# Bilko Mobile

Native iOS/Android mobile companion docs, Entra External ID auth bridge, Phase 0 status and implementation specs

# Bilko Mobile Phase 0 Auth Bridge — Status 2026-06-05

# Bilko Mobile Phase 0 Auth Bridge — Status 2026-06-05

## Executive summary

Bilko mobile direction is native **iPhone + Samsung/Android**, with **React Native + Expo** as the implementation path. The PWA/mobile-web direction is superseded for the companion mobile app.

Phase 0 backend/auth work is implemented locally and targeted tests are green. The mobile app build should **not** be dispatched until the remaining gates below are closed.

## What was changed

### Documentation and architecture

Updated or created:

- `docs/mobile/MOBILE-ARCHITECTURE.md`
- `docs/mobile/MOBILE-IMPL-SPEC-PHASE1.md`
- `docs/mobile/MOBILE-PRD.md`
- `docs/mobile/README.md`
- `docs/INDEX.md`
- `/Users/makinja/system/specs/bilko-tech-stack.md`
- `docs/architecture/ADR-037-BILKO-MOBILE-NATIVE-ENTRA-AUTH.md`
- `docs/backend/MOBILE-ENTRA-AUTH-BRIDGE-SPEC.md`

Key doc decisions:

- Native iOS/Android app is the target, not PWA.
- React Native + Expo is the chosen native cross-platform path.

- Entra External ID + OIDC Authorization Code + PKCE is the target mobile/customer login model.
- Phase 1 uses existing `/api/v1/*` endpoints; a dedicated `/mobile/*` BFF is deferred.
- Mobile refresh must not depend on browser cookies.
- Email claims from Entra are not trusted for login mapping.

# Backend/auth implementation

Added or changed:

- `apps/api/src/main/kotlin/no/alai/bilko/auth/EntraExternalIdService.kt`
  - Verifies Microsoft Entra External ID JWTs.
  - Fails closed if issuer/audience/JWKS config is missing.
  - Enforces RS256 and `kid`.
  - Verifies configured issuer and audience.
  - Extracts verified subject from `sub`, with `oid` fallback.
- `apps/api/src/main/resources/db/migration/V64__entra_external_identities.sql`
  - Adds `entra_external_identities` mapping table.
  - Maps verified `issuer + subject` to existing Bilko user.
  - Adds SECURITY DEFINER functions with fixed `SET search_path = public, pg_temp`:
    - `bilko_auth.find_user_by_entra_identity(text, text)`
    - `bilko_auth.mark_entra_login(text, text)`
- `apps/api/src/main/kotlin/no/alai/bilko/db/AuthUserRepository.kt`
  - Adds `findByEntraIdentity(issuer, subject)`.
  - Adds `markEntraLogin(issuer, subject)`.
- `apps/api/src/main/kotlin/no/alai/bilko/auth/AuthService.kt`
  - Adds `createSessionFromEntraIdToken(idToken)`.
  - Maps verified Entra identity to active Bilko user/org/role.
  - Issues existing Bilko access + refresh tokens.
- `apps/api/src/main/kotlin/no/alai/bilko/plugins/DI.kt`
  - Wires `EntraExternalIdService` into `AuthService`.
- `apps/api/src/main/kotlin/no/alai/bilko/routes/AuthRoutes.kt`
  - Adds `POST /api/v1/auth/entra/session`.
  - Adds `POST /api/v1/auth/mobile/refresh`.
  - Preserves existing web cookie refresh endpoint `POST /api/v1/auth/refresh`.
  - Hardened bad-body handling by removing `printStackTrace()` and detailed parser error echo from register bad-body response.

# Tests added or extended

- `apps/api/src/test/kotlin/no/alai/bilko/auth/EntraExternalIdServiceTest.kt`
  - valid token
  - wrong audience

    - wrong issuer
    - expired token
    - HS256 rejection
    - missing `sub`/`oid`
    - missing config fail-closed
- `apps/api/src/test/kotlin/no/alai/bilko/db/AuthUserRepositoryTest.kt`
    - Flyway migration execution on disposable PostgreSQL/Testcontainers DB, including V64.
    - mapped Entra identity returns user
    - unmapped identity returns null
    - inactive mapped user returns null
    - soft-deleted mapped user returns null
    - org/role/status assertions
    - `markEntraLogin()` metadata update
- `apps/api/src/test/kotlin/no/alai/bilko/auth/AuthServiceTest.kt`
    - refresh-token rotation/reuse rejection test
- `apps/api/src/test/kotlin/no/alai/bilko/routes/AuthRoutesHttpIntegrationTest.kt`
    - `/auth/entra/session` missing `idToken` returns 400
    - `/auth/entra/session` missing Entra config returns 503 `CONFIGURATION_ERROR`
    - `/auth/mobile/refresh` missing `refreshToken` returns 400
    - `/auth/mobile/refresh` with structurally valid but stale/non-DB refresh JTI returns 401
    - web cookie refresh/logout stale-token regression remains covered

# Validation evidence

Evidence files:

- `/Users/makinja/system/evidence/bilko-mobile-doc-review-20260604.md`
- `/Users/makinja/system/evidence/bilko-mobile-auth-local-security-audit-20260604.md`
- `/Users/makinja/system/evidence/securion-review-check-102962-20260605.md`

Commands recorded as green in evidence:

