QODY

QODY — AI-native QR restaurant ordering platform (SnowIT/ALAI)

- QODY Architecture & Operations
- Payments — Online Payment Toggle (online_payment_enabled)
- Super-Admin Panel
- Staff Password Reset
- Legal & Compliance Pack — QODY (NACRT, BiH+GDPR)
- Authentication Architecture
- CI/CD Pipeline — Deploy Path Validated (2026-07-07)
- QODY — Model fiskalizacije: kartice (Monri) i gotovina (POS kasa lokala) — 2026-08-03
- Privacy Policy v2.0 implementacija (pravnik Ilhan Erma) — MC #106821, 2026-08-04
- VAT-inclusive pricing fix (PDV uračunat u cijene) — MC #106829, 2026-08-04
- QODY CI gate — build-validation policy blocking (MC #106804, 2026-08-05)
- QODY API revision retention and DB connection budget — MC #107292
- Security Sweep — qody.ba (MC #107297)
- QODY Kitchen KDS live update — MC #107293 / #106857 — 2026-08-18

QODY Architecture & Operations

QODY Architecture & Operations

Product: QODY — AI-native QR-based restaurant ordering and payment platform
Operator: SnowIT d.o.o. Sarajevo (powered by ALAI Holding AS)
Market: Bosnia and Herzegovina (BiH) primary, Norway secondary
Status: LIVE — Demo environment (rg-qody-demo), Clean production (rg-qody-prod)
Updated: 2026-06-26

What is QODY?

QODY is a **sit / order & pay** web application for hospitality venues (restaurants, cafes, bars). Guests scan a QR code at their table, browse a digital menu, place an order, and pay — all without installing an app or creating an account. Orders appear in real-time on the kitchen display system (KDS), and staff manage the order lifecycle through to delivery.

Core Flow

1. Guest scans QR code on table → loads venue menu (anonymous, no login)
2. Guest builds cart → submits order → pays via Stripe (test) or Monri (BiH payment gateway, when live)
3. Order instantly appears on kitchen display (WebSocket + SSE realtime)
4. Staff accepts → preps → marks ready → delivers
5. Guest receives receipt, can track status live

Architecture Overview

Tech Stack

- **Backend:** Kotlin 2.1.0 + Ktor 3.0.3 + PostgreSQL 16 + Flyway + Exposed ORM + JWT
- **Frontend:** 3 Vite 6 + React 19 + TypeScript micro-frontends (MFEs) + Astro 5 landing
- **Database:** PostgreSQL 16 with Row-Level Security (RLS) enforced, qody_app role NOBYPASSRLS
- **Realtime:** WebSocket + SSE (Server-Sent Events), transactional outbox pattern
- **Payments:** Stripe Connect (live test mode), Monri architecture ready (Model B)
- **Infrastructure:** Azure Container Apps (ACA), swedencentral region, managed Postgres Flexible Server
- **CI/CD:** Azure Pipelines (local Docker fallback when ACR build stalls)

The 4 Apps + Landing

App	Purpose	DNS (demo)	Port (local)
qody-api	Backend: orders, payments, auth, realtime hub	api.qody.alai.no	8080
qody-guest	Guest ordering (anonymous)	qody.alai.no	5173
qody-admin	Venue manager dashboard (menu CRUD, QR gen, reports, settings)	admin.qody.alai.no	5175
qody-staff-kitchen	Kitchen display system (KDS) — live order board	kuhinja.qody.alai.no	5174
landing	Marketing site (premium "Concept A" design)	(future qody.ba)	N/A

Deployment Environments

Demo/Stage (LIVE)

- **Resource Group:** rg-qody-demo
- **Region:** swedencentral
- **Subscription:** 5b0b4d9b-e677-464e-abf0-5170cbce3b8e (Azure subscription 1)
- **Container Registry:** qodydemoacr.azurecr.io
- **Database:** qody-demo-db.postgres.database.azure.com (Postgres 16, Standard_B1ms)
- **URLs:** qody.alai.no (guest), api.qody.alai.no (API), admin.qody.alai.no (admin), kuhinja.qody.alai.no (kitchen)

Production/Clean (DEPLOYED, no custom DNS yet)

- **Resource Group:** rg-qody-prod
- **Container Registry:** qodyprodacr.azurecr.io
- **Database:** qody-prod-db.postgres.database.azure.com
- **No demo seed** — clean production environment

- **Custom domain pending:** Waiting on qody.ba domain registration (Asmir/SnowIT)

Note: rg-qody-demo is the live demo CEO/Asmir use. Always deploy there for active testing. rg-qody-prod exists but nothing points to it yet.

Feature Catalog (Talas 0?6)

Built over 6 waves (2026-06-22 to 2026-06-24):

Phase 0 (Foundation)

- Gradle Kotlin/Ktor scaffold
- Postgres RLS enforced (qody_app NOBYPASSRLS)
- /health endpoint with RLS self-check (fail-closed)
- 3 MFE shells (Vite) deployable independently
- CI: lint + compileKotlin + test

Phase 1 (Vertical Slice)

- Guest order flow (QR resolve → menu → cart → submit → pay)
- Stripe test payments
- KDS realtime (WebSocket + SSE)
- Admin menu CRUD

Phase 2 (AI + Payments)

- Groq AI chat integration
- Stripe Connect per-venue (destination charge + 0.5% application fee)
- Subscription billing (39 KM Pro / 49 KM Enterprise per month)

Talas 1 (Foundation++)

- Audit log (V11: audit_log table with RLS)
- Address field on venue (JSONB)
- Area/zone hierarchy (venue → area → table)
- 10 payment statuses (PRD-compliant: pending_payment, authorized, paid, failed, cancelled, expired, refund_requested, partially_refunded, refunded, voided)
- Order numbering (order_number TEXT with trigger, e.g., Q-QODYDE-000001)

Talas 2A (Super-Admin Backend)

- Super-admin backend services (SuperAdminService, RefundService, EnhancedSalesReportService)
- Refund request workflow (staff → admin approve/reject → gateway call)
- CSV export with all PRD §12.2 fields (order_number, merchant, location, table, datetime, subtotal, tip, total, payment_status, gateway_reference, order_status, refund_amount)
- Enhanced receipt generation

Talas 2B (Super-Admin UI)

- Super-admin panel (5 tabs: merchants, transactions, billing, refunds, audit)
- Sold-out toggle endpoint (PATCH /admin/menus/items/{id}/availability)
- Staff audio alerts (Web Audio API synthetic beep on new orders)
- Multilang BS-default (Bosnian default for admin + kitchen apps)

Talas 3 (qLub Parity)

- Google Review CTA (post-order, configurable URL per venue)
- Pay-at-table option (pending_cash payment status, order sent to kitchen before payment)
- Configurable tip presets (venue.tip_presets JSONB, default [0,5,10,15])
- Waiter-call button (guest → staff WS push notification)

Talas 4 (Handover + Geofence)

- Per-order handover QR (TOTP-like delivery confirmation, HandoverService, staff scans guest QR to mark delivered)
- Geofence soft opt-in (default OFF, graceful fallback, never hard-block — CEO directive)
- i18n gap-fill (19 missing T3 keys + admin.settings.* keys)

Talas 5 (Gap-Fill)

- CSV table column fix (table_session → restaurant_table join, populates table label in export)
- Email receipt graceful stub (SMTP not configured → logs request, returns 200 with note)
- QR token regeneration endpoint (PATCH /admin/tables/{id}/regenerate-qr, deactivates old token)
- Staff print button (window.print popup with monospace order ticket)
- KDS payment gate (no pending_payment orders visible to staff)

Talas 6 (M-07 Operating Hours)

- Operating hours / ordering schedule
- venue.ordering_enabled (master toggle)

- venue_operating_hours table (day_of_week, open_time, close_time, is_closed, RLS enforced)
- V15 migration
- Demo venue seeded 24/7 (never blocked during demos)

Additional Features (Security, Payments, I18n)

- **Password hashing:** argon2id (replaced SHA-256 in commit 40071db)
- **Step-up 2FA:** TOTP (RFC 6238) for sensitive mutations (fee changes, refunds, staff delete)
- **RLS isolation:** Postgres RLS policies enforce tenant_id boundary, fail-closed
- **JWT auth:** HS256 signing, distinct secrets for JWT vs QR tokens
- **Payments:** Stripe Connect (live test mode), Monri architecture ready (Model B: per-venue merchant accounts)
- **Internationalization:** BS (Bosnian), HR (Croatian), SR (Serbian), EN (English) — full coverage
- **Easy Mode:** Mobile-optimized admin dashboard (touch-friendly for phone/tablet)
- **Market-aware payments:** BiH-simple flow, EU-advanced flow (configurable per venue)

Database Schema

Migrations

V1–V19 in apps/api/src/main/resources/db/migration/

- **V1:** Baseline (organization, venue, table, staff, role, RLS policies)
- **V2:** Domain (menu, order, payment, modifier, translation)
- **V10:** PRD fee model (platform_fee_pct, monthly_fee on venue)
- **V11:** Talas 1 (audit log, address, area/zone, payment statuses, order numbering)
- **V12:** Talas 2A (super-admin backend, refunds, enhanced sales)
- **V13:** Talas 3 (waiter-call, tip presets, pay-at-table)
- **V14:** Talas 4 (handover QR, geofence)
- **V15:** Talas 6 (operating hours, venue.ordering_enabled, venue_operating_hours table)
- **V16:** Fix wave A (audit log field mismatch, websocket reconnect)
- **V17:** Market field (BiH-simple vs EU-advanced payment flows)
- **V18:** Step-up 2FA (TOTP, totp_secret on staff table)
- **V19:** Subscription billing (Stripe subscription tracking)

Key Tables

- **organization, venue** (with platform_fee_pct, monthly_fee, ordering_enabled, address JSONB, branding JSONB)

- **area** (zones within venue, V11)
- **restaurant_table** (with area_id nullable, qr_token_id FK)
- **qr_token** (signed HMAC tokens, nonce for uniqueness)
- **menu, category, menu_item, menu_item_translation** (multilang BS/HR/SR/EN)
- **modifier_group, modifier**
- **order** (with order_number TEXT, payment_status, handover_code, delivered_at)
- **order_line, order_line_modifier** (price snapshots)
- **payment** (unique index on provider + provider_payment_id)
- **table_session** (guest → table → order linkage)
- **staff** (argon2id password_hash, totp_secret for 2FA)
- **role** (OWNER, MANAGER, WAITER, KITCHEN, SUPERADMIN)
- **audit_log** (actor_user_id, action, entity_type, entity_id, metadata JSONB, RLS enforced)
- **waiter_call** (V13: guest call-waiter feature)
- **refund_request** (V12: staff → admin approval workflow)
- **venue_operating_hours** (V15: day_of_week, open_time, close_time, is_closed, RLS)
- **subscription** (V19: Stripe subscription tracking for billing)

Demo Seed

V7__demo_seed_idempotent.sql creates "QODY Demo Bistro" with menu, 2 tables, QR tokens, staff accounts. Idempotent (ON CONFLICT UPSERT). Loads only when ENV=demo.

Payment Architecture

Stripe Connect (Live Test Mode)

- **Model:** Per-venue connected accounts (destination charge + 0.5% application fee)
- **Fee:** 0.5% platform fee (configurable per venue, max 0.7%), guest pays menu total, restaurant nets total−fee, QODY keeps fee
- **Subscription:** 39 KM Pro / 49 KM Enterprise per month (Stripe BAM billing)
- **Onboarding:** Express hosted onboarding (Custom account for demo)
- **Prod hardening pending:** Use Express account type, production keys

Monri (Architecture Ready, Model B)

- **Model B:** Per-venue Monri merchant accounts (bank-agnostic)
- **Banks:** Raiffeisen, Atos, UniCredit, Intesa Sanpaolo (restaurant chooses their own bank)
- **Settlement:** Money flows restaurant-bank → restaurant directly (QODY does NOT hold funds)
- **QODY fee:** Invoiced separately (monthly: 29-49 KM + 0.5% of venue's Qody-channel revenue)

- **Credentials:** Per-venue Monri merchant_id + auth_token encrypted in DB (Azure Key Vault future)
- **Frontend:** Monri.js integration pending (currently Stripe Elements)
- **Test credentials:** Form sent to Monri AM Mahir Mulaomerović (+387 66 284 773)
- **Production:** Each restaurant signs own Monri + bank merchant agreement (7-10 days onboarding)

Recommendation: Start with Model B manual invoicing (certain, no dependency). Explore Monri split-payment API if they confirm support.

Regulatory Compliance (BiH): QODY is a software service provider, NOT a payment intermediary. No e-money/PI license needed under Model B. PCI-DSS scope = SAQ-A (Monri.js hosted checkout). BiH payment lawyer consult recommended (~500 EUR, confirm no PI license needed).

Security

- **Password hashing:** argon2id (Argon2id variant, replaced SHA-256)
- **Step-up 2FA:** TOTP (RFC 6238) for sensitive mutations (fee changes, refunds, staff delete, branding edits)
- **RLS isolation:** Postgres RLS policies enforce venue_id boundary, qody_app role NOBYPASSRLS, /health verifies bypassRls=false on startup (fail-closed)
- **JWT auth:** HS256 signing, 64 hex secrets (JWT_SECRET, QR_TOKEN_SECRET distinct)
- **CORS:** Configured per environment (demo origins vs prod origins)
- **Webhook verification:** Signed webhooks (HMAC), idempotency via unique index on (provider, provider_payment_id)

Internationalization

- **Languages:** BS (Bosnian), HR (Croatian), SR (Serbian), EN (English)
- **Scope:** All guest UI, all staff/kitchen UI, all admin UI (full coverage post-Talas 2B/3/4)
- **Default:** BS for BiH market, EN for Norway/international
- **Implementation:** i18n/bs.json, hr.json, sr.json, en.json; menu_item_translation table (V5); LangSwitcher component

Deployment Procedure

Build Images (Local Docker, amd64 Required for ACA)

```
cd apps/api
docker buildx build --platform linux/amd64 -t qodydemoacr.azurecr.io/qody-api:$(git rev-parse
--short HEAD) .

cd ../guest
docker buildx build --platform linux/amd64 \
  --build-arg VITE_API_BASE_URL=https://api.qody.alai.no \
  --build-arg VITE_STRIPE_PUBLISHABLE_KEY=pk_test_... \
  -t qodydemoacr.azurecr.io/qody-guest:$(git rev-parse --short HEAD) .

# Repeat for admin and staff-kitchen with VITE_API_BASE_URL
```

Push to ACR

```
az acr login --name qodydemoacr
docker push qodydemoacr.azurecr.io/qody-api:$(git rev-parse --short HEAD)
docker push qodydemoacr.azurecr.io/qody-guest:$(git rev-parse --short HEAD)
# ... push admin and staff-kitchen
```

Update Container Apps

```
TAG=$(git rev-parse --short HEAD)

# API (creates new revision, verify before promoting)
az containerapp update \
  --name qody-api \
  --resource-group rg-qody-demo \
  --image qodydemoacr.azurecr.io/qody-api:$TAG

# Verify /health returns 200 + bypassRls=false
curl https://api.qody.alai.no/health

# Promote to 100% traffic (or use traffic split for canary)
az containerapp ingress traffic set \
  --name qody-api \
  --resource-group rg-qody-demo \
  --revision-weight qody-api--0000XXX=100
```

```
# Guest/Admin/Kitchen (single-revision mode, auto-100%)
az containerapp update --name qody-guest --resource-group rg-qody-demo \
  --image qodydemoacr.azurecr.io/qody-guest:$TAG
az containerapp update --name qody-admin --resource-group rg-qody-demo \
  --image qodydemoacr.azurecr.io/qody-admin:$TAG
az containerapp update --name qody-staff-kitchen --resource-group rg-qody-demo \
  --image qodydemoacr.azurecr.io/qody-staff-kitchen:$TAG
```

Post-Deploy Verification (ZAKON PI2)

```
curl https://api.qody.alai.no/health  # Must return
{"status":"ok","db":{"rlsRoleCheck":{"bypassRls":false,"status":"PASS"}}}
curl -I https://qody.alai.no          # 200 + HTML
curl -I https://admin.qody.alai.no    # 200 + HTML
curl -I https://kuhinja.qody.alai.no  # 200 + HTML

# Real browser UAT: load guest menu, add item to cart, verify API calls work (not localhost)
```

?? CRITICAL: MFE Build Args

All three MFEs (guest, admin, staff-kitchen) **MUST** receive `VITE_API_BASE_URL` at build time. Vite bakes env vars into the bundle. If missing, the MFE defaults to `localhost:8080` → deploy succeeds, page loads, but API calls fail. **Always verify** build logs show:

```
===== QODY <MFE> BUILD: API base resolves to https://api.qody.alai.no =====
```

Recurring Lessons (Do NOT Re-Break)

1. **Build-arg or localhost bakes in:** MFE Dockerfiles default VITE_API_BASE_URL=http://localhost:8080 if not passed. Deploy looks fine (200), but guest can't reach API. Always pass --build-arg VITE_API_BASE_URL=https://api.qody.alai.no.
2. **Webhook / system DB calls MUST bypass RLS:** Any background job or webhook handler that queries across venues (e.g., SuperAdminService, billing export, webhook sync) must use the admin datasource (BYPASSRLS connection), NOT the per-request tenant-scoped datasource. Symptom: webhook succeeds but order not updated (RLS blocked the write).
3. **i18n single top-level key:** All i18n keys must be unique at root level (e.g., guest.menu.addToCart, not nested menu: { addToCart: ... }). The i18n library flattens keys. Duplicate keys across modules = one overwrites the other.

