled-batch.ejs`): create form (message, media URL, datetime-local UTC, recipients textarea `phone` or `phone,name`), open queue (pending + sending), batch list, paginated batch detail (100/page), and cancel buttons (pending items only). CSRF-protected; no inline JS.
- Migrations `server/migrations/005_scheduled_bulk.sql` (`send_batches`, `scheduled_messages`, Phase 6 settings seeds) and `server/migrations/006_scheduler_claim.sql` (adds the `sending` claim state).
- Admin settings: `bulk_max_recipients`, `schedule_max_days_ahead` and `scheduled_max_pending` are editable in the settings editor.

- Docs: scheduled/bulk endpoints, the scheduler/queue flow, runbook entries and security controls added to `docs/API.md`, `docs/ARCHITECTURE.md`, `docs/OPERATIONS.md`, `docs/SECURITY.md` and the customer `/docs` page.

### Changed

- `server/services/reaper.js` now resets scheduler claims stuck in `sending` for >10 min to `failed` / `interrupted_by_restart` (bumping the batch `failed_count` and reconciling silently), and prunes terminal `scheduled_messages` and `send_batches` older than 90 days. The summary includes `sendingInterrupted`.
- `server/services/send.js` validates `media_url` at a maximum of 500 characters (shared by immediate, scheduled and bulk sends) and exports `MAX_MEDIA_URL_CHARS`.
- `server/index.js` starts the scheduler alongside the reaper and webhook worker; `server/routes/api/v1.js` routes a non-empty `scheduled_at` through the queue service.

### Fixed

- Concurrency hardening (security-review follow-ups):
  - **Atomic batch completion:** `maybeCompleteBatch()` is a single conditional UPDATE guarded by `status IN ('scheduled','processing')` and `NOT EXISTS(open item)`, so concurrent callers race on the row and exactly one wins; `batch.completed` cannot be emitted twice.
  - **Atomic pending cap:** enqueue serializes the cap check + insert on the user row (`SELECT ... FOR UPDATE`), closing a TOCTOU that could overrun `scheduled_max_pending`.
  - **Claim pattern:** `pending → sending` before the engine call, so a cancel cannot race an in-flight send and a crash cannot duplicate a message.
  - **`media_url` length validation** prevents an over-long URL from surfacing as an unhandled "Data too long for column" 500.
  - **Scheduler fairness:** `ROW_NUMBER() ... PARTITION BY user_id` caps each user at 25 items per tick (global LIMIT 50), so one tenant can no longer consume a whole tick.
  - **Suspended-account 500:** account status is checked before the WhatsApp session lookup in the send/enqueue paths, so a suspended user receives a clean `403 account_suspended` instead of a 500.

## Phase 5 — Account settings, session reaper & outbound webhooks — 2026-09-11

### Added

- Account settings page (`server/routes/account.js`, `server/views/dashboard/account.ejs`): editable display name, read-only email, and a password change flow (bcrypt cost 12) that invalidates all other sessions while keeping the current one signed in. Password changes are rate-limited (5 / 15 min / IP+user) and logged without any password values.
- Stale-session TTL reaper (`server/services/reaper.js`, started from `server/index.js`): frees abandoned `pending` sessions (>10 min) and closed `connected` sessions, and prunes `webhook_deliveries` older than 30 days. Exposes `runOnce()` for scripts; no HTTP trigger.
- Outbound customer webhooks:
  - `server/services/netGuard.js`: SSRF egress guard extracted from `mediaFetch.js` (URL/IP validation + DNS pinning), reused by webhooks.

    - `server/services/webhookOut.js`: durable delivery queue with HMAC-SHA256 signatures (`X-ZNotice-Signature`), 10s timeout, +1 min/+5 min backoff, and terminal failure after 3 attempts.
    - `server/routes/webhookConfig.js` + `server/views/dashboard/webhooks.ejs`: configure/validate the endpoint, enable/disable, rotate the secret, send a test event, and view the last 25 deliveries.
    - Events emitted: `message.sent`, `message.failed` (send pipeline), `message.delivered`, `message.read` (engine ack transitions, no duplicates), and `test.ping`.
- Migration `server/migrations/004_outbound_webhooks.sql`: `user_webhooks` (one endpoint per user) and `webhook_deliveries` (queue + log).
- Docs: outbound webhook contract + signature verification in `docs/API.md` and `/docs`; reaper/webhook troubleshooting in `docs/OPERATIONS.md`; SSRF guard note in `docs/SECURITY.md`.

**Changed**

- `server/services/mediaFetch.js` now delegates URL/IP validation and DNS pinning to `netGuard.js` (behaviour and error messages unchanged).

## Phase 4 — Operations hardening & documentation — 2026-09-10

Backups, health monitoring, log rotation, and this documentation set.