- `./gradlew compileKotlin --no-daemon` → BUILD SUCCESSFUL
- `./gradlew compileKotlin compileTestKotlin --no-daemon` → BUILD SUCCESSFUL
- `./gradlew test --tests no.alai.bilko.auth.EntraExternalIdServiceTest --no-daemon` → BUILD SUCCESSFUL
- `./gradlew integrationTest --tests no.alai.bilko.db.AuthUserRepositoryTest --no-daemon` → BUILD SUCCESSFUL
- `./gradlew integrationTest --tests no.alai.bilko.auth.AuthServiceTest --no-daemon` → BUILD SUCCESSFUL
- `./gradlew integrationTest --tests no.alai.bilko.routes.AuthRoutesHttpIntegrationTest --no-daemon` → BUILD SUCCESSFUL
- Combined targeted run:

- `./gradlew compileKotlin compileTestKotlin test --tests no.alai.bilko.auth.EntraExternalIdServiceTest integrationTest --tests no.alai.bilko.db.AuthUserRepositoryTest --tests no.alai.bilko.auth.AuthServiceTest --tests no.alai.bilko.routes.AuthRoutesHttpIntegrationTest --no-daemon` → BUILD SUCCESSFUL

# Where we stand

## Done locally

- Mobile architecture direction cleaned up and made native-first.
- Entra External ID bridge implemented.
- Mobile-safe body refresh endpoint implemented.
- V64 migration implemented and executed via Testcontainers/Flyway integration test.
- Targeted local tests passed.
- Local security audit completed with no critical/high local findings; one route error-handling hygiene issue was fixed.

## Still blocked

- No independent Securion/QA domain PASS has been received.
- MC `#102962` was checked and is still open in `/Users/makinja/system/evidence/securion-review-check-102962-20260605.md`.
- Real Entra staging trigger substitutions are provisioned/verified, but the current Azure tenant is `AAD` (`alemalai.onmicrosoft.com`), not a separate CIAM customer tenant. Customer-facing CIAM tenant policy/MFA/passwordless configuration remains a later production gate.
- No live environment/browser/mobile-device verification has been performed for this auth bridge.

## Decision

Do **not** dispatch mobile app build yet.

Next concrete step is to stop waiting for passive task pickup and run/obtain a real independent security/QA review, then provision/test real Entra External ID config. After those pass, dispatch the native React Native + Expo mobile build.

# Stage deploy — substitution wiring (MC #102996, 2026-06-05)

`infrastructure/gcp/cloudbuild-stage.yaml` is now substitution-ready for Entra External ID metadata:

- Three substitutions added to the `substitutions:` block: `_ENTRA_EXTERNAL_ID_ISSUER`, `_ENTRA_EXTERNAL_ID_AUDIENCE`, `_ENTRA_EXTERNAL_ID_JWKS_URL`, all defaulting to `__UNSET__`.
- A conditional block in the `deploy-api-no-traffic` step builds `ENTRA_ENV_VARS`. If all three substitutions are non-`__UNSET__` and non-empty, they are appended to `--set-env-vars` as `;ENTRA_EXTERNAL_ID_ISSUER=...;ENTRA_EXTERNAL_ID_AUDIENCE=...;ENTRA_EXTERNAL_ID_JWKS_URL=...`. Otherwise `ENTRA_ENV_VARS=""` — the env-vars string is unchanged and current stage behaviour is preserved exactly.
- Values are provisioned in the `bilko-stage-auto-deploy` trigger as of 2026-06-05:
  - `_ENTRA_EXTERNAL_ID_ISSUER=https://login.microsoftonline.com/3454a03f-20b4-4bda-a116-2293c459aecd/v2.0`
  - `_ENTRA_EXTERNAL_ID_AUDIENCE=95b2a55f-8f48-4c9c-b4f8-eb455e3bdfd7`
  - `_ENTRA_EXTERNAL_ID_JWKS_URL=https://login.microsoftonline.com/3454a03f-20b4-4bda-a116-2293c459aecd/discovery/v2.0/keys`
- No stage smoke has been run against Entra yet. Stage will only receive these env vars after the committed Cloud Build wiring is pushed/merged and the trigger deploys a new API revision.

# Immediate next actions

1. Force active review path for MC `#102962` or run a direct independent Securion/QA validation with evidence.
2. Push/merge the clean Phase 0 branch through the gated path.
3. Trigger a stage build and confirm `MC#102996: Entra metadata present` log line appears and the three vars are visible in the Cloud Run revision env.
4. Run `/api/v1/auth/entra/session` against a real Entra test tenant token.
5. Run `/api/v1/auth/mobile/refresh` with real issued mobile refresh token.
6. Close Securion MC `#102989` with passing evidence.
7. Only after PASS evidence, dispatch native mobile implementation.

# Bilko Mobile — README

# Bilko Mobile Companion

Product and architecture docs for the Bilko native mobile companion app.

Decision update 2026-06-04: Bilko mobile targets **iPhone + Samsung/Android native app delivery via React Native + Expo**. Older PWA-for-MVP assumptions are superseded for this workstream.

# Documents

- `MOBILE-PRD.md` — product requirements, phased MVP scope, country-specific behavior, release plan.
- `MOBILE-ARCHITECTURE.md` — approved React Native/Expo architecture direction, API/security/storage approach, and later-phase offline/notification targets.
- `MOBILE-IMPL-SPEC-PHASE1.md` — Phase 1 implementation spec; not dispatch-ready until Phase 0 Entra/auth/backend blockers are closed.