4. **Idempotent seed migrations:** Seed migrations (demo venue, tables, menu) must use ON CONFLICT ... DO UPDATE or existence guards that run AFTER dependent rows exist. Existence guards that check BEFORE the row is created silently skip INSERTs → "deploy OK" but data missing.
5. **Landing scroll-reveal fail-safe:** Intersection Observer animations must have a fallback timeout (fade-in after 3s) or IntersectionObserver polyfill. Safari sometimes doesn't fire isIntersecting on initial load.
6. **Verify by live outcome, not green build:** curl 200 + green CI ≠ "works". Always run real-browser UAT (Playwright or manual) that exercises the feature end-to-end (e.g., guest menu → add item → checkout → order reaches KDS).

Secrets & Credentials

Storage: Bitwarden items (alem@alai.no vault)

- "QODY Demo Secrets" (demo environment)
- "QODY Prod Secrets" (production environment)

Azure Container Apps secrets (encrypted at rest, per-app): db-password, jwt-secret, qr-token-secret, stripe-secret-key, stripe-webhook-secret, monri-api-key (when live)

NEVER commit secrets. Always reference Bitwarden item names in documentation, not actual keys.

External Dependencies & Blocked Items

Waiting on Asmir/SnowIT

- **qody.ba domain** registration and DNS delegation (emailed asmirmc@gmail.com)
- **Monri test credentials** (form sent to Monri AM Mahir Mulaomerović)
- **Colors/branding inputs** for final qody.ba site polish
- **Actual restaurant photos** for Specijal (separate SnowIT client project)

Pending BiH Compliance (Pre-Commercial Launch)

- Monri production merchant agreements (per-venue, Model B architecture)

- BiH payment lawyer consult (~500 EUR, confirm no PI license needed)
- Fiscalization integration (eFiskalizacija.ba or fiscal printer)

Operational Continuity Decisions (OCD Register)

ID	Date	Owner	Decision	Rationale
OCD-QODY-001	2026-06-22	FlowForge	Azure ACA (not GCP Cloud Run)	Tenant isolation: QODY needs own RG, not Bilko's GCP project
OCD-QODY-002	2026-06-22	Alem	DNS: qody.alai.no (not qody-demo.alai.no)	Short, brandable URL for demo/pilot
OCD-QODY-003	2026-06-22	FlowForge	Unleash deferred (not deployed)	Feature flags optional for demo; reduces cost/complexity
OCD-QODY-004	2026-06-22	FlowForge	Stripe TEST keys (no live payments)	Demo only; payment flow tested with test cards
OCD-QODY-005	2026-06-22	FlowForge	PostgreSQL RLS enforced (NOBYPASSRLS)	Fail-closed security; multi-tenant isolation at DB layer
OCD-QODY-006	2026-06-24	CEO	Pricing: 0.5% platform fee (max 0.7%), 39-49 KM/month	PRD §2.3 model (NOT 5% — corrected from earlier prototype)
OCD-QODY-007	2026-06-24	Finverge	Monri Model B (per-venue merchant accounts)	Bank-agnostic, no PI license needed, QODY bills separately
OCD-QODY-008	2026-06-24	CodeCraft	MFE build-args NON-OPTIONAL	VITE_API_BASE_URL must be passed at docker build; missing = localhost bake-in
OCD-QODY-009	2026-06-24	CodeCraft	Subscription billing: Stripe BAM (not EUR)	39 KM Pro / 49 KM Enterprise per month (BiH market)
OCD-QODY-010	2026-06-26	Skillforge	BookStack canonical docs	https://docs.alai.no/books/qody — architecture/decisions/features source of truth

Support & Escalation

Owner: FlowForge (DevOps company, ALAI Holding AS)
Escalation: John (AI Director) → Alem (CEO, alem@alai.no, +47 404 74 251)

Runbooks: See repo RUNBOOK.md for deploy, rollback, health checks, secrets rotation, DB ops.

Incidents: Follow ~/system/runbooks/azure-aca-incident.md

Monitoring: Azure Monitor (6 alerts → alem@alai.no). Future: Sentinel ops-watchdog integration.

Support Model: See /tmp/qody-prd/support-model.md for full issue catalog, SLA tiers, resolution playbooks, and proactive monitoring design.

Account Aliases (for MC / Task Management)

- **qody** (product)
- **snowit** (operator)
- **asmir** (partner, tier-1 referral)

Related Documentation

- **Repo:** /Users/makinja/business/ALAI-Holding-AS/products/qody
- **CLAUDE.md:** Project working context
- **BUILD-BLUEPRINT.md:** Build plan (Phase 0→1→2, Talas 0→6)
- **DEPLOY-MAP.md:** Azure ACA deployment authority
- **RUNBOOK.md:** Operational procedures
- **PRD:** /tmp/qody-prd/PRD.txt (Asmir's technical spec v0.1)
- **Gap Analysis:** /tmp/qody-prd/prd-acceptance-audit.md (Talas 0-6 vs PRD audit)
- **Payment Architecture Decision:** /tmp/qody-prd/monri-architecture-decision.md (Finverge, Model B recommendation)
- **Support Model:** /tmp/qody-prd/support-model.md (45 realistic tickets catalogued, SLA, playbooks)

Last Updated: 2026-06-26
Status: LIVE demo (rg-qody-demo), prod clean (rg-qody-prod), Talas 0–6 shipped, V19 migrations applied
Co-Authored-By: Claude Opus 4.8 (1M context) <noreply@anthropic.com>

Payments — Online Payment Toggle (online_payment_enabled)

Online Payment Toggle (online_payment_enabled)

Overview

QODY supports per-venue online payment gating via the `venue.online_payment_enabled` flag. This allows operational control over whether guests can pay online (via card/Stripe) or must use pay-at-table, without requiring code changes or redeployment.

Database Schema

Column: `venue.online_payment_enabled`
Type: `BOOLEAN NOT NULL DEFAULT true`
Migration: `V20__online_payment_toggle.sql` (Flyway)
Location: `apps/api/src/main/resources/db/migration/V20__online_payment_toggle.sql`

Purpose & Context

This flag serves as a **Monri-readiness gate**. While QODY waits for Monri (the BiH payment gateway for production) and online card payments currently run on **Stripe TEST mode**, this toggle allows hiding online payment per-venue.

Intended production payment provider: Monri (Model B — per-venue merchant accounts). See payment architecture decision memo for details on the per-venue Monri integration model.

Behavior

When `online_payment_enabled = true` (default)

- Guest checkout displays online payment UI (Stripe card element via Stripe.js)
- Guest can pay online immediately via `pay_now` flow
- Pay-at-table option also available if `pay_at_table_enabled = true`

When `online_payment_enabled = false`

- Guest checkout **hides** the online payment section (Stripe PaySection component)
- Guest must pay via "Plati kod konobara" (pay-at-table) — this requires `venue.pay_at_table_enabled = true`
- Server-side guard: `POST /guest/order` returns **503 Service Unavailable** with error code `no_payment_path` if both `online_payment_enabled = false` AND `pay_at_table_enabled = false`

Guest API

The flag is surfaced to the guest frontend via:

`GET /guest/venue-settings`

Response field: `onlinePaymentEnabled: boolean`

Frontend Gate

File: `apps/guest/src/pages/CheckoutPage.tsx`
Logic: `venueSettings?.onlinePaymentEnabled !== false`

If the flag is false or undefined-as-false, the Stripe PaySection is not rendered.

Operational Procedure — Re-enabling Online Payment

Scenario

When Monri integration is complete and a venue is ready to accept online payments in production.

Steps

1. **No redeploy required** — this is a pure database change
2. Connect to the QODY database (Azure Postgres)
3. Run the following SQL:

```
UPDATE venue
SET online_payment_enabled = true
WHERE id = '<venue-id>';
```

4. Verify: refresh the guest checkout page — online payment UI should appear

Demo Venue Details

Venue ID: `00000000-0000-0000-0000-000000000002`
Name: QODY Demo Bistro
Current state (as of 2026-06-26):

- `online_payment_enabled = false`
- `pay_at_table_enabled = true`

Current Deployment State

Environment: Azure `rg-qody-demo` (Sweden Central)
API revision: qody-api--0000044
Guest revision: qody-guest--0000033
Commit: `a9668a5`
Verification: Proveo PASS 3/3 (live Playwright UAT, 2026-06-26)

Payment Architecture Notes

Current (temporary): Stripe Connect (TEST mode) — online payment disabled on demo venue pending Monri.
Production target: Monri (Model B — per-venue merchant accounts). Each restaurant will have its own Monri + bank account; QODY stores encrypted per-venue Monri credentials in Azure Key Vault and routes payments through the venue's own merchant account. QODY invoices platform fees separately (0.5% of subtotal + 29-49 KM/month per PRD §2.3). This model avoids QODY holding customer funds and requiring a BiH payment institution license.

Related Documentation

- QODY PRD: `/tmp/qody-prd/PRD.txt` (§2.3 commercial model, §4 payment requirements)
- Monri architecture decision: `/tmp/qody-prd/monri-architecture-decision.md`
- Gap analysis: `/tmp/qody-prd/GAP-ANALYSIS.md`
- Payment flow UAT evidence: `/tmp/qody-prd/uat-guest.md`

Safety Constraints

At least one payment method must be enabled:

- If `online_payment_enabled = false`, then `pay_at_table_enabled` MUST be `true`
- Server-side enforcement: `GuestRoutes.kt` in `POST /guest/order` returns 503 if both are disabled
- Error code: `no_payment_path`

Change Log

Date	Change	Commit
2026-06-26	Feature implemented and deployed (V20 migration, CheckoutPage gate, venue-settings API)	a9668a5

Document owner: John (ALAI Ops)
Last updated: 2026-06-26
Related MC tasks: #104391, #104389

Super-Admin Panel

Super-Admin Panel — MC #104287

Shipped: 2026-06-27
Environment: rg-qody-demo (Azure Container Apps)
Revisions: qody-api--0000046 · qody-admin--0000026
Commits: 749108b (API) · 594d9c9 (Admin frontend)
Validation: Independent code verifier · Securion security peer-review · Proveo live E2E · John live curl

---

Scope

Five capabilities were shipped and verified live. One capability (A-03, per-merchant Monri credential vault) is explicitly deferred and not yet built.

A-01 — Merchant Onboarding UI

Provides SUPERADMIN users with the ability to create new merchants and control their lifecycle from the admin panel.

- **Backend:** `POST /superadmin/merchants` — creates a new merchant/org record.
- **Backend:** `POST /superadmin/merchants/{id}/suspend` — suspends a merchant; guest ordering blocked at venue level.
- **Backend:** `POST /superadmin/merchants/{id}/activate` — re-activates a suspended merchant.
- **Frontend:** `MerchantsTab` — create modal + per-row suspend/activate buttons in the super-admin panel.
- A suspended venue returns 404 on the `/superadmin/impersonate/{venueId}` endpoint (see A-06).

A-02 — Per-Merchant Fee Management

Allows SUPERADMIN to configure per-venue commercial terms with enforced range guards at multiple layers.

- **Endpoint:** `PUT /superadmin/merchants/{id}/fees`
- **Bug fixed:** Frontend previously sent `PATCH` (405 Not Allowed); corrected to `PUT`.
- **Range guards (enforced at Kotlin level + DB CHECK constraint, migration V10):**
  - Platform fee: 0.3% – 0.7%
  - Monthly fee: 29 – 49 KM
- Out-of-range input: HTTP 400 with descriptive message.
- Valid input: HTTP 200.
- Proveo re-verified the 500→400 fix after the range-guard correction.

A-03 — Per-Merchant Monri Gateway Credential Vault (DEFERRED)

Not yet built. Deferred, externally blocked on:

- Monri test credentials pending (MC #104270).
- Azure Key Vault architecture decision pending.

Design intent: per-venue Monri credentials stored encrypted in Azure Key Vault; routing logic in payment service retrieves at request time. No implementation exists in the current revision.

A-04 — Cross-Merchant Transaction Search

Enables SUPERADMIN to search across all venues/merchants for specific transactions.

- **Endpoint:** `GET /superadmin/transactions?search=<query>`
- Server-side `ILIKE` match on `providerPaymentId` and `orderNumber`.
- Scope: all orgs/venues (superadmin only; RLS bypassed for this role).

A-06 — Controlled Support Impersonation

Allows SUPERADMIN to obtain a short-lived, venue-scoped session token to diagnose issues within a specific merchant's context.

- **Endpoint:** `POST /superadmin/impersonate/{venueId}`
- Returns a 30-minute MANAGER-scoped JWT (`type=impersonate`).
- Venue-scoping enforced at Postgres RLS level (not cosmetic application-layer filtering).
- Every call writes a mandatory `audit_log` row (`action=superadmin_impersonate`).
- Token issuance is **atomic** with the audit write: the transaction aborts and returns an error if the audit insert fails — no token without an audit trail.
- Returns 403 for non-SUPERADMIN callers; 404 for suspended venues.

Security Design — Impersonation

Control	Implementation	Rationale
Least privilege role	MANAGER-scoped JWT (not OWNER, not SUPERADMIN)	Limits blast radius; cannot modify org settings or fees
Short TTL	30-minute token expiry	Reduces window for token misuse or exfiltration
Mandatory audit	Atomic audit_log write; aborts on failure	No impersonation can occur without a traceable record
RLS venue isolation	Postgres Row-Level Security enforces venue boundary	Confirmed real by Securion peer-review (HIGH finding: audit-atomicity — fixed before ship)
SUPERADMIN gate	403 for any other role	Endpoint unreachable to merchant users or kitchen/waiter roles

Securion finding (HIGH — fixed): Initial implementation issued the token before writing the audit row. Fixed to wrap both operations in a single DB transaction; token is returned only after the commit succeeds.

A-08 — Monthly Billing Export

Provides SUPERADMIN with a per-venue revenue and fee CSV for a given calendar month.

- **Endpoint:** `GET /superadmin/billing-export?month=YYYY-MM`
- **CSV columns:** `venue_id`, `venue_name`, `org_id`, `org_name`, `currency`, `period_start`, `period_end`, `total_turnover`, `platform_fee_pct`, `fee_owed`, `monthly_fee`, `total_owed`

- **Frontend:** `BillingTab` with a month picker; exports on demand.

Validation Summary

Verifier	Method	Result
Independent code verifier	Static code review	PASS
Securion (Parisa Tabriz)	Security peer-review of A-06 impersonation	HIGH finding (audit atomicity) — fixed; venue-scoping confirmed RLS-real
Proveo (Angie Jones)	Live E2E: 4 PASS + A-02 range-guard 500→400 re-verify	PASS
John	Live curl against rg-qody-demo endpoints	PASS

What Is NOT in This Release

- **A-03 (Monri credential vault):** Deferred — blocked on Monri test creds (MC #104270) and Key Vault architecture.
- Support password-reset UI, merchant onboarding billing, suspend/activate per-merchant billing — tracked separately in MC #104303.