**Added**

- `znotice-backup.sh` (`/usr/local/bin/znotice-backup.sh`): daily MariaDB dump (`--single-transaction --routines --triggers`, gzip-verified) plus an app + secrets archive. Retains 14 DB dumps / 7 app archives; credentials never printed; backups stored `0640 root:www-data`.
- `znotice-health.sh` (`/usr/local/bin/znotice-health.sh`): 5-minute probe of systemd state, public HTTPS (pinned to loopback), engine reachability, and available RAM. Auto-restarts the app only; never the engine.
- Cron schedules: `/etc/cron.d/znotice-backup` (02:30 daily), `/etc/cron.d/znotice-health` (every 5 min).
- `/etc/logrotate.d/znotice`: weekly rotation, 8 copies, compressed.
- Documentation set in `docs/`: `README.md`, `ARCHITECTURE.md`, `OPERATIONS.md`, `API.md`, `BUSINESS.md`, `SECURITY.md`, `CHANGELOG.md`.

**Infrastructure**

- NGINX vhost with Let's Encrypt TLS, ACME webroot challenge, security headers, and loopback proxy to `127.0.0.1:5071`.
- `znotice.service` systemd unit (user `www-data`, `Restart=always`, `EnvironmentFile=.env`).
- WPPConnect engine container `wppconnect` (v2.10.10) bound to `127.0.0.1:21465`.

## Phase 3 — Manual payments, integration orders, admin panel, capacity guard — 2026-09-10

**Added**

- Payment adapter registry: `server/services/payments/index.js` with the `manual` provider, `ZN-XXXXXXXX` reference generation, idempotent `settlePaid()` / `settleFailed()`, and instruction templating.

- Manual credit top-up: `POST /dashboard/billing/checkout`, `GET /dashboard/billing/pay/:id`, `POST /dashboard/billing/cancel/:id`.
- Integration service orders ($20): `POST /dashboard/integration/request`, order lifecycle `requested → paid → in_progress → delivered` (+ `cancelled`).
- Full admin panel (`server/routes/admin.js`): overview stats, user suspend/activate, credit adjustments, payment approval queue, integration order management, session management with force-disconnect, and editable settings.
- Capacity guard in `server/routes/whatsapp.js` using `max_concurrent_sessions` (default 15), counting `pending` + `connected` sessions while allowing owners mid-pairing.
- Migration `server/migrations/003_phase3.sql`: unique `transactions.provider_ref`, `integration_orders.status` gains `paid`, `integration_orders.transaction_id` FK, and new settings seeds.
- Admin views: `server/views/admin/*.ejs`.

## Phase 2 — QR connect, send pipeline, webhooks, terms & admin basics — 2026-09-09

### Added

- WhatsApp connection flow (`server/routes/whatsapp.js`): `POST /connect`, `GET /qr`, `POST /disconnect`; WPPConnect client wrapper (`server/services/wppconnect.js`) with per-session bcrypt bearer tokens.
- Outbound send pipeline (`server/services/send.js`): validation, phone normalization, per-key rate limiting, atomic credit charge, engine send, failure persistence + automatic refund.
- Credit accounting (`server/services/credits.js`): ledger-based balance with row-locked atomic `chargeCredits` / `addCredits`.
- Inbound delivery webhooks (`server/routes/webhooks.js`): `onack` status progression (monotonic), `qrcode`, `status-find`, `closesession`, `logoutsession`; secret path token / header auth.
- Dashboard pages: overview, WhatsApp, API keys (create/revoke), messages (paginated), send test, billing.
- API-key management (`server/routes/apiKeys.js`) and API v1 (`server/routes/api/v1.js`): `GET /status`, `GET /messages`, `POST /send`.
- Terms & Acceptable Use page (`server/views/terms.ejs`) with the WhatsApp ban-risk disclosure; public pricing and docs pages.
- Admin basics and `settings` service with 60s cache.
- Migration `server/migrations/002_phase2.sql`: `users.accepted_terms_at`, `messages(user_id,status)` index, `default_country_code` and `send_rate_per_min` settings.

## Phase 1 — Scaffold & infrastructure — 2026-09-08

### Added

- Node.js/Express 4 application skeleton: `server/index.js`, `server/config.js`, `server/db.js` (connection pool + idempotent migration runner), `server/migrate.js`.
- MariaDB-backed sessions, helmet CSP, rate limiting, CSRF middleware, authentication middleware.
- Web auth (`server/routes/auth.js`): register, login, logout with bcrypt and session regeneration.

- Public pages (`server/routes/pages.js`): landing, pricing, docs, terms.
- EJS layout and views (`server/views/**`).
- Base schema `server/migrations/001_init.sql`: `users`, `api_keys`, `wa_sessions`, `messages`, `credit_ledger`, `transactions`, `integration_orders`, `settings`.
- `server/create-admin.js` CLI to create/reset an admin and print a generated password.
- systemd unit, NGINX vhost and TLS provisioning for `znotice.consultbofa.com`.