# Scope

The mobile app is positioned as a daily owner/operator companion for HR, RS, and BA SMBs. Phase 1 focuses on Entra/Bilko login, dashboard/list visibility, receipt capture/upload, and narrow travel-order quick-add. Approvals, invoice quick actions, compliance alerts, push notifications, OCR, and durable offline queue are later phases unless explicitly re-scoped.

The web app remains the primary workspace for full accounting, double-entry journals, complex reporting, bank reconciliation, and administration.

# Bilko Mobile PRD

# Bilko Mobile Companion PRD

> **Product:** Bilko Mobile Companion
> **Markets:** Croatia (HR), Serbia (RS), Bosnia & Herzegovina (BA)
> **Status:** Direction approved — native iPhone + Samsung/Android companion; product scope remains draft
> **Date:** 2026-05-25
> **Updated:** 2026-06-04
> **Owner:** ALAI / Bilko Product

## 1. Source Context

This PRD is derived from the current Bilko product and regulatory documentation:

- `docs/business-requirements/BRD.md`
- `docs/business-requirements/FUNCTIONAL-REQUIREMENTS.md`
- `docs/business-requirements/USER-STORIES.md`
- `docs/regulatory/MULTI-REGION-OVERVIEW.md`
- `docs/regulatory/HR/README.md`
- `docs/regulatory/RS/README.md`
- `docs/regulatory/BA/README.md`
- `docs/frontend/PAGES.md`
- `docs/PRICING-STRATEGY.md`

## 2. Product Positioning

Bilko Mobile should be a **daily owner/operator companion**, not a full replacement for the Bilko web accounting workspace.

The mobile app should help Balkan SMB owners and accountants do the urgent work that naturally happens away from a desk:

- capture receipts, supplier invoices, and other documents;
- check cash position and overdue invoices;

- approve expenses and invoice actions;
- create or send simple invoices;
- receive country-specific compliance and e-invoice status alerts;
- collaborate with an accountant on missing documents.

The web app remains the primary workspace for full bookkeeping, double-entry journals, complex reporting, configuration, reconciliation, and administration.

# 3. Target Users

## 3.1 SMB Owner / Director

Needs quick answers and fast actions:

- “How much cash and unpaid revenue do I have?”
- “Who is overdue?”
- “Can I send this invoice now?”
- “Did SEF/eRačun accept my invoice?”
- “What documents does my accountant need?”

## 3.2 Field Operator / Freelancer

Needs mobile capture and simple invoicing:

- scan receipt immediately;
- create a simple invoice after job completion;
- share invoice PDF via email/WhatsApp;
- work when connection is weak and sync later.

## 3.3 Accountant / Bookkeeper

Needs lightweight collaboration:

- request missing documents;
- comment on expense/invoice records;
- approve or reject submitted documents;
- reduce back-and-forth over email and messaging apps.

# 4. Goals

1. Increase daily engagement with Bilko through mobile-first workflows.
2. Reduce missing receipts and late expense capture.
3. Give owners instant cash, unpaid invoice, and VAT/PDV visibility.
4. Support time-sensitive regulatory workflows through alerts and status views.
5. Keep accounting integrity by making the backend the source of truth.
6. Make the app useful in HR, RS, and BA through country plugins and localized terminology.

# 5. Non-Goals

The first internal mobile build / Phase 1 must not attempt to implement:

- full double-entry journal editing;
- full bank reconciliation workflow;
- complex financial report builder;
- payroll;
- manual regulatory XML editing;
- full admin/settings console;
- durable offline queue;
- OCR extraction;
- push notifications;
- `/mobile/*` BFF rebuild unless explicitly scoped;
- BiH e-invoice submission before official specs/platform requirements are stable.

# 6. Country Requirements

## 6.1 Croatia (HR)

Mobile must support:

- EUR currency display;
- Croatian VAT/PDV labels and rates surfaced from backend/domain logic;
- eRačun/HR-FISK readiness/status views as backend support matures;
- certificate/connection readiness warnings where applicable;
- submission success/failure notifications.

Priority: prepare for 2026 eRačun/HR-FISK operational use without moving regulated generation/signing logic into the mobile client.

## 6.2 Serbia (RS)

Mobile must support:

- RSD currency display;
- Serbian PDV terminology;
- SEF status visibility for B2B invoices;
- notifications for accepted/rejected/failed/overdue e-invoice states;
- handoff to web for complex correction workflows.

Priority: SEF visibility and owner alerts, because Serbian B2B e-invoicing is already mandatory.

## 6.3 Bosnia & Herzegovina (BA)

Mobile must support:

- BAM currency display;
- 17% VAT/PDV basics surfaced from backend/domain logic;
- entity/company profile display where relevant;
- document capture, cash dashboard, VAT reminders, and accountant collaboration.

Priority: do not implement speculative e-invoice submission until official BA specs and platform requirements are stable.

# 7. Product MVP Scope

This section describes the broader product MVP ambition. It is **not** the first engineering slice. The first internal iPhone/Android build is narrowed in section 12 to login, dashboard/list visibility, camera capture/upload, and basic HR travel-order quick-add after backend blockers are closed.

## 7.1 Today Dashboard

Required:

- cash/bank balance summary when available;
- unpaid invoices total;
- overdue invoices count and amount;
- expenses pending approval;
- upcoming VAT/PDV obligation/deadline card;
- country-specific currency and labels.

Should:

- deep link to invoice/expense lists;
- show “needs attention” items first;
- work from cached data when offline in Phase 2+.