Staff Password Reset

Overview

QODY supports two complementary password-reset mechanisms for venue staff. **Part A** is manager/owner-initiated (admin UI, no email required). **Part B** is self-service via email token. Both paths were verified live on `rg-qody-demo` on 2026-06-28 (qody-api--0000047, qody-admin--0000027, qody-staff-kitchen--0000012; commit `e4c035e`).

Part A — Manager/Owner-Initiated Reset

Endpoint

```
POST /admin/staff/{id}/reset-password
```

Authorization

- Requires `OWNER` or `MANAGER` role.
- **Guard:** a MANAGER cannot reset an OWNER or another MANAGER (returns 403). Only an OWNER can reset any staff member. This prevents privilege escalation and account lockout.

Behaviour

1. Generates a cryptographically random one-time temporary password.
2. Sets the staff account password to this temporary value (argon2id hash stored).
3. Returns the plaintext temporary password **once** in the response body — it is never stored in plaintext and cannot be retrieved again.
4. Displays the temporary password in the StaffView modal in the admin UI.
5. The action is written to the audit log.

Dependencies

No external dependency. Works in demo and production environments regardless of SMTP configuration.

Part B — Self-Service Forgot/Reset Flow

Step 1: Request a reset link

```
POST /auth/forgot-password
Body: { "email": "staff@example.com", "venueSlug": "my-restaurant" }
```

- Unauthenticated endpoint.
- Always returns a generic `200 OK` regardless of whether the email exists (anti-enumeration).
- Rate-limited: 3 requests per 15 minutes per email + IP combination.
- Token generation and email dispatch run async (fire-and-forget) so response timing does not leak account existence.

Step 2: Apply the reset token

```
POST /auth/reset-password
Body: { "token": "<token>", "newPassword": "<password>" }
```

Response codes:

Status	Meaning
`200 ok`	Password updated successfully.
`400 invalid`	Token not found or does not match stored hash.
`400 expired`	Token is older than 1 hour.
`400 used`	Token has already been consumed.
`400 weak`	New password does not meet minimum requirements.

Entry points

- "Zaboravljena lozinka?" link on the admin login screen and the kitchen login screen.
- `/reset-password?token=<token>` page (linked from the email).

Token Security

Property	Value
Entropy	256-bit SecureRandom token
Storage	Only the SHA-256 hash is stored (UNIQUE index) — plaintext never persisted
Single-use	Atomic CAS: `UPDATE ... WHERE used_at IS NULL` prevents TOCTOU double-use
TTL	1 hour
Password hashing	argon2id
Password length	8–1024 characters (upper bound guards against DoS via hashing large inputs)
Scope	Token is bound to a specific `staff_id` — cross-venue reset is not possible
Unauthenticated datasource	Forgot/reset routes use the admin bypass datasource (no session required)

Database migration

V21 adds the `password_reset_tokens` table with `token_hash` (UNIQUE), `staff_id` (FK), `expires_at`, and `used_at` columns.

Email Delivery

Part B uses the existing `EmailService`. In the current demo environment, no SMTP credentials are configured — the service performs a graceful no-op and the reset link is not delivered by email. Part A (manager-initiated) works fully in demo.

> **Production note:** configure an SMTP or Resend secret to enable Part B email delivery. Without it, self-service resets are non-functional for end users.

Validation Chain

1. **verifier(code)** — internal code review
2. **Securion peer-review** — APPROVE-WITH-FIXES; issues found and fixed: timing-oracle, TOCTOU double-use, argon2 DoS guard, UNIQUE index on token hash, MANAGER-to-OWNER reset guard
3. **FlowForge** — deployed to `rg-qody-demo`
4. **Proveo** — browser E2E tests, all flows PASS
5. **John live curl** — all flows verified PASS on 2026-06-28

Related

- MC task: #104424
- Live revisions: qody-api--0000047 | qody-admin--0000027 | qody-staff-kitchen--0000012
- Commit: `e4c035e`
- Production SMTP config: set `SMTP_HOST`, `SMTP_USER`, `SMTP_PASSWORD` (or `RESEND_API_KEY`) environment variables on qody-api container app.

Legal & Compliance Pack — QODY (NACRT, BiH+GDPR)

QODY — Legal & Compliance Documentation

⚠ **STATUS: DRAFT — čeka pravni pregled**

Svi dokumenti u ovom direktoriju su **NACRTI** koji čekaju pregled od strane:

1. BiH pravnika (zakon o zaštiti ličnih podataka, ugovorno pravo)
2. SnowIT d.o.o. Sarajevo (operator / data controller)

NIJEDAN dokument nije pravno obavezujući dok ga SnowIT i pravnik ne odobre i objave.

Dokumenti

Dokument	Status	Jezik	Svrha
politika-privatnosti.md	DRAFT	BS	GDPR/BiH politika privatnosti za goste i merchantse
uvjeti-koristenja.md	DRAFT	BS	Opšti uslovi korištenja platforme
politika-kolacica.md	DRAFT	BS	Cookie policy (trenutno samo essential)
dpa-merchant.md	DRAFT	EN/BS	Data Processing Agreement (merchant ↔ SnowIT)
merchant-agreement.md	DRAFT	BS	Komercijalni ugovor za ugostiteljske objekte

---

Prije Publikacije — Kontrolna Lista

SnowIT mora potvrditi:

☐ **Kontakt email za privatnost**: Predloženo `privacy@qody.ba` ili `info@qody.ba` ili `info@snowit.ba`?

☐ **Retention periods**: Koliko dugo se čuvaju podaci o narudžbama? (predlog: 3 godine za poreske razloge)

☐ **DPO (Data Protection Officer)**: Da li SnowIT ima imenovanog DPO-a? Ako ne, kontakt-osoba?

☐ **Komercijalni uslovi**: Provizija per-merchant je KONFIGURABILAN (0.3–0.7%, default 0.5%) + mjesečna naknada (29–49 KM). Da li merchant-agreement treba fiksnu tabelu ili varijabilnu?

☐ **Bankarski račun / platne instrukcije**: Gdje merchant plaća mjesečnu naknadu?

☐ **Nadležni sud**: Koji sud u BiH je nadležan za sporove? (predlog: Općinski sud u Sarajevu)

☐ **Osiguranje / odgovornost**: Da li SnowIT ima osiguranje odgovornosti? Limit?

BiH pravnik mora pregledati:

☐ **Zakon o zaštiti ličnih podataka BiH**: Da li je opis pravnog osnova tačan? (legitimni interes za fulfillment, ugovor za plaćanje)

☐ **AZLP registracija**: Da li SnowIT mora registrirati processing kod Agencije za zaštitu ličnih podataka?

☐ **Retention periods**: Da li su predloženi periodi u skladu sa BiH poreskim propisima?

☐ **Data subject rights**: Da li je procedura za ostvarivanje prava (pristup, brisanje, prigovor) adekvatna?

☐ **International transfers**: Da li je opis prenosa u EU (Azure swedencentral, Stripe EU) dovoljno jasan za GDPR?

☐ **Uvjeti korištenja**: Da li su klauzule o odgovornosti i ograničenjima valjane po BiH zakonu o zaštiti potrošača?

☐ **Merchant DPA**: Da li je struktura processor–controller odnosa pravilno opisana?

☐ **Fiskalizacija**: Da li merchant-agreement mora sadržavati odredbe o eFiskalizaciji?

☐ **Payments licensing**: Potvrđeno Model B (per-venue Monri accounts, QODY ne drži sredstva) = ne treba PI/e-money licenca?

Tehni?ki PLACEHOLDER-i za popunjavanje:

☐ `[PLACEHOLDER: privacy email]` → stvarna email adresa

☐ `[PLACEHOLDER: retention period]` → tačan broj godina/mjeseci

☐ `[PLACEHOLDER: platform fee %]` → per-merchant vrijednost ili varijabilna tabela

- ☐ `[PLACEHOLDER: mjesečna naknada KM]` → per-merchant vrijednost ili tarifa
- ☐ `[PLACEHOLDER: SnowIT DPO]` → ime i kontakt ili 'nema DPO-a'
- ☐ `[PLACEHOLDER: SnowIT bank account]` → IBAN za plaćanje

Pravna napomena

FlowForge (ALAI Holding AS) je tehnički provajder, NE pravni savetnik.
Ovi dokumenti su generisani na osnovu:

- GDPR tipskih template-a
- Javnih BiH propisa (zakon o zaštiti ličnih podataka)
- Razumijevanja QODY arhitekture (Model B payments, RLS, minimalni cookies)

OBAVEZNO je angažovati licenciranog BiH pravnika prije publikacije bilo kog dokumenta.

Kreirao: FlowForge (via John)
Datum: 2026-06-29
MC Task: #104526
Status: DRAFT — čeka SnowIT + pravnik

Authentication Architecture

QODY Authentication Architecture

Status: LIVE (demo + prod: 2026-06-30, commit e63ab88)
Revisions:
• Demo: qody-api--0000060 / qody-admin--0000033 / qody-staff-kitchen--0000016
• Prod: qody-api-prod--0000009 / qody-admin-prod--0000005 / qody-staff-kitchen-prod--0000004
MC: #104547 (cookie migration), #104553 (prod deploy), #104554 (rate-limiting)
Author: Securion (Parisa Tabriz) | FlowForge (Kelsey Hightower) deploy

---

Overview

QODY uses a **custom JWT authentication system** with httpOnly cookie transport, cryptographic CSRF protection, login rate-limiting, and optional TOTP 2FA step-up. The system deliberately **does NOT use Microsoft Entra** (unlike ALAI's other products like Bilko) — QODY is a self-contained BiH product requiring no Azure AD integration.

Authentication Stack:

- **JWT library:** auth0/java-jwt with HMAC256 signing
- **Password hashing:** argon2id (OWASP-recommended)
- **2FA:** TOTP (Time-based One-Time Password, RFC 6238)
- **Session transport:** httpOnly cookies (SameSite=Lax, Secure)
- **CSRF protection:** Signed double-submit cookie pattern
- **Rate-limiting:** Dual-key sliding-window (per email+IP and per IP)

Why NOT Entra: QODY targets BiH restaurants (Bosnia and Herzegovina market) where Azure/Microsoft SSO adds no value and creates unnecessary external dependencies. Self-contained JWT auth allows offline merchant onboarding, faster login flows, and complete control over session management. The architecture decision is documented in BookStack "QODY Architecture Decisions".

Login Rate-Limiting

QODY implements dual-key sliding-window rate limiting to protect staff and super-admin login endpoints against brute-force attacks and password spraying.

Implementation

Service: `LoginRateLimiter.kt` (in-memory, per-replica)

Thresholds:

- **5 failed attempts per (email + IP) in 15-minute window** — protects individual accounts
  - Resets on successful login
  - Prevents targeted account attack
- **20 failed attempts per IP in 15-minute window** — protects against spray attacks
  - Does NOT reset on success (spray defense needs longer persistence)
  - Prevents attacker trying many accounts from single IP

Response:

- Before threshold: HTTP 401 with vague message (no email-existence leak)
- After threshold: HTTP 429 Too Many Requests
  - Header: `Retry-After: <seconds>`
  - Body: `{"error":"Too many failed attempts","code":"RATE_LIMITED"}`

Endpoints Protected:

- `POST /staff/auth/login`
- `POST /superadmin/auth/login`

Known Limitation:
In-memory storage means limits are per-replica. For multi-replica deployments, the effective limit multiplies (5 attempts × N replicas). For true global limits, migrate to Redis (MC #104549 — same task covers WebSocket ticket store).

Live Verification

Curl test on https://api.qody.ba/staff/auth/login with invalid credentials (2026-06-30):

- Attempts 1-5: HTTP 401
- Attempt 6: HTTP 429 (threshold enforced)
- Subsequent attempts: HTTP 429 with Retry-After header

Evidence: `/Users/makinja/system/evidence/104554/prod-ratelimit-live-verify.txt`

Production Deployment

QODY authentication hardening (httpOnly cookie + CSRF + WebSocket ticket + rate-limiting) is now live on **production** and demo.

Demo Environment

Status: LIVE (2026-06-30, commit e63ab88)
Resource Group: rg-qody-demo (swedencentral)
Domains: demo.app.qody.ba | demo.admin.qody.ba | demo.kuhinja.qody.ba | demo.api.qody.ba
Cookie Domain: `.qody.ba`

Note: Demo topology changed on 2026-06-30 — `qody.alai.no` DECOMMISSIONED. All demo traffic migrated to `demo.*.qody.ba` with `COOKIE_DOMAIN=.qody.ba` (same as prod).

Production Environment

Status: LIVE (2026-06-30, commit e63ab88)
Resource Group: rg-qody-prod (gentlecliff-98883162, swedencentral)
Domains: app.qody.ba | admin.qody.ba | kuhinja.qody.ba | api.qody.ba
Cookie Domain: `.qody.ba`
Image Tag: 20260630-httponly-prod-104553

Revisions:

- qody-api-prod--0000009 (100%)
- qody-admin-prod--0000005 (100%)
- qody-staff-kitchen-prod--0000004 (100%)

Environment Variables:

- `COOKIE_DOMAIN=.qody.ba`
- `DEV_AUTH_MODE` — NOT SET (cookie-only mode)
- `CORS_ORIGINS=https://app.qody.ba,https://admin.qody.ba,https://kuhinja.qody.ba,...`

Production Verification Evidence

Health Check:

```
curl https://api.qody.ba/health
```

 PASS — RLS check: `bypassRls=false`, `status=PASS`

Frontends: All serving HTTP/2 200

- https://app.qody.ba
- https://admin.qody.ba
- https://kuhinja.qody.ba

Login Endpoint: Live and processing requests

- Rate-limiting confirmed (429 after 5 failed attempts)
- Application logs show proper auth flow
- CORS headers present for all origins

Rate-Limit Verification: Direct curl test on live prod:

- 5 sequential failed login attempts → HTTP 401
- 6th attempt → HTTP 429 + Retry-After header
- Evidence: `/Users/makinja/system/evidence/104554/prod-ratelimit-live-verify.txt`

Known Gap: Full Browser UAT on Production

Status: PENDING

The production database is clean (no demo seed). Super-admin account `asmirmc@gmail.com` exists and is active (role SUPERADMIN, not deleted — verified in prod DB 2026-08-02).
⚠ **CORRECTION (2026-08-02):** an earlier revision of this page stated the prod password was `PLACEHOLDER_PW`. That was true when written but was **remediated on 2026-07-06** — the literal placeholder was a live, guessable SUPERADMIN credential and has been rotated (argon2id via rehash-on-login; Bitwarden "QODY Prod Secrets" → `super_admin_password` updated). Do not treat the placeholder string as current.

What's Verified:

- Code deployed (same commit as demo)
- Health checks PASS
- Frontends serving 200
- Login endpoint live
- Rate-limiting active (429 confirmed)
- COOKIE_DOMAIN=.qody.ba set
- CORS includes all prod domains
- Demo browser UAT PASSED (Proveo 10/10 on demo.*.qody.ba with identical code)

What's Pending:

- ☐ Full browser login flow on prod (password rotated 2026-07-06; browser UAT still not run end-to-end)
- ☐ Playwright verification of Set-Cookie headers on prod

Equivalence Confidence:
Demo and prod run **identical code** (same commit e63ab88, same build args, same COOKIE_DOMAIN=.qody.ba). Demo passed full Proveo browser UAT (10/10 cross-role scenarios). Prod differs only in database content (clean vs seeded). Cookie flow proven on demo → high confidence in prod.

Next Step for Full Clearance:
Run Proveo browser UAT on prod → verify Set-Cookie headers contain `Domain=.qody.ba; HttpOnly; Secure; SameSite=Lax`. The password blocker is resolved (rotated 2026-07-06).

Known Gap: no self-service password reset for SUPERADMIN (open)

Status: OPEN — tracked as MC #106666 (H).

Verified live on 2026-08-02:

- `POST https://api.qody.ba/superadmin/auth/forgot-password` → **HTTP 404** (endpoint does not exist).
- `POST https://api.qody.ba/auth/forgot-password` without `venueSlug` → **HTTP 400** `{"error":"email and venueSlug are required"}`.
- The merchant flow resolves staff via `VenueTable.slug` → `StaffTable(email, venueId)`, so a venue-less SUPERADMIN can never be found by it. The admin UI "Zaboravili ste lozinku?" form (Slug lokala + Email) is therefore a dead end for super-admins.