# 7.2 Document Inbox and Capture

Required:

- capture receipt or bill using camera;
- import PDF/image from mobile share sheet;
- create expense or supplier invoice draft;
- attach document image/PDF;
- show upload status and retry errors in Phase 1;
- queue upload if offline in Phase 2+.

Should:

- OCR-assisted extraction for supplier, amount, date, VAT/PDV, and category in Phase 3+ after legal/cost review;
- manual correction before submit;
- dead-letter state when sync repeatedly fails.

# 7.3 Invoice Quick Actions

Required:

- list invoices by status;
- view invoice details;
- share/download PDF when backend PDF exists;
- send payment reminder;
- mark invoice paid, subject to permissions and audit trail.

Should:

- create simple invoice from existing customer and item/service list;
- save draft offline and sync later;
- start regulated e-invoice submission only through backend-supported workflows.

# 7.4 Approvals

Required:

- approve/reject expense draft;
- approve/reject invoice before sending/submission;
- comment on rejection;
- push notification for approval requests.

Should:

- show audit history for approvals;
- support role-based permissions aligned with the web app.

## 7.5 Compliance and Status Alerts

Required:

- VAT/PDV deadline reminders;
- overdue invoice reminders;
- e-invoice status notifications for supported markets;
- failed submission alerts;
- deep links from notification to relevant record.

Should:

- certificate/token expiry warning for HR/regulated integrations where applicable;
- user-configurable notification preferences.

## 7.6 Accountant Collaboration

Required:

- comments on invoice/expense/document;
- missing-document task list;
- mark requested document as provided;
- push notification when accountant requests action.

Should:

- simple in-app thread per record;
- unread indicator.

# 8. Functional Requirements

| ID | Requirement | Priority |
|---|---|---|
| MOB-FR-001 | User can log in with Microsoft Entra External ID / Bilko account. | P0 |
| MOB-FR-002 | User can select/switch company if account has multiple companies. | P2 |
| MOB-FR-003 | App displays country-specific currency and tax labels. | P0 |

| ID | Requirement | Priority |
|---|---|---|
| MOB-FR-004 | User can view dashboard summary. | P0 |
| MOB-FR-005 | User can view invoice list and detail. | P0 |
| MOB-FR-006 | User can capture receipt/bill with camera. | P0 |
| MOB-FR-007 | User can submit expense draft with attachment. | P0 |
| MOB-FR-008 | App queues supported draft actions offline. | P2 |
| MOB-FR-009 | User receives approval and compliance push notifications. | P3 |
| MOB-FR-010 | User can approve/reject expense or invoice action. | P1 |
| MOB-FR-011 | User can create a simple invoice draft. | P1 |
| MOB-FR-012 | User can share invoice PDF generated by backend. | P1 |
| MOB-FR-013 | User can see SEF/eRačun/HR-FISK status where backend supports it. | P1 |
| MOB-FR-014 | User can comment on documents/invoices/expenses. | P2 |
| MOB-FR-015 | User can manage notification preferences. | P2 |

# 9. UX Structure

Recommended bottom tabs:

1. **Today** — dashboard and attention queue.
2. **Inbox** — captured documents and missing-document tasks.
3. **Invoices** — invoices, reminders, simple create/share.
4. **Expenses** — expense drafts and approvals.
5. **More** — company switcher, settings, security, help.

Key design rule: mobile screens should optimize for one-handed action and short sessions, not dense accounting tables.

# 10. Data and Backend Dependencies

The Phase 1 mobile app needs backend/API support for:

- Microsoft Entra External ID bridge: token validation, Bilko user/org mapping, role mapping, and mobile-safe session/API token issuance;
- existing `/api/v1` dashboard/report data or a minimal dashboard projection;
- invoice list/detail read-only endpoints;
- expense draft create/update/submit;
- attachment upload with clear size limit and retry behavior;
- country/org metadata via `GET /auth/me` or equivalent.

Later phases need backend/API support for:

- `/mobile/*` BFF endpoints where projection efficiency justifies them;
- approval actions with audit log;
- notification token registration;
- e-invoice status endpoint per supported provider/market;
- sync endpoints or mutation queue-compatible APIs.

Backend remains authoritative for:

- tax/VAT/PDV calculation;
- double-entry posting;
- invoice numbering;
- regulated XML generation/submission;
- audit trail;
- permissions.

# 11. Success Metrics

Phase 1 success should be measured by:

- successful Entra/Bilko login rate;
- internal testers who can open Today, Invoices, Expenses, and More on iPhone + Android;
- receipt capture/upload success rate;
- mobile crash-free sessions;
- API/auth failures per session.

Later product MVP success should additionally measure:

- weekly active mobile users / active Bilko companies;
- receipt capture count per company per month;
- reduction in missing-document tasks older than 7 days;
- approval turnaround time;
- invoice status alert open rate;
- simple invoice creation or PDF share usage;
- sync failure/dead-letter rate.

# 12. Release Plan

## Phase 0 — Backend/Auth Prerequisites

- Microsoft Entra External ID app registration and OIDC/PKCE configuration;
- Bilko backend auth bridge for Entra token validation;
- Bilko user/org/role mapping;
- mobile-safe session/API token issuance without browser httpOnly cookie dependency;
- align document upload size limit in backend and docs.

## Phase 1 — Internal iPhone/Android Companion Build

- Entra/Bilko login/session;
- Today dashboard;
- invoice and expense read-only lists;
- camera capture;
- expense draft submission with attachment upload;
- basic HR travel-order quick-add if backend payload is confirmed;
- no durable offline queue, no OCR, no push notifications.