Consequence: every forgotten SUPERADMIN password currently requires a manual `password_reset_tokens` INSERT against the prod database by an operator with `qody_flyway` access. Done once on 2026-08-02 (evidence: `~/system/evidence/106666/A-manual-reset-2026-08-02.md`).

Note for implementers: `PasswordResetService.resetPassword` is already venue-independent (token_hash → staff_id, `purpose='password_reset'`), and `https://admin.qody.ba/reset-password?token=…` renders correctly for a super-admin token. Only the *initiation* half needs building.

Cookie Transport

Cookie Attributes

QODY sets **two cookies** on successful login:

Cookie Name	HttpOnly	Secure	SameSite	Domain	Path	Max-Age
`qody_auth`	✅ Yes	✅ Yes	Lax	`.qody.alai.no` (demo) / `.qody.ba` (prod)	`/`	28800 (8 hours)
`qody_csrf`	❌ No	✅ Yes	Lax	Same as above	`/`	28800 (8 hours)

Environment-driven domain: The `COOKIE_DOMAIN` env var controls the `Domain` attribute:

- Demo: `COOKIE_DOMAIN=.qody.alai.no`
- Prod: `COOKIE_DOMAIN=.qody.ba`

Why SameSite=Lax (not Strict)

QODY's deployment topology:

```
Demo:  admin.qody.alai.no  → fetch →  api.qody.alai.no
       kuhinja.qody.alai.no → fetch →  api.qody.alai.no

Prod:  admin.qody.ba → fetch →  api.qody.ba
       kuhinja.qody.ba → fetch →  api.qody.ba
```

The frontend (admin MFE) and API are **different subdomains** but share the same eTLD+1 (`alai.no` for demo, `qody.ba` for prod). Under browser same-site rules, these are considered **same-site** despite the subdomain difference.

SameSite values comparison:

- `Strict`: Would block cookies on cross-site **top-level navigations** (e.g., clicking a link from email → qody.ba would not send cookie → forced re-login). Too restrictive for normal use.
- `Lax`: Allows cookies on same-site cross-subdomain fetches AND top-level GET navigations. **Blocks** cookies on cross-site POST/PUT/DELETE (where CSRF attacks originate). This is the sweet spot.
- `None`: Would send cookies on **all** cross-site requests, including from attacker origins. Requires additional CSRF layer (which we have, but defense-in-depth says use Lax).

Result: `SameSite=Lax` provides CSRF protection against external attackers while allowing natural user flows.

Cookie-Only Mode vs Dev Mode

Production/demo (cookie-only):

- Login endpoint returns `{"token": "", ...}` (empty string)
- JWT lives **only** in the `qody_auth` httpOnly cookie (JavaScript cannot read it)
- All API calls use `credentials: 'include'` to send cookies automatically

Local dev (bearer token fallback):

- Environment variable `DEV_AUTH_MODE=bearer` enables dual-write mode
- Login endpoint returns **both** the cookie AND `{"token": "eyJ...", ...}` in response body
- Frontend with `VITE_AUTH_MODE=bearer` reads the body token and sends `Authorization: Bearer` header
- Allows developers to run localhost without reverse-proxy/custom-domain setup

This is detected automatically — no developer config needed in production builds.

CSRF Protection

QODY uses the **Signed Double-Submit-Cookie** pattern (OWASP-recommended stateless CSRF defense).

How It Works

On login (server):

1. Issue JWT in `qody_auth` cookie (httpOnly)
2. Compute `csrf_token = HMAC-SHA256(JWT_SECRET, staffId + ":" + exp)` (hex-encoded, 64 chars)
3. Issue second cookie: `qody_csrf=<csrf_token>` (NOT httpOnly, so JavaScript can read it)

On every mutating request (POST/PUT/PATCH/DELETE) — server:

1. Read `qody_auth` cookie → validate JWT as usual
2. Read `X-QODY-CSRF-Token` request header
3. Re-derive `expected = HMAC-SHA256(JWT_SECRET, sub + ":" + exp)` from JWT claims
4. Constant-time compare `X-QODY-CSRF-Token` to `expected`
5. Return `403 CSRF_TOKEN_INVALID` if mismatch or header missing

GET and HEAD requests are exempt (must remain idempotent).

Frontend Requirements

- On login, read the `qody_csrf` cookie value (readable — no httpOnly)

- Store in memory (module-level variable, NOT localStorage)
- Attach `X-QODY-CSRF-Token: <value>` to every non-GET fetch in `apiFetch()` and `saFetch()`
- On logout, clear in-memory value (server clears both cookies via `Max-Age=0`)

CORS header allowlist: `X-QODY-CSRF-Token` is added to `Cors.kt` `allowHeader()` alongside `X-Step-Up-Token`.

Why This Pattern

- **Stateless:** No server-side session store required (QODY API is stateless, JWT-only)
- **HMAC-signed:** The CSRF token is cryptographically bound to the session; an attacker cannot forge it without knowing `JWT_SECRET`
- **Defense-in-depth:** SameSite=Lax blocks most CSRF, but if a browser has SameSite bugs, the CSRF token catches it

---

WebSocket Authentication

The Old Way (Removed)

Before MC #104547: WebSocket connections used `wss://api.qody.alai.no/staff/stream?token=<jwt>`.

Problem: The 8-hour JWT appeared in:

- Browser history (`wss://...?token=eyJ...`)
- Server access logs (query params are logged)
- Any intermediate proxy logs

This is a **credential leak** (the token survives browser restarts via history autocomplete).

The New Way (Ticket-Based)

Current flow (2026-06-30 deploy):

1. **Obtain a WebSocket ticket:**

```
POST /staff/auth/ws-ticket
Cookie: qody_auth=<jwt>
X-QODY-CSRF-Token: <csrf>
```

```
Response: {"ticket": "abc123...", "expiresAt": "..."}
```

2. **Connect WebSocket with the ticket:**

```
wss://api.qody.alai.no/staff/stream?ticket=abc123...
```

Ticket properties:

- **60-second TTL** (not 8 hours)
- **Single-use** (consumed on first connection, invalid after)
- **Stored in-process** (current implementation uses in-memory map — see "Known Follow-Ups" below)

Why this is secure:

- The ticket is short-lived (60s) — even if leaked via browser history, it expires fast
- It's single-use — replaying the URL doesn't work
- The long-lived JWT never appears in the URL

Why not just read the cookie on WebSocket upgrade?
The WebSocket API in browsers cannot send custom headers on the initial upgrade request. Cookies **are** sent, but the `WebSocket` constructor doesn't expose them. The `?ticket=` query param is the standard workaround (used by Socket.IO, etc.) — we just made the ticket ephemeral.

Authentication Endpoints

Staff & Admin Login

Endpoint	Role	Request Body	Response (Cookie-Only)	Response (Dev Mode)
`POST /staff/auth/login`	STAFF / OWNER	`{"venueSlug": "...", "email": "...", "password": "..."}`	`{"token": "", "staffId": "...", "name": "...", "role": "...", ...}` + Set-Cookie headers	Same JSON but `token` field contains JWT
`POST /superadmin/auth/login`	SUPER_ADMIN	`{"email": "...", "password": "..."}`	Same pattern	Same pattern

Cookies set:

- `qody_auth=<jwt>; HttpOnly; Secure; SameSite=Lax; Domain=<COOKIE_DOMAIN>; Path=/; Max-Age=28800`

- `qody_csrf=<hmac>; Secure; SameSite=Lax; Domain=<COOKIE_DOMAIN>; Path=/; Max-Age=28800`

Logout

Endpoint	Effect
`POST /staff/auth/logout`	Clears `qody_auth` and `qody_csrf` cookies (Max-Age=0)
`POST /superadmin/auth/logout`	Same as above

Frontend action on logout:

1. `POST /staff/auth/logout` (with `credentials: 'include'` so cookies are sent)
2. Clear in-memory CSRF token variable
3. Purge legacy localStorage keys (`qody_admin_token`, `qody_staff_token`, etc.) — these may persist from before the migration

WebSocket Ticket

Endpoint	Role	Response
`POST /staff/auth/ws-ticket`	STAFF / OWNER	`{"ticket": "abc123...", "expiresAt": "2026-06-30T22:00:00Z"}`

Then connect: `wss://api.qody.alai.no/staff/stream?ticket=abc123...`

Password Reset (Out of Band)

Endpoints:

- `POST /auth/forgot-password` — sends email with single-use token
- `POST /auth/reset-password` — validates token, sets new password

These are NOT cookie-based — the reset token is a URL parameter emailed to the user. Out of scope for this doc.

TOTP 2FA Step-Up Compatibility

QODY supports TOTP 2FA for high-privilege actions (e.g., refunds, merchant suspend).

The JWT-to-cookie migration does NOT affect TOTP. Verification:

1. The TOTP flow issues a **step-up nonce** (`stepUpToken`), not a new session JWT.

2. This nonce travels in the `X-Step-Up-Token` request header (already in CORS `allowHeader`).
3. The TOTP code itself travels in the POST body of `/admin/mfa/step-up` (`{"code": "NNNNNN"}`).
4. The session JWT (now in `qody_auth` cookie) continues to satisfy the `authenticate("staff-jwt")` guard on all TOTP routes.

No change needed to the TOTP flow logic — only the JWT **extraction point** changed (from `Authorization` header to `qody_auth` cookie).

Recovery codes also travel in POST body. Unchanged.

CORS Configuration

Moving to cookie-based auth requires `credentials: 'include'` on all frontend fetch calls. Browsers enforce:

- `Access-Control-Allow-Credentials: true` (already set in `Cors.kt`)
- `Access-Control-Allow-Origin` **MUST NOT be** `*` (wildcard rejected when credentials are included)

QODY CORS_ORIGINS (explicit allowlist):

Demo environment:

```
https://qody.alai.no,https://admin.qody.alai.no,https://kuhinja.qody.alai.no,https://demo.app.qody.ba,https://demo.admin.qody.ba,https://demo.kuhinja.qody.ba
```

Prod environment:

```
https://app.qody.ba,https://admin.qody.ba,https://kuhinja.qody.ba
```

Local dev (hard-coded in Cors.kt):

```
http://localhost:5173, http://localhost:5174, http://localhost:5175
```

Critical: The `CORS_ORIGINS` env var is populated in `DEPLOY-MAP.md` secrets table and set in Azure Container Apps configuration. The code reads this env var and splits on comma. No wildcard fallback.

Critical Ops Lesson: Edge Cookie Incident

Cross-reference: `technical_bilko_entra_edge_cookie` (BookStack / MEMORY.md)

Bilko failure pattern (2026-06-23): Bilko's edge middleware checked for a `refresh-cookie` **or** an `Authorization` header to detect auth state. When Bilko switched to cookie-only auth, the middleware didn't know about the new cookie name → treated all requests as unauthenticated → redirect loop.

QODY guard requirement:
Any future route guard, middleware, reverse proxy rule, or Ktor plugin that checks "is this request authenticated?" **MUST read the `qody_auth` cookie by name.** It MUST NOT:

- Inspect `Authorization: Bearer` as primary path (frontend no longer sends this in production)
- Inspect `qody_admin_token` from localStorage (server cannot see localStorage)
- Use any other cookie name

Specific guard locations in QODY:

1. `authenticate("staff-jwt")` plugin in `Application.configureSecurity()` — token extraction changed from `Authorization` header to `qody_auth` cookie (with local-dev `Bearer` fallback)
2. `verifyStaffJwt()` in WebSocket handler — now validates the `?ticket=` param (which is itself issued via a cookie-authenticated endpoint)
3. `requireStepUp()` in TOTP routes — reads `X-Step-Up-Token` header (not session cookie; no change needed)

If you add a new middleware/ingress rule: it MUST check `qody_auth` cookie. Otherwise, all users will appear unauthenticated.

Known Follow-Ups

1. WebSocket Ticket Store (MC #104549)

Current state: The WS ticket store is **in-process** (ConcurrentHashMap in `RealtimeHub.kt`).

Problem: If the API scales to multiple replicas (horizontal scaling), a ticket issued by replica A won't be visible to replica B → connection fails.

Solution: Move ticket store to **Redis** (shared state across replicas). TTL handled by Redis `SETEX` (60s auto-expiry). Single-use handled by Redis `GET` + `DEL` atomic operation.

Impact: Non-blocking. Single-replica demo works fine. Defer until multi-replica deployment.

2. CORS Origin List (Dynamic Management)

Current state: CORS origins are hardcoded in `CORS_ORIGINS` env var.

Future improvement: Store allowed origins in the database (per-venue or system-wide config table). This allows adding new custom domains (e.g., `order.my-restaurant.ba`) without redeploying the API.

Impact: Low priority. Current env-var approach works for all known domains.

Recurring Build-Arg Trap (Cross-Reference)

CRITICAL for MFE rebuilds: All three MFEs (guest, admin, staff-kitchen) **MUST** receive `VITE_API_BASE_URL` at Docker build time. Vite bakes env vars into the bundle.

Symptom of missing build-arg:

- Deploy succeeds (200 OK)
- Page loads fine
- API calls fail (browser tries `localhost:8080` or `undefined`)

Evidence of correct build:

```
===== QODY <MFE> BUILD: API base resolves to https://api.qody.alai.no =====
```

How to verify live:
Open browser console → check network tab → API calls should go to `api.qody.alai.no`, NOT `localhost`.

This has broken QODY 3 times (see MEMORY.md "Talas 2B", "Talas 4", "UAT fix-wave A"). The fix is durable in the Dockerfile (build-arg default), but **always verify** the build log when deploying MFEs.

Security Properties After Implementation

Threat	Before (localStorage + Bearer)	After (httpOnly cookie)
XSS exfiltrates session JWT	❌ Trivial — JWT readable via `localStorage.getItem()`	✅ Eliminated — httpOnly cookie not readable by JS
CSRF (state-mutating requests)	✅ Not a concern (Bearer token not auto-sent)	✅ Mitigated by SameSite=Lax + HMAC-signed double-submit cookie
XSS injects CSRF token	N/A	✅ Mitigated — CSRF token is HMAC-signed with server secret; forgery requires `JWT_SECRET`
Cookie stolen via non-HTTPS	✅ N/A (already HTTPS-only)	✅ Mitigated — Secure flag, HSTS deployed on Azure ACA
Cookie sent to wrong subdomain	N/A	✅ Scoped to `Domain=.qody.alai.no` or `.qody.ba` — not sent to unrelated ALAI products
JWT exposed in WS URL	❌ Present (8-hour token in `?token=` query param)	✅ Eliminated — 60s single-use ticket instead
Session fixation	✅ Not applicable (JWT stateless)	✅ Not applicable (JWT stateless)

Net result: The httpOnly cookie migration **eliminates** the XSS credential exfiltration vector (the primary threat) while maintaining CSRF protection via defense-in-depth (SameSite + HMAC).

Non-Goals (Out of Scope)

- **Refresh tokens:** Out of scope. QODY uses 8-hour access tokens (matches a restaurant shift). Refresh token rotation would require server-side token store. Deferred.
- **Guest MFE (`qody-guest`):** Guest flow is anonymous (QR-token scoped, no login). This spec covers only authenticated staff/admin flows.
- **Password reset tokens:** Single-use URL tokens emailed to user. Do not use cookies. Out of scope.
- **OAuth / Social Login:** QODY is BiH-market only, no Google/Facebook login. Out of scope.