## Phase 2 — Cache, Company Switch, and Import

- company switcher;
- cached dashboard/list state;
- file/share import;
- sync status UI;
- optional durable offline upload queue after security review.

## Phase 3 — Notifications and Approvals

- push token registration;
- approval request notifications;
- expense/invoice approve/reject;
- VAT/PDV reminders.

## Phase 4 — Simple Invoicing

- simple invoice draft;
- backend PDF generation/share;

- payment reminder;
- mark paid where permitted.

## Phase 5 — Country Regulatory Status

- RS SEF status first;
- HR eRačun/HR-FISK status/readiness next;
- BA status only after stable official specs/platforms exist.

# 13. Acceptance Criteria for MVP

Phase 1 internal build can be accepted when:

- authenticated user can log in via Entra/Bilko and use Today, Invoices, Expenses, More;
- receipt capture/upload works on iOS and Android;
- country/currency labels are correct for HR/RS/BA companies;
- sensitive tokens/session material are stored only in secure storage;
- user can recover from failed upload without losing the current capture state;
- no code path stores tokens in AsyncStorage or Zustand.

Product MVP can be considered ready later when offline queue, approvals/audit, and push deep links are implemented and verified.

# 14. Open Questions

1. Which market should be the commercial pilot: RS because SEF is mandatory, or HR because 2026 eRačun/HR-FISK creates urgency?
2. Should Phase 2 add `/mobile/*` BFF endpoints, or keep adapting existing `/api/v1/*` endpoints?
3. Which OCR provider should be used, and what data residency/privacy constraints apply?
4. Which push provider should be selected: Expo Notifications, FCM/APNs directly, or a unified service?
5. Should simple invoice creation be in MVP or Phase 4 only?

# Bilko Mobile Architecture

# Bilko Mobile Companion Architecture

> **Product:** Bilko Mobile Companion
> **Markets:** HR / RS / BA
> **Status:** Direction approved — iPhone + Samsung/Android native companion; implementation details remain draft
> **Date:** 2026-05-25
> **Updated:** 2026-06-04
> **Related PRD:** `docs/mobile/MOBILE-PRD.md`

## 1. Architecture Decision

Approved framework: **React Native with Expo** for iPhone and Samsung/Android delivery.

Rationale:

- Bilko already has a TypeScript/React-oriented web product, so React Native maximizes language, tooling, and design-system reuse.
- Expo provides faster iOS/Android delivery, camera/file-system/push/secure-storage primitives, OTA update options, and a simpler CI path.
- The app does not require heavy native computation in MVP; regulated tax/e-invoice logic remains server-side.
- Native Swift/Kotlin would increase delivery cost and split the codebase too early.

Decision status: **final for Bilko mobile direction**. The older PWA-for-MVP note in `/Users/makinja/system/specs/bilko-tech-stack.md` is superseded for mobile app work; responsive/PWA web may still exist as web capability, but it is not the mobile companion app delivery path.

## 2. Core Principles

1. **Backend is source of truth** for accounting, tax, invoice numbering, e-invoice submission, audit logs, and permissions.
2. **Mobile is an action companion**, optimized for capture, approvals, alerts, and simple workflows.
3. **Offline-first only where valuable**: drafts, queued uploads, cached dashboard, and pending approvals.
4. **Country behavior is configuration/plugin-driven**, not hardcoded screen forks.
5. **No regulated secrets or certificates in the mobile app** unless a future legal/security review explicitly approves it.
6. **Security and auditability are product features**, not afterthoughts.

# 3. Proposed App Structure

```
apps/mobile/
├── app/                        # Expo Router routes
│   ├── (auth)/
│   ├── (tabs)/
│   │   ├── today/
│   │   ├── inbox/
│   │   ├── invoices/
│   │   ├── expenses/
│   │   └── more/
│   ├── invoice/[id].tsx
│   ├── expense/[id].tsx
│   └── document/[id].tsx
├── src/
│   ├── api/                    # typed API client + endpoint modules
│   ├── auth/                   # session, refresh, company switcher
│   ├── components/             # UI primitives and feature components
│   ├── country/                # country metadata and formatting adapters
│   ├── db/                     # local SQLite schema and repositories
│   ├── features/               # dashboard, inbox, invoices, expenses
│   ├── notifications/          # push registration, handlers, deep links
│   ├── security/               # secure storage, app lock, logger scrubber
│   ├── sync/                   # queue, retry, conflict handling
│   ├── types/                  # shared DTOs and local models
│   └── utils/
├── assets/
├── app.config.ts
```

```
└── package.json
```

# 4. Navigation

Recommended navigation: **Expo Router** with authenticated tab shell.

```
Root
├── Auth Stack
│   ├── Login
│   ├── Forgot Password
│   └── MFA / Device Verification (future)
└── App Tabs
    ├── Today
    ├── Inbox
    ├── Invoices
    ├── Expenses
    └── More
```

Deep links:

- `bilko://invoice/{id}`
- `bilko://expense/{id}`
- `bilko://document/{id}`
- `bilko://approval/{id}`
- `bilko://compliance/{country}/{type}`

Universal links can be added later when production mobile domains are defined.

# 5. Country Plugin Layer

The mobile app should request country/company metadata from the backend after login.