Live Verification (2026-06-30 Deploy Evidence)

From: `/Users/makinja/system/evidence/104547/flowforge-deploy-evidence-20260630.txt`

Deployment Metadata

- **Branch:** `feat/qody-httponly-cookie-104547`
- **Commit:** `b15efa2`
- **Environment:** `rg-qody-demo` (swedencentral, Azure)
- **Images:** `qodydemoacr.azurecr.io/qody-{api,admin,staff-kitchen}:20260630-httponly-104547`
- **Revisions:**
  - `qody-api--0000058` (100% traffic)
  - `qody-admin--0000031` (100% traffic)
  - `qody-staff-kitchen--0000014` (100% traffic)
  - `qody-guest--0000032` (unchanged, not rebuilt)

Environment Configuration (qody-api)

```
COOKIE_DOMAIN=.qody.alai.no
CORS_ORIGINS=https://qody.alai.no,https://admin.qody.alai.no,https://kuhinja.qody.alai.no,https://demo.app.qody.ba,https://demo.admin.qody.ba,https://demo.kuhinja.qody.ba
DEV_AUTH_MODE=(not set — cookie-only mode active)
```

Verification Tests

1. Frontend endpoints (200 OK):

```
 https://qody.alai.no → HTTP/2 200
 https://admin.qody.alai.no → HTTP/2 200
 https://kuhinja.qody.alai.no → HTTP/2 200
 https://demo.app.qody.ba → HTTP/2 200
 https://demo.admin.qody.ba → HTTP/2 200
 https://demo.kuhinja.qody.ba → HTTP/2 200
```

2. API health check:

```
 https://api.qody.alai.no/health → HTTP/2 200
{
  "status": "ok",
  "version": "0.1.0",
  "db": {
    "connected": true,
```

```
    "rlsRoleCheck": {
      "role": "qody_app",
      "bypassRls": false,
      "status": "PASS"
    }
  }
}
```

3. HttpOnly cookie auth verification:

```
POST /staff/auth/login
Body: {"venueSlug":"qody-demo","email":"owner@demo.qody","password":"demo1234"}

Response Headers:
set-cookie: qody_auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...; HttpOnly; Secure; SameSite=Lax;
Domain=.qody.alai.no; Path=/; Max-Age=28800
set-cookie: qody_csrf=14b7a79615f0f46a51a5c38753b306be6fb3b0301ef256237d6971c73bbd7c16;
Secure; SameSite=Lax; Domain=.qody.alai.no; Path=/; Max-Age=28800

Response Body:
{"token":"","staffId":"00000000-0000-0000-0000-000000000080","name":"Demo
Owner","role":"OWNER","venueId":"00000000-0000-0000-0000-000000000002","venueName":"QODY Demo
Bistro"}
```

 VERIFIED:

- `Set-Cookie` header contains `qody_auth` with `HttpOnly`, `Secure`, `SameSite=Lax`, `Domain=.qody.alai.no`
- `Set-Cookie` header contains `qody_csrf` with `Secure`, `SameSite=Lax`, `Domain=.qody.alai.no` (no HttpOnly)
- Response JSON `"token"` field is **EMPTY** (`""`) — no bearer token in body

4. CSRF protection verification:

```
PATCH /admin/settings
Cookie: qody_auth=<valid jwt>
(WITHOUT X-QODY-CSRF-Token header)

Response: HTTP/2 403
{"error":"CSRF token required","code":"CSRF_TOKEN_INVALID"}
```

 VERIFIED: Mutation endpoint correctly rejects requests without CSRF token.

Summary

QODY authentication is production-grade httpOnly-cookie-based JWT auth:

- Session credential not readable by JavaScript (XSS-proof)
- CSRF protected by SameSite=Lax + HMAC-signed double-submit cookie
- WebSocket auth uses ephemeral 60s single-use tickets (no JWT in URL)
- TOTP 2FA step-up compatible (unchanged)
- CORS properly configured (explicit origin allowlist, credentials: true)
- Live on demo (2026-06-30, revisions verified)
- Edge-cookie-trap lesson integrated (guards check `qody_auth` cookie by name)

Remaining work:

- MC #104549: Move WS ticket store to Redis (multi-replica support)
- Proveo validation (MC #104547 gates)

Documentation sources:

- Design spec: `/Users/makinja/business/ALAI-Holding-AS/products/qody/docs/auth-cookie-spec.md` (Securion)
- Deploy evidence: `/Users/makinja/system/evidence/104547/flowforge-deploy-evidence-20260630.txt` (FlowForge)
- Live verification: curl + browser UAT on https://api.qody.alai.no

---

Last updated: 2026-06-30
Next review: Before production cutover (qody.ba merchant onboarding)

CI/CD Pipeline — Deploy Path Validated (2026-07-07)

CI/CD Pipeline — Deploy Path Validated (2026-07-07)

MC: #104863
Status: COMPLETE — First fully green end-to-end demo deploy run
Evidence: https://dev.azure.com/alai-holding/QODY/_build/results?buildId=272

Summary

QODY's Azure DevOps CI/CD pipeline is now **fully validated** end-to-end. Build #272 (2026-07-07 00:22 UTC) was the first run to successfully complete all 4 stages:

1. **CI Gates** — Backend tests (Java 21 + Gradle) + frontend builds (npm ci + TypeScript/Vite) + Gitleaks secret scan
2. **Build images → ACR** — 4 Docker images (qody-api, qody-guest, qody-admin, qody-staff-kitchen) built and pushed to qodydemoacr.azurecr.io
3. **Security: Trivy image scan** — All 4 images scanned (HIGH + CRITICAL severity, exit-code 1 on findings) — **CLEAN** (jackson-databind 2.18.8, netty-bom 4.1.135.Final, Alpine patched)
4. **Deploy → QODY demo ACA** — Canary deploy (qody-api: 0%→smoke curl→100%) + single-revision updates (guest/admin/kitchen) + **flagship pay-to-kitchen UAT** (Playwright hard gate) — **PASS**

Live verification (John, 00:24 UTC):

- https://demo.api.qody.ba/health → `200` `{"status":"ok","db":{"rlsRoleCheck":{"bypassRls":false,"status":"PASS"}}}`
- demo.app/admin/kuhinja.qody.ba → All `200` with non-empty HTML

Root Cause (Why Pipeline Was Dead)

The Real Problem (tool-verified):
The `azure-qody` Azure RM service connection (and `qody-demo` ADO environment) had **never been explicitly authorized** for use by pipeline definition #3 ("QODY CI-CD"). Azure DevOps resource protection requires this one-time authorization the first time a pipeline run reaches a job that references the resource.

Every previous run either:

- Had `deployDemo=false` (never reached `Build_Images` stage), OR
- Was a tag run (e.g., `v1.7.0-onboarding-rc1`) that **silently stalled forever** at `Checkpoint.Authorization` (no error surfaced, just `status: inProgress` indefinitely)

Fixed via:

```
PATCH .../pipelines/pipelinePermissions/endpoint/62c293a8-2241-465a-a9ac-93af127472de
PATCH .../pipelines/pipelinePermissions/environment/4
body: {"pipelines":[{"id":3,"authorized":true}]}
```

Both returned `authorized: true` immediately. The stuck `Checkpoint.Authorization` record transitioned `inProgress` → `completed`/`succeeded` within the same second — direct, unambiguous proof.

Initial hypothesis (tag/branch trigger mismatch) was a red herring: The tag run *did* fire (build #236 exists, its `CI_Gates` stage succeeded), it just hung afterward on the authorization gap. The tag/branch filter theory is likely correct as a matter of ADO semantics, but it was NOT the blocker.

Path to Green (5 Runs, 5 Real Bugs Fixed)

Each failed run exposed a genuine latent issue:

Run	Failure	Root Cause	Fix
#253	`Build_Images` failed	Missing `apps/guest/nginx.default.conf` + `apps/staff-kitchen/nginx.default.conf` (referenced by Dockerfile COPY lines since MC #104749, but never committed to git)	Committed the 2 missing `.conf` files (HSTS + security headers, identical pattern to `admin`)

Run	Failure	Root Cause	Fix
#265	`Security_Image_Scan` failed	16 HIGH CVEs in jackson-databind + 2 HIGH in p11-kit	Bumped jackson-databind → 2.18.8, netty-bom → 4.1.135.Final, `apk upgrade` in Dockerfiles
#267	`Security_Image_Scan` failed	14 HIGH CVEs in netty (after jackson fix)	Explicit netty-bom override in `build.gradle.kts`
#269	`Deploy_Demo` flagship UAT failed	`deploy_demo` job used `checkout: none` → `$(Build.SourcesDirectory)/e2e` didn't exist for Playwright test script	Added `checkout: self` before UAT step
#272	**SUCCESS**	All stages green	N/A

Each failure was a real bug (not flakiness), and each fix made the system more correct.

CVE Fixes (Image Security)

Before: 16 HIGH jackson-databind CVEs + 14 HIGH netty CVEs + 2 HIGH p11-kit Alpine CVEs
After (run #272): CLEAN — Trivy scan returned exit-code 0 for all 4 images

Dependency bumps:

- `jackson-databind` → `2.18.8` (latest, no HIGH/CRITICAL CVEs)
- `netty-bom` → `4.1.135.Final` (all netty-* artifacts now inherit this version, CVEs resolved)
- `apk upgrade` in all Dockerfiles (p11-kit 0.25.6-r0 → patched)

Production Deploy Path (Prepared, Not Yet Live)

Service Principal: `qody-azdo-pipeline-prod` (separate identity from demo's `qody-azdo-pipeline`)
Authentication: Workload Identity Federation (no static secret)
Scope: Least-privilege — `Azure Container Apps Contributor` on `rg-qody-prod` (not full `Contributor`), `AcrPush` on `qodyprodacr` only
Securion Verdict: **PASS** (v2 proposal, 2 blocking findings fixed, 4 required verifications folded in)

Deploy_Prod stage: Drafted at `infrastructure/pipeline-proposals/deploy-prod-stage.yml.proposal` (deferred from live `azure-pipelines.yml`) — includes:

- Separate prod images (`prod-$(SHORT_SHA)` in `qodyprodacr`, different build-args from demo)
- Trivy scan for prod images (inside `Deploy_Prod` stage, before canary/update steps)
- Manual-only trigger (`Build.Reason: Manual`, not auto-trigger on tag push)
- ADO environment `qody-prod` with **manual approval** (approver: alem@alai.no) + **branch control** (main-only)

Status: Proposal only, NOT executed. Activation = separate task requiring CEO approval.

Full proposal (with Securion review history): `/Users/makinja/system/evidence/104863/d2-prod-sp-proposal.md`

Deploy Path Truth (As of 2026-07-07)

Path	Status	Evidence
Manual run with `deployDemo=true`	✅ **PROVEN** end-to-end	Build #272 (all stages green, flagship UAT PASS)
Semver tag `v*` from `main`	⚠️ **Analytically sound**	Tag/branch ancestry checked + trigger semantics verified, but NOT empirically confirmed (no executed tag run post-fix). Recommend: watch the first real `v*` tag deploy as final validation.

Caveat (Proveo): Run #272 was triggered `reason: manual` with `templateParameters: {deployDemo: "true"}` — NOT a `v*` tag push. The tag-trigger mechanism itself was deliberately not re-tested with a second live deploy (to honor the task's "ONE live validation deploy" constraint). Tag trigger proof rests on static evidence, not an executed tag run.

Evidence Location

Directory: `/Users/makinja/system/evidence/104863/`

Key files:

- `d1-run272-success.md` — Build #272 timeline + live verification curls
- `d1-root-cause-and-fix.md` — Full narrative: hypothesis evolution, ADO API proofs, safety actions (cancelled stale runs)
- `d1-run253-failure.md`, `d1-run265-failure.md`, `d1-run269-failure.md` — Each failure's root cause + demo-health curl (proving no rollback needed)
- `d1-cve-bump.md` — CVE details + dependency versions

- `proveo-validation.md` — Independent Proveo (Angie Jones) verification against primary sources (ADO REST API, live curls, git blobs)
- `d2-prod-sp-proposal.md` — Production SP proposal (Securion PASS, not yet executed)
- `securion-sp-verdict.md` — Securion review history (v1 BLOCK → v2 PASS)

Guardrails Added

1. **RUNBOOK.md D3 guardrail (committed to main):** Every QODY deploy task (demo or prod) MUST cite the Azure DevOps **pipeline run URL** in MC evidence. `mc.js done`/`ready` on a QODY deploy task without a run URL is incomplete evidence (ZAKON PI2).
2. **Revision-suffix collision fix (committed to main):** `Deploy_Demo`'s `qody-api` canary step used `REV_SUFFIX="$(SHORT_SHA)"` — a redeploy of the exact same commit would collide with the existing ACA revision name and fail. Fixed to `REV_SUFFIX="$(SHORT_SHA)-$(Build.BuildId)"` — unique per pipeline run regardless of `SourceVersion`. Mirrored into the deferred `Deploy_Prod` proposal.

Next Steps

1. **First real `v*` tag from `main`:** Watch as empirical confirmation of the tag trigger path (analytically sound, but not yet executed post-authorization-fix).
2. **Prod deploy activation:** Separate task, requires CEO approval. Execute `d2-prod-sp-proposal.md` commands, enable `Deploy_Prod` stage in `azure-pipelines.yml`, verify manual-only + approval gate.
3. **CI contention (optional, out-of-scope for #104863):** The `alai-holding` ADO org shares Microsoft-hosted parallel-job capacity across QODY/Bilko/LumisCare. Consider requesting additional parallelism or self-hosted agents.

Last Updated: 2026-07-07
Author: Skillforge (documentation) + FlowForge (pipeline execution) + John (orchestration)
Proveo Verdict: PASS (all 4 claims verified against primary sources)

QODY — Model fiskalizacije: kartice (Monri) i gotovina (POS kasa lokala) — 2026-08-03

QODY — Model fiskalizacije: kartice (Monri) i gotovina (POS kasa lokala) — 2026-08-03

Status: VAŽEĆI MODEL (CEO odluka 2026-08-03, zamjenjuje prethodni tehnički plan integracije od 2026-08-02) **Vlasnik odluke:** Alem (CEO), utvrđeno u živom razgovoru s Asmirom (SnowIT, operater na terenu), zapisano u MC #106721 **Odnosi se i na:** Bilko (isti model razmišljanja, vidi sekciju 7)

---

1. Sažetak za odluku

1. QODY **ne gradi** fiskalnu integraciju. Fiskalizacija se rješava na dvije druge strane, ne kod nas.
2. **Kartično plaćanje:** Monri izdaje fiskalni račun gostu u trenutku plaćanja karticom. Fiskalizacija ide kroz Monri.
3. **Gotovina:** vlasnik lokala je po zakonu dužan isprintati fiskalni račun na SVOJOJ POS kasi. To je njegov uređaj i njegova zakonska obaveza, ne QODY-jeva.
4. Monri već ima vlastitu integraciju s POS kasama kafića/restorana i tu integraciju **oni** završavaju — nije naš inženjerski posao.
5. **Naša obaveza je dokumentacija i obavještavanje korisnika** (onboarding), ne izgradnja fiskalne veze.
6. Posljedica: faze 1–3 iz jučerašnjeg tehničkog plana integracije s fiskalnim terminalima **se ne grade** (vidi sekciju 4).
7. Pravni temelj ostaje isti kao jučer: obveznik fiskalizacije je **lokal**, ne QODY/SnowIT — ovaj model to samo dodatno potvrđuje jer QODY nigdje ne dodiruje fiskalni tok.
8. Dvije stavke iz koda ostaju otvorene bez obzira na ovu odluku (pay-at-table potvrda gotovine, refund bez veze na fiskalni broj) — vidi sekciju 6.

2. Model po na?inu pla?anja

Način plaćanja	Ko fiskalizuje	Uređaj/sistem	Šta radi QODY
Kartica (online, kroz QODY)	**Monri**	Monrijev sistem, u trenutku autorizacije plaćanja	Prosljeđuje narudžbu i plaćanje Monriju; ne generiše, ne prosljeđuje i ne čuva fiskalni broj — to je Monrijev dokument, ne naš
Gotovina (plaćanje na licu mjesta)	**Vlasnik lokala**	Lokalova vlastita, već postojeća (ili zakonski obavezna) POS/fiskalna kasa	Prikazuje narudžbu osoblju radi ručnog unosa na kasu; QODY-jeva digitalna "potvrda narudžbe" NIJE i ne zamjenjuje fiskalni račun
Pay-at-table (odgođeno gotovinsko)	Vlasnik lokala, u trenutku naplate	Lokalova POS kasa	Isto kao gotovina — uz napomenu da trenutni kod nema eksplicitnu potvrdu "gotovina primćena" (gap, sekcija 6)

Nema scenarija u kojem QODY sam generiše, šalje ili čuva fiskalni broj. To je namjerno — vidi sekciju 3, zašto je to i pravno poželjno.

---

3. Pravni okvir ukratko

- **Obveznik fiskalizacije je lokal (ugostiteljski objekat), ne platforma.** Ovo stoji identično u oba entiteta: Zakon o fiskalnim sistemima FBiH (Sl. novine FBiH 81/09), član 4 stav 1, i Zakon o fiskalizaciji RS (Sl. glasnik RS 15/22), član 4 stav 1 — obveznik je "lice registrovano za promet robe/usluga klijentima". Ugostiteljstvo nije na listi izuzeća ni u jednom od entiteta.
- **Monri "Model B" arhitektura (novac ide direktno lokalu, QODY/SnowIT provizija fakturiše se odvojeno) je pravni štit** — ona drži lokal kao jedinog obveznika fiskalizacije/prometa. Ako bi se ikad prešlo na model gdje novac prvo ide na QODY/SnowIT nalog pa se prosljeđuje lokalu (pooled/marketplace plaćanje), rizik da SnowIT/QODY postane obveznik raste. **Zaključak: Model B se ne mijenja bez pravne provjere.**
- **Kazne po entitetima (razlog zašto onboarding mora biti izričit, ne pretpostavljen):**
  - FBiH: 2.500–20.000 KM za pravno lice (član 52 st. 1(b)), uz zabranu rada 6–12 mjeseci kod ponovljenog prekršaja (član 52 st. 4). Kažnjivo je već i obavljanje djelatnosti prije fiskalizacije, ne samo pojedinačni neizdat račun.
  - RS: 8.000–24.000 KM za pravno lice, uz privremenu zabranu rada 30 dana (prvi prekršaj) do 60 dana (ponovljeni prekršaj u roku od godinu dana) — član 17–18, 20

Zakona o fiskalizaciji RS.
    - Kazne padaju na **lokal**, ne na SnowIT/ALAI — ali reputacioni i ugovorni rizik za nas je realan ako lokal nije jasno upozoren.
- **QODY-ov digitalni "račun" gostu nije fiskalni dokument** — ne po FBiH (član 33 — fiskalni račun mora biti izdat preko odobrenog, fiskaliziranog uređaja) ni po RS zakonu (član 6, isti princip). Ovo važi bez obzira na način plaćanja, uključujući kartično — i kad Monri izda pravi fiskalni račun, QODY-jeva vlastita "potvrda narudžbe" ostaje interni dokument, ne fiskalni.
- **Nova FBiH regulativa (2026, Zakon o fiskalizaciji transakcija)** uvodi softverski/cloud model (ESET) koji je arhitektonski bliži QODY-ju, ali se ne primjenjuje odmah — primjena najkasnije 18 mjeseci od stupanja na snagu, uz prelazni period do 4 godine gdje stari uređajni sistemi i dalje važe. Danas (2026) FBiH lokali rade po starom modelu. Ovo ne mijenja današnju odluku — spominje se jer neko za godinu-dvije treba znati da postoji, kad Monri/dobavljači fiskalnih sistema budu ažurirali svoje integracije.

---

4. Šta NE gradimo i zašto

Jučerašnji tehnički dokument (`tehnicka-integracija-fiskalni-terminali-2026-08-02.md`, isti MC #106721) predlagao je da QODY sam gradi integraciju s fiskalnim uređajima/operaterima (model "(b)" iz pravnog memoa: lokal fiskalizuje, QODY šalje nalog preko API-ja). Taj plan je pretpostavljao da MI gradimo tehničku vezu prema RS OFS operateru i FBiH fiskalnim uređajima. **Ta pretpostavka je danas ispravljena.**

Otpisano, eksplicitno, da niko za mjesec dana ne krene ovo graditi:

- **Faza 1 (RS — OFS integracija, `fiscal_receipt` tabela, outbox dispatcher, GTIN mapiranje meni stavki)** — ne gradi se. Monri/lokalova POS kasa rješava fiskalizaciju gotovine i kartice; QODY ne treba vlastitu vezu prema RS OFS uređaju.
- **Faza 2a (FBiH — integracija sa starim uređajnim fiskalnim sistemom preko partnera koji je trebalo identifikovati)** — ne gradi se, iz istog razloga.
- **Faza 2b (FBiH — budući ESET/CPF cloud model)** — ne gradi se sada; ostaje kao nešto što treba pratiti kad postane primjenjivo (za ~1,5–2 godine), ali nije naš posao da ga danas gradimo.
- **Faza 3 (pravi fiskalni QR račun gostu, zamjena internog `RCT-...` formata)** — ne gradi se, jer QODY nikad ne postaje izdavalac fiskalnog dokumenta.

Šta OSTAJE, jer je nezavisno korisno i nije otpisano ovom odlukom (Faza 0 iz jučerašnjeg plana, sada preformulisana kao "onboarding i higijena podataka", ne "priprema za fiskalnu integraciju"):

- MC #106738 (M) — preimenovanje QODY digitalnog "Računa" u "Potvrda narudžbe" u guest UI-u, i18n stringovi. Ostaje otvoreno i vrijedno — razlog se nije promijenio: taj dokument nikad nije bio fiskalni račun, bez obzira gradi li QODY integraciju ili ne.

- MC #106739 (M) — PDV kolone u `export.csv` za knjigovođu (podaci već postoje u bazi, samo se ne izvoze) + čitanje PDV stope iz konfiguracije umjesto hardkodiranog "17%". Ostaje otvoreno, nezavisno od modela fiskalizacije.
- `venue.bih_entity` (FBiH/RS/Brčko) polje u shemi — i dalje korisno za tačno adresiranje onboarding poruke po entitetu (kazne i rokovi se razlikuju), iako QODY ne šalje ništa fiskalnom uređaju.

---

5. Obaveze prema korisniku (onboarding)

Ovo mora biti eksplicitna stavka u onboarding checklisti za svaki novi *plaćeni* lokal, ne pretpostavka da vlasnik "to već zna":

1. **Gotovinski promet naplaćen kroz QODY mora biti fiskalizovan na vašoj vlastitoj POS/fiskalnoj kasi**, u trenutku naplate. Ovo je vaša zakonska obaveza (FBiH: Zakon o fiskalnim sistemima 81/09; RS: Zakon o fiskalizaciji 15/22), ne QODY-jeva.
2. **QODY-jev digitalni "račun"/"potvrda narudžbe" NIJE fiskalni dokument.** On prikazuje narudžbu i informativan iznos PDV-a, ali ga ne izdaje fiskalni uređaj i on nema pravni status poreskog dokumenta ni u FBiH ni u RS.
3. **Za kartično plaćanje kroz QODY, fiskalni račun izdaje Monri** u trenutku autorizacije plaćanja — provjerite s vašim Monri predstavnikom da je vaš POS/fiskalni uređaj ispravno povezan s Monrijevim sistemom prije nego počnete primati kartična plaćanja kroz QODY.
4. **Neizdavanje fiskalnog računa nosi novčanu kaznu i, kod ponovljenog prekršaja, privremenu zabranu rada** (FBiH: 2.500–20.000 KM + zabrana 6–12 mjeseci; RS: 8.000–24.000 KM + zabrana 30–60 dana). Ova odgovornost je na vama kao vlasniku lokala, ne na SnowIT-u/QODY-ju.
5. Preporuka: potvrdite sa svojim knjigovođom/poreznim savjetnikom da je vaša fiskalna kasa spremna za promet koji dolazi kroz QODY (i gotovinski i kartični), prije lansiranja plaćenog rada.

Ovaj tekst je namijenjen direktnom korištenju u onboarding checklisti i/ili merchant ugovoru (uz pravnu redakciju formulacije u samom ugovoru — vidi sekciju 6, pitanje 5 iz brief-a za poreznog savjetnika).

---

6. Otvoreno / za potvrdu

Za pisanu potvrdu (nije naš pravni nalaz — vidi napomenu o izvoru ispod):

> **Izvor: SnowIT/Asmir, 2026-08-03** — tvrdnja da Monri izdaje fiskalni račun gostu kod kartičnog plaćanja dolazi od Asmira (SnowIT, operater na terenu s direktnim kontaktom kod Monrija), iznesena u živom razgovoru s CEO-om. **Traži se pismena potvrda od Monrija**, uz produkcijski merchant ugovor, prije nego se ovaj model tretira kao ugovorno/pravno utvrđen za prve plaćene lokale.

Za poreznog savjetnika (puna lista od 8 pitanja: `brief-porezni-savjetnik-2026-08-02.md`, isti MC) — najbitnija tri u svjetlu ovog modela:

- Da li naša Monri postavka (novac direktno lokalu) i dalje drži lokal kao *jedinog* obveznika fiskalizacije, i potvrđuje li to Monrijev produkcijski merchant ugovor u praksi.
- Da li QODY/SnowIT time što gost prolazi kroz QODY UI (a fiskalizuje Monri/lokal) preuzima ikakvu regulatornu obavezu — ili ostaje čist tehnički kanal.
- Formulacija u Merchant Agreement-u koja provedivo prebacuje odgovornost za fiskalizaciju gotovine i kartice na lokal.

Gapovi u našem kodu koji ostaju otvoreni bez obzira na ovu odluku (ne mijenjaju se time što QODY ne gradi fiskalnu integraciju — ovo su gapovi u samom order/payment toku):

- **(a) Pay-at-table nema staff potvrdu "gotovina primljena".** `pending_cash` status se postavlja pri kreiranju narudžbe, ali ne postoji eksplicitan endpoint kojim osoblje potvrđuje da je gotovina zaista naplaćena i da treba fiskalizovati na kasi. Ovo je operativni/proizvodni gap, nezavisan od modela fiskalizacije — treba zatvoriti radi pouzdanog zatvaranja narudžbi, ne samo radi fiskalizacije.
- **(b) Refund je stub i nema veze s fiskalnim brojem.** `RefundService` mijenja status plaćanja, ali gateway poziv je trenutno placeholder ("STUB-{uuid}") i `RefundRequestTable` nema polje koje referencira originalni fiskalni/Monri broj. Kad Monri produkcijski refund bude uveden, ovo treba riješiti zajedno s njim — trenutno nije blokator jer refund tok uopšte nije proizvodni.

---

7. Šta ovo zna?i za Bilko

Isti model razmišljanja primjenjuje se i na Bilko, i CEO je izričito naglasio (MC #106721, 2026-08-03) da će ovaj dokument trebati kao referenca tamo:

- **Bilko nije, i ne postaje, obveznik fiskalizacije za svoje korisnike.** Isto pravno polazište kao kod QODY-ja — obveznik je firma/preduzetnik koji koristi Bilko, ne softver koji vodi njeno knjigovodstvo.
- **Bilko-ova uloga je da dokumentuje i podsjeća** (rokovi, obaveze, izvještaji), ne da fiskalizuje promet u ime klijenta. Ovo je direktno u skladu s postojećim Bilko katalogom obaveza — vidi `docs/regulatory/OBLIGATION-CATALOG-DRAFT.md` u Bilko repozitoriju (SSOT za periodične/uslovne obaveze po jurisdikciji, CEO odobreno 2026-07-14, MC #105640).

- **Vezano, ali odvojeno:** MC #106761 pokreće širi projekat regulatornog sistema za 4 jurisdikcije (HR, FBiH, RS-entitet, Srbija) — agenti/knowledge-sourcing za praćenje izmjena zakona. Dio te inicijative (radni naziv "SOURCING-RULES") razvija se na grani `codecraft/106761-sourcing-rules` u Bilko repozitoriju — **ta grana još nije mergovana u main**, pa se ovdje ne linkuje kao gotov dokument, samo kao rad u toku koji treba pratiti (`node ~/system/tools/mc.js show 106761`).
- Zajednička poruka za oba proizvoda: **softver ne postaje fiskalni/poreski obveznik umjesto klijenta** — ostaje alat koji naplaćuje/knjiži/podsjeća, dok zakonska odgovornost i potpis ostaju kod licenciranog savjetnika i kod samog obveznika.

Izvori

- `~/system/evidence/106721/pravni-memo-fiskalizacija-bih-2026-08-02.md` — pravni okvir (FBiH Zakon 81/09, RS Zakon 15/22, kazne, Monri Model B)
- `~/system/evidence/106721/tehnicka-integracija-fiskalni-terminali-2026-08-02.md` — tehnički kontekst RS/FBiH sistema i kodni gapovi (preporučeni model integracije iz ovog dokumenta je NADOMJEŠTEN ovom CEO odlukom; koriste se samo činjenične tvrdnje o RS/FBiH sistemima i o gapovima u kodu)
- `~/system/evidence/106721/brief-porezni-savjetnik-2026-08-02.md` — 8 pitanja za poreznog savjetnika
- MC #106721 — puna istorija odluke, uklj. CEO odluku od 2026-08-03 zapisanu verbatim (`node ~/system/tools/mc.js history 106721`)
- MC #106738, MC #106739 — quick-win zadaci koji ostaju otvoreni nezavisno od ove odluke
- MC #106761 — širi regulatorni sistem projekat (Bilko), spomenut u sekciji 7
- `/Users/makinja/business/ALAI-Holding-AS/products/Bilko/docs/regulatory/OBLIGATION-CATALOG-DRAFT.md` — Bilko katalog obaveza (SSOT)

Dokument pripremljen 2026-08-03 na osnovu žive CEO odluke i postojeće pravne/tehničke analize istog MC-a. Zamjenjuje pretpostavku "QODY gradi fiskalnu integraciju" iz jučerašnjeg tehničkog plana.

Privacy Policy v2.0 implementacija (pravnik Ilhan Erma) — MC #106821, 2026-08-04

MC #106821 · Status: **Implementirano, spremno za deploy-gate** · 2026-08-04 · Grana `fix/106821-privacy-policy`, worktree `~/business/ALAI-Holding-AS/products/qody-106821-privacy`

1. Kontekst

Pravnik Ilhan Erma je mailom (posredstvom Asmira, 2026-08-04) dostavio finalni tekst Politike privatnosti koji zamjenjuje interni draft v1.0 iz MC #104526 (koji je nosio napomenu „podložno pravnoj reviziji“). Izvor istine za sadržaj je repo fajl `docs/legal/politika-privatnosti-pravnik-2026-08-04.md` — sav frontend tekst je preslikan iz njega, ne parafraziran.

2. Šta je promijenjeno

Implementacija pokriva četiri deliverable-a, u jednom PR-u (commitovi `6af2683` inicijalna implementacija + `2777887` fix nakon peer-review nalaza), na grani `fix/106821-privacy-policy`:

1. **Landing stranica Politike privatnosti v2.0** (`apps/landing/politika-privatnosti/index.html`) — svih 11 sekcija pravnikovog teksta, DPO „Ilhan Erma, licencirani pravnik“, kontakti `privacy@qody.ba` (sekcije 1 i 8) i `dpo@qody.ba`, datum ažuriranja 2026-08-04, TOC ažuriran, string „podložno pravnoj reviziji“ uklonjen, postojeći `legal.css` layout zadržan.
2. **Footer na sve 4 landing stranice** (`index.html`, `politika-privatnosti`, `politika-kolacica`, `uvjeti-koristenja`) — standardizovan tekst „© 2026 SnowIT d.o.o. — QODY. Sva prava zadržana.“ + 4 linka (Uslovi korištenja | Politika privatnosti | Politika kolačića | Kontakt).
3. **CookieBanner** (`apps/guest/src/components/CookieBanner.tsx`) — naslov „Vaša privatnost nam je važna“ + pravnikov tekst, tri dugmeta (Prihvati sve / Odbij opcionalne / Postavke kolačića), consent state perzistiran u `localStorage`, nijedan opcioni skript se ne aktivira prije saglasnosti, link na politiku kolačića.

4. **InviteAcceptScreen** (`apps/admin/src/InviteAcceptScreen.tsx`) — obavezan checkbox „Pročitao/la sam i prihvatam Uslove korištenja QODY platforme.“ (submit blokiran dok nije čekiran), info linija s linkom na Politiku privatnosti, odvojen opcioni checkbox za marketing saglasnost (default neoznačen). Nigdje se ne koristi zabranjena fraza „prihvatam Politiku privatnosti“.