Example mobile country config:

```
type BilkoCountry = 'HR' | 'RS' | 'BA'

type CountryMobileConfig = {
  country: BilkoCountry
  currency: 'EUR' | 'RSD' | 'BAM'
  vatLabel: 'PDV' | 'VAT'
```

```
  invoiceStatusProvider?: 'SEF' | 'HR_FISK_ERACUN' | null
  supportsRegulatedEInvoiceSubmission: boolean
  supportsMobileSimpleInvoice: boolean
  complianceReminderTypes: string[]
}
```

Initial defaults:

| Country | Currency | Regulated status in mobile | Submission in mobile |
|---|---|---|---|
| HR | EUR | eRačun/HR-FISK readiness/status when backend supports it | Backend-only, mobile initiates only approved backend workflow |
| RS | RSD | SEF status and failure alerts | Backend-only, mobile initiates only approved backend workflow |
| BA | BAM | VAT/PDV reminders and document/accountant workflow | No e-invoice submission until official specs are stable |

# 6. API Boundary

Mobile should eventually consume a stable API/BFF layer rather than directly reproducing web internals.

**Phase 1 decision:** use existing `/api/v1/*` endpoints where they are already live, plus a small Phase 0 auth bridge for Microsoft Entra External ID. Do **not** build the full `/mobile/*` BFF before the first iPhone/Android internal build.

**Phase 2+ target endpoints:**

```
GET    /mobile/bootstrap
GET    /mobile/dashboard
GET    /mobile/country-config
GET    /mobile/invoices
GET    /mobile/invoices/{id}
POST   /mobile/invoices/{id}/reminders
POST   /mobile/invoices/{id}/mark-paid
POST   /mobile/invoice-drafts
GET    /mobile/expenses
GET    /mobile/expenses/{id}
POST   /mobile/expense-drafts
```

```
POST   /mobile/approvals/{id}/approve
POST   /mobile/approvals/{id}/reject
POST   /mobile/documents/upload-url
POST   /mobile/documents/complete-upload
GET    /mobile/sync/pull?since={cursor}
POST   /mobile/sync/push
POST   /mobile/push-token
DELETE /mobile/push-token/{id}
```

All mutation endpoints must:

- enforce server-side permissions;
- be idempotent where retry is expected;
- emit audit log entries;
- return server-generated status/version/cursor.

# 7. Local Storage

Phase 1 storage:

- **Expo SecureStore** / native Keychain-Keystore for Entra/Bilko session tokens.
- **In-memory Zustand state** for dashboard/list data.
- **No SQLite offline queue in Phase 1**; failed uploads show retry UI and must not claim offline support.

Phase 2+ target storage:

- **SQLite** for structured local cache and sync queue.
- **Expo FileSystem** for queued attachments.
- **Expo SecureStore** / native Keychain-Keystore for local encryption key.

Local data categories:

| Data | Storage | Offline? | Notes |
|---|---|---|---|
| Access token / ID token | SecureStore | Yes | short-lived; issued via Entra/Bilko auth bridge |
| Refresh token/session token | SecureStore | Yes | rotation required; never AsyncStorage |
| Company/session metadata | SQLite | Yes | non-secret but sensitive |
| Dashboard cache | SQLite | Read-only | stale indicator required |
| Invoice/expense list cache | SQLite | Read-only | limited retention |

| Data | Storage | Offline? | Notes |
|---|---|---|---|
| Draft expense/invoice | SQLite | Yes | sync queue item |
| Captured document file | FileSystem | Yes | encrypted/retention-managed |
| Push token | Server + memory/cache | No | revocable |

# 8. Offline and Sync Strategy

Phase 1 offline scope:

- no durable offline queue;
- no SQLite cache;
- show clear network error and preserve the current in-memory capture state where possible.

Phase 2+ offline target scope:

- view last dashboard/list cache;
- create receipt/expense draft;
- attach photo/PDF;
- create simple invoice draft if product approves Phase 4 scope;
- queue approval action only if the record version is known.

Sync queue item:

```
type SyncQueueItem = {
  id: string
  entityType: 'expenseDraft' | 'invoiceDraft' | 'document' | 'approval'
  entityId: string
  operation: 'create' | 'update' | 'upload' | 'approve' | 'reject'
  payload: unknown
  fileUri?: string
  baseVersion?: string
  retryCount: number
  createdAt: string
}
```

Retry rules:

1. Drain queue on app foreground and network restore.
2. Use FIFO order per entity.

3. Use exponential backoff.
4. Preserve item after failure; never silently discard user-captured document.
5. Move to user-visible “needs attention” state after max retries.

Conflict rules:

- Dashboard/list cache: server wins.
- Drafts not submitted: client can edit locally.
- Submitted records: server version wins; user sees conflict/failure if stale.
- Approvals: require current server version or explicit server conflict response.

# 9. Push Notification Design

Provider options:

1. Expo Notifications for fastest Phase 3 implementation.
2. Direct FCM/APNs if advanced routing/compliance requirements require it later.
3. Third-party service only if product/ops wants non-engineering campaign controls.

Phase 1: **no push permission prompt and no push-token registration** unless a backend device-token endpoint is explicitly added and approved. Avoid asking users for notification permission before the product can deliver useful notifications.

Transactional notification types:

| Type | Trigger | Deep link |
|---|---|---|
| `approval.requested` | Expense/invoice needs user approval | approval detail |
| `document.requested` | Accountant requests missing document | inbox task |
| `invoice.overdue` | Invoice crosses overdue threshold | invoice detail |
| `tax.deadline` | VAT/PDV deadline approaching | compliance card |
| `einvoice.accepted` | SEF/eRačun provider accepted invoice | invoice detail |
| `einvoice.rejected` | Provider rejected/failed invoice | invoice detail |
| `certificate.expiring` | HR/country integration certificate warning | settings/compliance |

Rules:

- No sensitive invoice/customer amounts in lock-screen notification body by default.
- Deep link payload contains IDs only, not PII.
- Respect notification preferences and legal consent requirements.