3. Verifikacija (Writer ? Witness)

Nezavisna peer-verifikacija (Angie Jones / Proveo persona, odvojena sesija od buildera) prošla je kroz dva kruga:

Round 1 — PARTIAL, 5 nalaza:

- UTF-8/ćirilica korupcija u TOC-u („2.Ције podatke prikupljamo?“ umjesto „2. Koje podatke prikupljamo?“) — greška heredoc encodinga pri inicijalnom pisanju fajla.
- Footer na `apps/landing/index.html` nije bio ažuriran iako je AC2 tvrdio suprotno.
- Rečenica o kartičnim podacima bila parafrazirana umjesto doslovnog citata pravnika.
- Mrtvi i18n fallback u `InviteAcceptScreen.tsx` — `t("admin.invite.tosRequired") || "..."` nikad ne bi pao na fallback jer nepostojeći i18n ključ vraća istinit (truthy) string.
- Build gate blokiran jer je MC owner bio „john“ umjesto „codecraft“, pa sankcionisani `run-build.sh` runner nije mogao izvršiti buildove.

Fix ciklus (commit 2777887): sva 4 sadržajna nalaza ispravljena (heredoc encoding root-cause, footer standardizovan na svih 4 stranice, kartična rečenica vraćena na doslovan pravnikov tekst, i18n fallback zamijenjen literalnim stringom), owner promijenjen na codecraft → buildovi izvršeni: guest app 283 modula, admin app 1947 modula, oba PASS bez TypeScript grešaka.

Round 2 — PASS: svih 5 nalaza nezavisno reprodukovano kao popravljeno (verifikator sam ponovo pokrenuo buildove — byte-identični brojevi modula/bundle veličina, ne samo pročitan builderov transkript). Uživo Playwright testiranje protiv stvarnih `dist/` bundlova: cookie baner (3 dugmeta po role/label, settings panel, perzistencija preko reloada, 3 distinktna consent stanja u localStorage), invite ekran (mockovan peek endpoint, submit dugme disabled→enabled tačno na ToS checkbox, marketing checkbox potvrđeno neoznačen po defaultu, pravnikov tekst doslovan u DOM-u). Pun diff commit-a 2777887 pročitan direktno — potvrđeno da nijedan drugi paragraf nije dirnut. Zabranjene fraze i provjera odsustva analytics/marketing skripti i dalje čiste.

4. DEPLOY GATE — JOŠ NIJE DEPLOYANO

Implementacija je verifikovana lokalno (build + Playwright protiv lokalno serviranih bundlova), ali deploy je izričito van scope-a ovog MC-a i čeka sljedeće uslove:

- **Email adrese moraju postojati živo:** `privacy@qody.ba` i `dpo@qody.ba` (CF Email Routing) — čeka odluku CEO-a/Asmira o destinaciji poruka.
- **Politika kolačića i Uslovi korištenja i dalje sadržajno nedovršeni:** obje stranice još nose napomenu „Verzija 1.0 (2026-06-30) — podložno pravnoj reviziji“ — footer im je ažuriran u ovom MC-u, ali sadržaj čeka odvojen pravnikov pregled.
- **PI2 deploy verifikacija** mora se odraditi kao poseban zadatak nakon deploya (landing = CF Pages `qody-landing`; guest/admin = ACA s `VITE_API_BASE_URL` build-argom) — browser/production dokaz, ne samo build zeleno.
- Puni invite-accept submit round-trip (kreiranje naloga) nije end-to-end testiran — van pravnog/consent scope-a ovog MC-a, izričito deklarisano kao netestirano, ne kao PASS.

5. Follow-up

MC #106826 — otvoren za dodavanje marketing-consent polja na backend invite-accept endpoint (trenutno checkbox state postoji samo lokalno u formi jer backend polje ne postoji; nije izmišljen backend ugovor).

Evidence: `/Users/makinja/system/evidence/106821/` *(dispatch-plan.md, s1-fix-report.md, s2-verify-report.md, s2-verify-report-round2.md, p2p-native-verify-transcript.md, screenshotovi i JSON izvještaji Playwright testova).*

VAT-inclusive pricing fix (PDV ura?unat u cijene) — MC #106829, 2026-08-04

MC #106829 · Status: **Implementirano lokalno, NIJE pushano/deployano** · 2026-08-04 · Grana `fix/106829-vat-inclusive`, worktree `~/business/ALAI-Holding-AS/products/qody-106829-vat`, baza `azdo/main @ 0f0acf4`

1. Kontekst

CEO direktiva 2026-08-04 (doslovno): „ako meni na qody stavi 3 km kafa mi ne dodajemo pdv — pdv je obracun u tu cjenu“. Cijene na meniju su BRUTO (PDV 17% je već uračunat) — kafa od 3 KM ostaje 3 KM za gosta, ne 3.51 KM. Postojeći kod je radio suprotno: `OrderService.kt` je PDV dodavao POVRH cijene (add-on formula, `lineTax = lineTotal * taxRate`, zaokruživano na 4 decimale po liniji), što je gosta naplaćivalo više nego što je meni pokazivao, a usput je i platformska provizija (`SuperAdminService.kt`) računata na tu naduvanu osnovicu — što znači da je ALAI od lansiranja naplaćivao lokale na precijenjenoj osnovi.

2. Šta je promijenjeno (D1–D6)

Implementacija je prošla kroz forged-prompt proces (panel od 5 eksperata → mehanik CLEAR TO DISPATCH → S1 build → S2 nezavisna verifikacija), commitovi `58e6b84` (glavna implementacija) + `e922518` (fix lažno-pozitivnog grep matcha u D5 komentaru), 18 izmijenjenih fajlova. **NIJE pushano na azdo/origin** — commitovi su samo lokalni na grani.

- **D1 — ekstrakcija PDV-a per-rate-group:** `OrderService.validateCart` grupiše linije korpe po `item.taxRate`, sabira BRUTO iznos po grupi, PDV izdvaja formulom `rate/(1+rate)` izračunatom u letu (nikad perzistiranom kao multiplikator, jer 17/117 skraćuje se na `NUMERIC(5,4)`), zaokružuje na 2 decimale PO GRUPI STOPE (ne 4dp po liniji kao ranije — BiH fiskalna praksa zaokružuje na nivou računa/grupe stope). `total` ostaje nepromijenjeni bruto iznos s menija; `subtotal` postaje osnovica (`total - taxTotal`). Testni slučaj: 3.00 KM stavka @ 0.17 → `taxTotal=0.44`, `total=3.00` (ne 3.51).
- **D2 — era diskriminator u šemi:** nova migracija `V30__vat_inclusive_pricing_era.sql` dodaje `"order".pricing_model` (backfill postojećih redova na `'v1_addon'` PRIJE nego default postane `'v2_inclusive'`); `order.version` (optimistic-lock brojač) namjerno NIJE

preimenovan/reupotrijebljen za ovu svrhu.
- **D3 — receipt/CSV lockstep:** umjesto izvođenja `vatRate` iz `taxTotal/subtotal` pri čitanju (fragilno preko era granice), `OrderService.submitOrder` sada snimi `tax_rate_snapshot` JEDNOM pri kreiranju narudžbe; `ReceiptService` čita taj snapshot direktno, više ne dijeli iznova.
- **D4 — billing korekcija (bez tihe historijske izmjene):** `tax_total` i `pricing_model` dodani u CSV izvoz (`EnhancedSalesReportService` — stvarna lokacija exporta, forged-prompt je pogrešno citirao `SalesReportService.kt`, greška disclosed i dokumentovana u kodu); `SuperAdminService.kt` dobija opširan komentar koji eksplicitno iznosi nalaz o precijenjenom obračunu provizije i tačan upit za rekonstrukciju raspona, uz eksplicitno upućivanje na OPEN CEO DECISION #1 — nikakva automatska korekcija historijskih naplata nije izvršena.
- **D5 — guest/admin frontend lockstep:** `apps/guest/src/format.ts::cartSubtotal()` verifikovano ne radi nikakvu poresku matematiku (0 matcheva za rate/formula), pa strukturalno ne može divergirati od backend formule; ostavljen s komentarom koji upućuje na `OrderService.kt`. `CartPage.tsx`/`CheckoutPage.tsx`/`ReportsView.tsx` verifikovano (ne pretpostavljeno) da koriste API vrijednosti bez lokalnog preračunavanja.
- **D6 — testovi:** svi postojeći testovi koji su tvrdili staru add-on formulu ažurirani na PDV-uključene iznose (uključujući stvarnu regresiju uhvaćenu tokom rada: hardkodovan iznos 11.70 umjesto 10.00 u dva test fajla). Novi test fajl `VatInclusivePricingIntTest.kt` (5 testova): jednostruka stopa, mixed-rate (dvije različite stope u istoj korpi, dokazano da se NE blendaju), idempotency replay preko era granice, group-rounding kontrast (2.18 grupno vs 2.19 naivno po liniji). Rezultat: **107 Kotlin integracionih/unit testova, 0 failures, 0 errors**; guest vitest 30/30; staff-kitchen vitest 22/22; admin 4 pre-postojeća faila (nepovezana s ovim MC-om, root-uzrok potvrđen — testni fajl i njegova zavisnost netaknuti u grani).

3. Verifikacija

Put: panel od 5 eksperata (petter-graff, bilko-racunovodstvo-hr, markos-zachariadis, bruce-momjian, devils-advocate) → mehanik **CLEAR TO DISPATCH** → S1 build → S2 nezavisna verifikacija (Angie Jones/Proveo persona, odvojena sesija, Writer≠Witness).

S2 VERDICT: PASS — verifikator je nezavisno reprodukovao SVE acceptance signale (ne vjerujući builderovim ispisima): ponovo pokrenuo `./gradlew integrationTest` i sam sabrao JUnit XML rezultate (107/0/0, tačno poklapanje), ponovo pokrenuo guest/kitchen vitest testove, ručno preračunao PDV matematiku prije čitanja test asertacija (3.00→0.44/2.56 osnovica; grupno zaokruživanje 2.18 vs naivno 2.19; mixed-rate 3.45 vs pogrešno-blendovano 3.47 — sve se poklopilo), root-uzrokovao 4 pred-postojeća admin test faila preko git diff-a (grana dira samo `types.ts` u admin app-u), provjerio odsustvo kolizije verzije migracije, tražio UTF-8 korupciju (nije nađena).

3–4 disclosed gapa (nijedan blokirajući):

1. `SalesReportService.kt` (JSON izvještaj) ne grana eksplicitno na `pricing_model` za agregatne sume — matematički odbranjeno komentarom u kodu (sabiranje stvarno naplaćenih iznosa je uvijek validno), ali tehnički djelimično, ne potpuno, ispunjava D2-ov zahtjev „MUST branch“.
2. Mixed-rate narudžbe dobijaju JEDAN blendovan `tax_rate_snapshot` na nivou narudžbe (ne breakdown po stopi) jer `ReceiptDto` ima samo jedno `vatRate` polje — nije regresija (postojeće arhitekturno ograničenje), ali relevantno tek kad BiH lokal stvarno ima više PDV stopa (danas sva live podaci 17%, nije nezavisno potvrđeno protiv žive baze).
3. `ReportsView.tsx` „bez lokalnog preračuna“ potvrđeno samo preko `git diff --stat` (fajl netaknut u grani), ne direktnim čitanjem sadržaja od strane verifikatora.
4. Live DB probe (distinct `tax_rate` vrijednosti, broj historijskih narudžbi) — odbijen dozvolskim slojem sesije i builderu i verifikatoru identično; ostaje otvoreno za deploy/PI2 fazu.

4. OPEN CEO DECISIONS (4, nijedna auto-riješena)

1. **Historijski kredit lokalima za platformsku proviziju** — ALAI je proviziju obračunavao na precijenjenoj (add-on) osnovici za svaku `v1_addon`-era naplatu. CEO treba odlučiti: retroaktivni kredit, jednokratni otpis, ili bez akcije, i za koji vremenski raspon (upitno tek nakon što D2-ov diskriminator omogući upit).
2. **Rekonstrukcijski izvještaj za historijske narudžbe gostiju** — obavezan ili ne? Prag zavisi od broja pogođenih historijskih narudžbi (probe nije mogao biti izvršen zbog dozvolskog sloja) — odvojeno pitanje od odluke #1 (to je ALAI-jeva provizija, ovo je pitanje gostijskih povrata).
3. **PDV tretman napojnice/tipa po BiH zakonu** — panel eksplicitno NIJE mogao potvrditi (HR-persona odbila certificirati BiH praksu izvan svoje domene). Praćeno zajedno s postojećim taskom #106745 (BiH poreski savjetnik) kao njegovo 9. otvoreno pitanje.
4. **Stripe/Monri verifikacioni prag prije produkcije** — ovo mijenja stvarne naplaćene iznose, ne samo prikaz. CEO treba odlučiti koji verifikacioni gate (npr. N uspješnih test-mode naplata pregledanih od strane čovjeka) je potreban prije promovisanja D1–D5 u produkciju, odvojeno od D6-ove automatske test pokrivenosti.

5. Follow-up

MC #106830 — mixed-rate snapshot: proširiti `ReceiptDto`/snapshot mehanizam da podrži breakdown po stopi umjesto jednog blendovanog `vatRate` polja, relevantno kad BiH lokal stvarno uvede više PDV stopa.

Evidence: `/Users/makinja/system/evidence/106829/` *(dispatch-plan.md, panel/1-5*.md, s1-build-report.md, s2-verify-report.md, p2p-native-verify-transcript.md); forged prompt:* `/Users/makinja/system/prompts/forged/106829.md`*.*

QODY CI gate — build-validation policy blocking (MC #106804, 2026-08-05)

QODY CI gate — build-validation policy sada BLOKIRA merge (MC #106804)

Datum: 2026-08-05 · **Izvršio:** FlowForge (dispatch John) · **Nezavisno verifikovao:** verifier agent (6/6 PASS, živi REST pozivi)

Šta je promijenjeno

- Azure DevOps, repo QODY, grana `refs/heads/main`: **Build validation policy prebačena s `isBlocking=false` na `isBlocking=true`** (policy id=2, revision 1→2).
- Od sada se **nijedan PR ne može completati dok "CI Gates" validation build ne prođe** — ručni complete prije builda vraća `GitPullRequestUpdateRejectedByPolicyException`.

Zašto

2026-08-03 je PR #301 (payment fix) mergovan **ručno bez ijednog uspješnog validation builda** — policy je postojao ali je bio savjetodavan (`isBlocking=false`). Uz to je CEO primao ~30 "canceled build" mailova jer agentska velocity mergeva otkazuje superseded buildove. Puna istraga: MC #106804, forge dokument `~/system/prompts/forged/106804.md`.

Živi dokaz (ne "green build" nego stvarni pokušaj proboja)

- Probni PR #316: pokušaj complete PRIJE builda → **ODBIJEN** policy-jem; PR abandonovan, probni branch obrisan, main HEAD netaknut (`956f9f35`).
- Bypass audit: nijedan ACE na repou nema `PolicyExempt` / `PullRequestBypassPolicy`.
- Evidence: `~/system/evidence/106804/{d0-final,d2,peer-verify}/` (before/after JSON, response JSON-i, nezavisni peer verdikt sa sha256 hashevima).

Šta NIJE ura?eno (svjesno) i zašto

1. **Reviewer policy (min 1, bez self-approve) NIJE upaljena**: org ima **samo jedan ljudski ADO identitet** (Alem Basic — svi agenti rade kroz njegov PAT), pa bi obavezan nezavisni reviewer blokirao 100% mergeva. Čeka drugi identitet → vezano za Entra povezivanje (#106841, pauzirano CEO odlukom "Ne" 2026-08-05).
2. **2. self-hosted parallel slot NIJE kupljen**: ADO org nema podešen billing (nije vezan za Azure subscription, korijen: org nije povezan s Entra tenantom). CEO odluka: ne raditi Entra plan za sada. QODY CI ovim NIJE pogođen (ide na MS-hosted pool); Bilko cancel-lanac se rješava throttlingom (#106809).

Otvoreni rizici (za budu?e hardening odluke)

- Alemov PAT identitet drži `EditPolicies` + `ManagePermissions` na QODY repou → **isti identitet koji merga može i ukinuti policy**. Gate je stvaran, ali samoodbranjiv nije — pravi fix traži odvojene identitete (Entra).
- Company Mesh peer-verify infrastruktura (lokalni Ollama) je srušena — peer-verifikacija za ovaj task rađena native-agent putem; Ollama treba restart/popravku.

Povezani taskovi

#106804 (ovaj), #106809 (Bilko main validacija), #106810 (Rollback-Rehearsal istraga), #106811 (mail digest — TEK nakon punog merge ciklusa s novim gate-om), #106841 (Entra, pauzirano), #106802 (slot, blocked), #106815 (PAT rotacija).

Dopuna 2026-08-05 — istraga 2 pala builda (MC #106810)

Dva "failed main" runa od 2026-08-03 (20:18, 20:35) NISU glavni CI: to je ručni **stage-rollback rehearsal alat** (`Bilko-Gate0-Stage-Rollback-Rehearsal`, defId=4) koji dira samo stage ACA app-ove.

Pao je na vlastitom query bugu (`az containerapp revision list` bez `--all` → prazan rezultat → fail-closed). Bug ispravljen 2026-08-04, run 20260804.4 SUCCEEDED. "Requested for Alem Basic" je PAT artefakt. Nema buildNumber kolizije (obje definicije koriste ADO default brojanje). Zaključak: resolved-in-place. Evidence: `~/system/evidence/106810/istraga-2026-08-05.md`.

QODY API revision retention and DB connection budget — MC #107292

QODY API revision retention and DB connection budget — MC #107292

Status: Published to QODY BookStack on 2026-08-17 **BookStack location:** `https://docs.alai.no/books/qody` **Source commit:** `b6062f2500fd4074023a9a0aa120af0fb54c3a9d` **Incident:** 2026-08-17, `qody-prod-db` active connections reached 46/50 **Evidence:** `/Users/makinja/system/evidence/107291/`

What happened

`qody-api-prod` used Azure Container Apps Multiple revision mode with `minReplicas=1`. Eight active revisions therefore meant eight running API replicas even though seven revisions carried 0% traffic. Each replica opened runtime/admin Hikari pools and a dedicated PostgreSQL LISTEN connection. The stale revisions held 32 idle application sessions and pushed the normal Azure metric baseline to roughly 40–41 connections.

The Sev-1 alert triggered at 46 connections and self-resolved, but the unsafe baseline remained. Emergency mitigation MC #107291 deactivated six old zero-traffic revisions and retained the serving revision plus one rollback. Idle application sessions dropped from 32 to 8; Azure active connections settled near average 16 / maximum 18 while API health and RLS checks remained PASS.

Important metric correction

`ResiliencyRequestsPendingConnectionPool` belongs to the `Microsoft.App/containerApps` proxy/resiliency namespace. It is not a Hikari/JDBC/PostgreSQL pool metric. Its 09:20–09:25 spike did not correlate with the DB connection spike at 10:50 UTC and must not be used as proof of a DB leak. If monitored, use Average/Maximum (never Total), label it diagnostic-only, and correlate it with request retries, timeouts, and latency.

Permanent retention control

After successful API canary promotion in both demo and production, the pipeline:

1. captures `az containerapp revision list -o json`;
2. validates exactly one healthy/provisioned 100%-traffic revision and confirms it is the revision just promoted;
3. protects the revision that served 100% immediately before promotion as the last-known-good rollback;
4. retains any newer 0%-traffic revision as possible functional-smoke evidence, even if its ACA probe state says Healthy;
5. deactivates only healthy 0%-traffic revisions older than the protected rollback.

The pure planner is `infrastructure/scripts/retain-api-revisions.mjs`. It makes no Azure calls and is fixture-tested, including a sanitized real incident revision list. On ambiguous state it exits non-zero before any mutation. Unhealthy/unprovisioned canaries are also retained as incident evidence. Partial deactivation is safe to retry because planning is idempotent. Before a new canary is created, the pipeline also aborts if more than two API revisions are already active. Failed-canary evidence must then be manually triaged and explicitly deactivated before another deploy; it cannot accumulate outside the validated four-replica-slot budget.

This control is API-only: static guest/admin/kitchen MFEs do not own database pools. Their traffic-cutover behavior remains under MC #106744.

Connection-budget invariant

Revision retention and pool sizing are one joint invariant:

```
active revisions × replicas per revision × max connections per replica
+ platform/reserved connections
<= PostgreSQL max_connections
```

At the incident configuration, `2 × 2 × (12 runtime + 5 admin) = 68`, already above the B1ms ceiling of 50 before Azure/system reserve. The shipped remediation uses runtime max 4 and admin max 5 with an eight-connection reserve: `(4 + 5) × 2 × 2 + 8 = 44/50`, leaving six connections of hard-cap headroom. This is not a normal operating target; existing Sev-1 RULE-04b intentionally pages

earlier at 43. The four modeled replica slots also cover the canary peak: the traffic-serving revision may scale to two replicas while the prior rollback and new 0%-traffic canary remain at one minimum replica each (`2 + 1 + 1 = 4`). `ConnectionBudget` parses bounded environment overrides and fails startup before creating either pool if the joint invariant is unsafe.

The admin pool now has explicit 60-second idle, 10-second acquisition, and 3-second validation timeouts. Its permanently borrowed LISTEN connection is not idle in Hikari and is therefore not reaped by idleTimeout.

Pool telemetry and attribution

Both Hikari pools use the official Micrometer tracker. A one-minute internal reporter reads pool MXBeans without borrowing a connection and exports active, idle, pending, total, maximum, timeout count, and mean/max acquisition latency to structured ACA console logs. Public `/health` does not expose pool capacity. PostgreSQL `application_name` is `qody-<runtime|admin>@<ACA revision>`, sanitized and collision-safe within the 63-byte server limit. This identifies a pool/revision for a connection's lifetime; it does not identify individual requests.

Canonical author-only alert definitions are:

- RULE-04c — Hikari pending connection waiters;
- RULE-04d — Hikari acquisition-timeout counter increase;
- RULE-04e — configured joint-budget headroom below four.

They extend the DB-pool portion of MC #104302 and leave existing PostgreSQL RULE-04/RULE-04b unchanged.

After an approved deployment, verify the platform-provided revision identity and DB attribution (commands documented here, not executed during implementation):

```
az containerapp exec -g rg-qody-prod -n qody-api-prod \
  --revision <serving-revision> \
  --command "printenv CONTAINER_APP_REVISION"

# Through an approved DB session; expected qody-runtime@... / qody-admin@...
SELECT application_name, state, count(*)
FROM pg_stat_activity
WHERE datname = 'qody'
GROUP BY application_name, state;
```

Runbook correction

The former QODY pointer `~/system/runbooks/azure-aca-incident.md` did not exist. The versioned incident playbook is now `docs/security/incident-response.md`, with QODY-specific rollback and automatic-retention behavior in `RUNBOOK.md`.

Rollback

The newest healthy 0%-traffic revision stays active specifically for fast rollback. Operators may route 100% traffic to it using the existing `az containerapp ingress traffic set` procedure in `RUNBOOK.md`. Automatic retention never changes traffic weights and never deactivates failed canaries.

Scope boundaries

Not introduced by this task without separate CEO approval:

- PostgreSQL SKU or `max_connections` change;
- API replica-count increase;
- PgBouncer/pgcat;
- `ALTER ROLE ... CONNECTION LIMIT`;
- standalone/cron cleanup or failed-canary cleanup;
- production deployment.

Production rollout completion — 2026-08-18

- Final source: `71fd83c778c90ae1ed320b095b6387fbc96c7a49`; production build **1126 succeeded** after the configured CEO approval gate.
- `qody-api-prod--71fd83c7-1126` is Healthy/Provisioned at 100% traffic; `qody-api-prod--956f9f35-939` remains Healthy at 0% as rollback. Exactly two API revisions are active.
- Live Hikari telemetry: runtime maximum 4, admin maximum 5, configured app maximum 36, available-to-app 42, headroom 6, pending 0, timeout count 0.
- PostgreSQL: latest maximum 17 connections, 30-minute maximum 20 during rollout, failed connections 0. Standard_B1ms was retained because verified headroom made a SKU mutation unnecessary.
- RULE-04c/d/e are enabled, routed to `qody-ops-oncall`, and their exact production queries returned no firing condition after rollout.
- API, Guest, Admin, and Kitchen public endpoints returned HTTP 200; production browser deploy-gate passed 2/2.
- Request-path analysis disproved a route-driven leak: traffic remained 2–3 requests per five-minute bucket; stale active revisions held the static pool baseline.

Evidence: `/Users/makinja/system/evidence/107292/prod-postdeploy/final-summary.json` and `/Users/makinja/system/evidence/107292/prod-postdeploy/request-path-verdict.json`.

Security Sweep — qody.ba (MC #107297)

Security Sweep — qody.ba (MC #107297)

Date: 2026-08-18
Canonical repository: Azure DevOps QODY, GitHub private mirror
Production merge: `630f4cff5d7f1bfd6cf163e503223cbf23c4d671`
Azure PR: `#350` · **CI build:** `#1120`

Baseline

- Repository already private; reachable history and working candidate had zero Gitleaks findings.
- Live `wrangler.toml` returned its exact control-file content.
- Unknown/sensitive paths returned the landing homepage with HTTP 200 because no 404 page existed.
- Landing responses lacked CSP, HSTS, frame and permissions protection.
- Root asset directory mixed runtime files with Wrangler config, Functions, TypeScript source and generated caches.
- Contact Function threw on malformed JSON, had no OPTIONS/body/rate controls, logged name/venue/email/IP/country, and returned success without checking provider delivery.
- `apps/landing/` was excluded from repository Semgrep scans.

Remediation

- Created physical runtime-only `apps/landing/public/` root; Functions, config, source, tests and package metadata remain outside it.
- Updated Wrangler output and DEPLOY-MAP to deploy only `public/` after npm audit/preflight/type/tests.
- Added true 404, CSP/HSTS/browser headers and defense-in-depth `.assetsignore`.
- Hardened contact Function with streaming 8 KiB bound, strict JSON/field/email/phone validation, origin/method controls, honeypot, rate limit, no PII logs, provider

timeout/status enforcement, generic failures and no-store/security headers.
- Added honeypot consistently to HTML, TypeScript source and generated app.js; preflight asserts runtime/source consistency.
- Added pinned test/type tooling and 7 contact-control tests; npm audit zero.
- Added SHA-pinned GitHub mirror security CI and removed canonical landing from `.semgrepignore`.
- Canonical Azure build policy #1120 passed; AzDO PR #350 merged without bypass and GitHub main mirror fast-forwarded.
- Added exact Cloudflare edge tombstone for stale cached `/wrangler.toml` until cache expiry.

Verification

- Local preflight/type/tests/audit: PASS; 7/7 tests.
- Semgrep landing: 0 findings / 0 errors.
- Gitleaks current/all reachable history: 0.
- Local Wrangler smoke: runtime root 200; config/source/package/unknown paths 404; Function controls pass.
- Independent review: PASS, P0=0, P1=0.
- Cloudflare preview Playwright: 21/21 PASS, zero console errors.
- Production Playwright: 21/21 PASS, zero console errors.
- Production sensitive/control paths: 404 after edge routing.

Boundaries and residuals

1. In-memory rate limiting is best-effort per Worker isolate; add Cloudflare Rate Limiting/Turnstile only under a separate abuse-volume decision.
2. MailChannels remains a third-party delivery dependency; Function now fails honestly on non-2xx delivery.
3. Existing payment webhook replay finding MC #106337 remains separate and was not duplicated or modified.
4. Canonical dirty developer checkout was not reset or cleaned; remediation used an isolated worktree.

Evidence

`/Users/makinja/system/evidence/107297/`

QODY Kitchen KDS live update — MC #107293 / #106857 — 2026-08-18

QODY Kitchen KDS live update — final evidence

Status: Functionally implemented, deployed, and live-verified on demo and production surfaces.
Tasks: MC #107293 (missed KDS updates), MC #106857 (demo Kitchen credential blocker), MC #107308 (deterministic first health request).

Incident and confirmed causes

An already-open Kitchen Display System could remain `Uzivo` while a new order was absent until manual `SINKRONIZIRAJ`. A controlled pre-fix reproduction dropped the new-order WebSocket frames: the order remained absent after 20 seconds and appeared immediately after manual sync. Confirmed causes were:

1. order events could be broadcast before the transaction committed, so a separate listener connection could fetch no payload;
2. PostgreSQL NOTIFY/WebSocket delivery is intentionally non-durable and the KDS had no bounded authoritative replay;
3. the existing-order pay-at-table transition had no KDS event;
4. the demo KITCHEN fixture password in the long-lived database no longer matched the documented fixture, blocking exact-role UAT.

Implemented remediation

- Backend broadcasts order/payment/lifecycle events only after commit.
- `ORDER_PAYMENT_MODE_UPDATED` maps to the KDS `order_update` wire event.
- Visible Kitchen boards reconcile from the authoritative database every 10 seconds, pause while hidden, refresh on visibility return, abort hung requests, coalesce overlaps, and

reject stale responses that race newer WebSocket events.
- Pending-payment and terminal orders remain excluded; no unbounded polling was added.
- A demo-only repeatable Flyway migration repairs the canonical KITCHEN fixture without exposing a reset endpoint or loading in production.
- E2E login now uses the real httpOnly-cookie KITCHEN form path; obsolete bearer/localStorage and OWNER fallbacks were removed. It waits deterministically for the initial `/staff/orders` response and connected WS label.

Source, reviews, and rollout

- Core fix commit `14d38c6b68381a74e73375b712f898d87af8734d`, PR **347**, merged and deployed to production in build **1126**.
- Credential repair commit `b884161362c2f05243c21812ca70dbb56ba22dae`, PR **355**, policy build **1135 PASS**, merged.
- Exact-role harness commit `c47d846457d42c4a2937e9fc7cbbb630e20b798d`, PR **358**, policy build **1143 PASS**, merged as `5aca15661402ccfede109c70ec67063a3e2cd9ee`.
- Demo API canary `qody-api--bc227a76-1140`: health/RLS PASS, 100% traffic; previous serving revision retained as rollback.
- Demo Kitchen `qody-staff-kitchen--0000031`: Healthy/Provisioned at 100%; former serving revision retained at 0% rollback.
- Production Kitchen `qody-staff-kitchen-prod--71fd83c7-1126`: Healthy/Provisioned at 100%.
- Staff-Kitchen unit suite 28/28 PASS; backend tests PASS; real-PostgreSQL credential migration test PASS.
- CodeCraft review P0=0/P1=0; Redžo and Gemini final verdicts APPROVE.

Exact live acceptance proof

The final test uses the canonical **KITCHEN** role and real cookie authentication. No OWNER fallback, token injection, manual sync, or reload is permitted.

Flow	Result	Evidence
K-02 normal WebSocket	**PASS**	Order appeared in **47 ms** with two order-related WS frames
K-02b missed frame	**PASS**	Exactly one `order_created` frame deliberately dropped; one card recovered in **9,919 ms**
Duplicate guard	**PASS**	Exactly one recovered KDS card
Cleanup	**PASS**	Both test orders canceled successfully

Flow	Result	Evidence
Credential causal check	**PASS**	Canonical demo KITCHEN login changed from pre-deploy 401 to post-deploy 200/KITCHEN

Evidence paths

- Final manifest: `/Users/makinja/system/evidence/106857/kitchen-live-update/final-live-verification.json`
- Live test log: `/Users/makinja/system/evidence/106857/kitchen-live-update/k02-k02b-live-r5.stdout.log`
- Screenshots and WS frames: `/Users/makinja/system/evidence/106857/kitchen-live-update/live-proof-deterministic/`
- API canary/retention: `/Users/makinja/system/evidence/106857/kitchen-live-update/manual-api-canary-result.json`
- Root-cause verdict: `/Users/makinja/system/evidence/107293/root-cause-verdict.md`
- Dual review: `/Users/makinja/system/evidence/107293/dual-review/dual-review-summary.json`

Administrative note

Company Mesh verification prompts timed out because no responder payload was available. This was an orchestration-gate failure, not a product/test failure. No peer PASS was fabricated; closure is based on deterministic live Playwright evidence, Azure policy builds, merged PRs, deployed revision identity, cleanup proof, and independent Redžo/Gemini/CodeCraft reviews.