# 10. Security Architecture

Required baseline:

- Microsoft Entra External ID for customer identity where supported by ALAI auth standard;
- OIDC Authorization Code + PKCE for mobile login;
- tokens only in Keychain/Keystore-backed secure storage;
- refresh/session token rotation;
- biometric/PIN app unlock after inactivity;
- TLS-only API traffic;
- no hardcoded secrets/API keys that grant backend access;
- local database encryption where feasible;
- PII and token scrubbing from logs/crash reports;
- remote logout/session revocation;
- RBAC enforced server-side for every mutation.

Sensitive screens:

- company financial summary;
- invoice detail;
- expense detail with attachment;
- settings/security;
- regulatory/certificate status.

Recommended controls:

- hide sensitive content in app switcher where supported;
- Android screen-capture prevention for sensitive screens if UX accepts it;
- jailbreak/root detection as warning/risk signal, not a sole control for MVP.

# 11. Analytics and Observability

Track product events without PII:

- `mobile_login_success`
- `mobile_dashboard_viewed`
- `mobile_receipt_captured`
- `mobile_draft_synced`
- `mobile_sync_failed`
- `mobile_approval_approved`
- `mobile_invoice_shared`
- `mobile_notification_opened`

Operational telemetry:

- crash-free sessions;
- API error rate;
- upload retry count;
- sync queue depth;
- dead-letter count;
- push delivery/open rate if provider supports it.

# 12. Build Variants

| Variant | Bundle ID example | API | Analytics | Push |
|---|---|---|---|---|
| Dev | `no.alai.bilko.dev` | local/dev | off | sandbox |
| Staging | `no.alai.bilko.staging` | staging | limited | sandbox |
| Production | `no.alai.bilko` | production | consent-based | production |

Secrets and environment values must be injected via CI or secure config, not committed.

# 13. Testing Strategy

Required test layers:

- unit tests for country formatting and sync queue logic;
- API contract tests for mobile BFF endpoints;
- component tests for key forms and offline states;
- device/simulator smoke tests for iOS and Android;
- E2E tests for login, capture draft, sync, approval, notification deep link;
- security review before production release.

Critical acceptance tests:

1. Receipt captured offline survives app restart and syncs after network restore.
2. HR/RS/BA company displays correct currency and country labels.
3. Push notification opens correct invoice/expense/document.
4. Stale approval action receives conflict and does not silently approve wrong version.
5. Token is not stored in AsyncStorage/MMKV/plain SQLite.

# 14. Implementation Phases

## Phase 0 — API and Design Foundations

- define mobile BFF contracts;
- finalize React Native/Expo decision;
- define country config payload;
- define design tokens reuse from Bilko web.

# Phase 1 — Read-Only App

- login/session;
- company switcher;
- Today dashboard;
- invoice/expense lists;
- cached read-only state.

# Phase 2 — Capture and Upload Queue

- camera capture;
- file import/share extension path;
- expense draft;
- attachment upload queue;
- sync failure recovery.

# Phase 3 — Notifications and Approvals

- push registration;
- deep links;
- approval actions;
- VAT/PDV reminders.

# Phase 4 — Simple Invoice Actions

- invoice draft create;
- backend PDF share;
- mark paid/payment reminder;
- audit trail verification.

# Phase 5 — Regulatory Status Views

- RS SEF status;
- HR eRačun/HR-FISK readiness/status;
- BA only after stable official requirements.

# 15. Open Technical Questions

1. Does Bilko currently expose mobile-safe BFF endpoints, or should `apps/api` add `/mobile/*` ?
2. Which auth/session mechanism is canonical for mobile refresh token rotation?
3. Which local DB encryption approach is acceptable under Expo constraints?
4. Which OCR provider meets cost, language, and privacy requirements for HR/RS/BA receipts?
5. Should push be Expo-managed for MVP or direct APNs/FCM from day one?
6. What is the maximum local retention period for cached financial data and attachments?

# Bilko Mobile Implementation Spec — Phase 1

# Bilko Mobile Companion — Phase 1 Implementation Spec

> **Document type:** Implementation Specification (Phase 1) **MC task:** #102483 **Author:** Paul Hudson / Skybound **Date:** 2026-05-28 **Status:** Direction approved; not dispatch-ready until Phase 0 auth/backend blockers are closed **Updated:** 2026-06-04 **Depends on:** `docs/mobile/MOBILE-ARCHITECTURE.md`, `docs/mobile/MOBILE-PRD.md`

---

# 1. Technology Decision

## 1.1 Framework — React Native with Expo (Managed Workflow)

**Decision: React Native + Expo SDK (managed workflow first, bare only if forced)**

Rationale grounded in the existing Bilko codebase:

- The web app is Next.js 15 / TypeScript / React 19. React Native shares language, linting config (`@alai/eslint-config`, `@alai/tsconfig`), and design system primitives (Lucide icons, Tailwind-aligned token nomenclature). A Flutter build would require duplicating all of this in Dart with no reuse.
- The Zustand store shapes in `apps/web/lib/stores/` can be mirrored/adapted in React Native. Do **not** copy them verbatim: the web stores depend on a browser/cookie-oriented API client, while mobile uses native secure storage and OIDC/PKCE auth. Business logic in `packages/domain-hr/`, `packages/domain-rs/`, `packages/domain-ba/` is plain TypeScript and portable where imports remain React Native-safe.

- Native Swift + Kotlin is two separate codebases with two release tracks. It would double the build, test, and maintenance load for an early-phase product with no native computation requirements.
- Expo Managed Workflow provides camera (`expo-camera`), secure storage (`expo-secure-store`), file system (`expo-file-system`), and push notifications (`expo-notifications`) without a native build machine dependency in early development. The team gets TestFlight/Play Console builds via EAS Build with no Mac required for CI runners.
- Eject to bare workflow only if a capability unavailable in the managed SDK is required (e.g., a specific background processing extension or a native SE/eSIM integration). This is unlikely in Phase 1 or Phase 2 scope.

**Not chosen:** Flutter. Dart is a cold start for the team. No design system reuse. Separate CI pipeline. No business justification at current headcount. **Not chosen:** Native Swift/Kotlin. Two codebases. No shared types or API client. Onerous CI/code-signing setup for MVP.

# 1.2 Navigation — Expo Router v4 (file-based, built on React Navigation v7)

Expo Router is the canonical navigation layer in Expo SDK 52+. It is file-system-based (mirrors Next.js App Router conventions the team already knows), supports typed routes, and integrates deep links and universal links via its built-in linking config. It wraps React Navigation v7 internally.

Directory layout under `apps/mobile/app/` maps directly to routes — `(auth)/login.tsx`, `(tabs)/today/index.tsx`, etc. No separate navigator configuration file is needed for the basic shell; the file tree is the configuration.

Do NOT use React Navigation v7 standalone (without Expo Router). The file-based approach is preferred.

# 1.3 State Management — Zustand (mirror web stores)

The web app already has Zustand 4.5.0 installed and uses typed stores in `apps/web/lib/stores/`. Mobile should mirror the same store shape for dashboard, invoices, expenses, and auth.

Pattern: one store file per domain slice — `src/store/authStore.ts`, `src/store/dashboardStore.ts`, `src/store/invoiceStore.ts`, `src/store/expenseStore.ts`, `src/store/settingsStore.ts`.

Stores hold: remote data state, loading/error flags, optimistic update helpers, and (Phase 2) sync queue references. They do NOT hold tokens — tokens live in Keychain only (see section 2.3).

Alternatives considered and rejected:

- Redux Toolkit: more ceremony, overkill for the screen count in Phase 1.
- TanStack Query: good fit for server-state, but adds another dependency and does not solve the auth/session store or the future offline queue. Adding it in Phase 2 for caching is fine, but Zustand is sufficient for Phase 1.
- Jotai: atomic model is less predictable for the dashboard aggregation pattern.

# 1.4 API Client — Typed fetch wrapper (adapt `apps/web/lib/api.ts`)

The web app's `api.ts` is a useful reference for endpoint shape, error handling, and multipart upload, but mobile must not reuse browser-only cookie/session assumptions. The web client is a typed fetch wrapper around `getApiBaseUrl()` with:

- In-memory access token (`_accessToken` module variable)
- Automatic 401 → refresh → retry cycle
- Typed error objects with `status`, `code`, `details`
- `multipart/form-data` upload support (already used for `expenses.uploadDocument` and `invoices.uploadReceipt`)

Mobile should reproduce the **endpoint/error/upload pattern** in `src/api/client.ts` with these differences:

1. Token/session material is loaded from `expo-secure-store` on app launch and stored back on refresh/session update.
2. `credentials: 'include'` cookie refresh is not used in React Native.
3. Customer login uses Microsoft Entra External ID via OIDC Authorization Code + PKCE (`expo-auth-session`) and then exchanges/validates the Entra token with Bilko backend for Bilko org/role context.

API base URL resolution: always `https://api.bilko.cloud/api/v1` (production) or an env-var override. The `window.location` hostname trick used in the web `api-base.ts` does not apply in React Native. Use `EXPO_PUBLIC_API_URL` instead.

Library: native `fetch` (available in React Native). No axios. No `ky`. These are unnecessary given the thin abstraction already proven in the web codebase.

# 1.5 Token Storage — expo-secure-store

**Entra/Bilko access, ID, refresh/session tokens only in `expo-secure-store`.**

Reasoning: `expo-secure-store` maps to iOS Keychain Services and Android Keystore under the hood. It satisfies the hard security requirement in `MOBILE-ARCHITECTURE.md` section 10: "tokens only in Keychain/Keystore-backed secure storage." `AsyncStorage` is plaintext on disk and is forbidden for token storage.

Key names:

- `bilko_access_token`
- `bilko_id_token`
- `bilko_refresh_or_session_token`
- `bilko_token_expires_at` (ISO string — used to proactively refresh before expiry)

On cold start the auth store reads from SecureStore; on logout it wipes all auth keys.

`react-native-keychain` is an alternative but it requires native code changes and does not work in the Expo managed workflow without a custom dev client build. `expo-secure-store` is the managed-workflow-safe choice.

# 1.6 Camera and OCR Library — expo-camera + expo-image-manipulator

`expo-camera` is the managed workflow camera primitive. It handles permission requests, preview, and photo capture on both iOS and Android. Phase 1 uses it only for still photo capture.

`expo-image-manipulator` (also managed) is used to resize images to max 1920px on the longest dimension and compress to JPEG ≤ 0.85 quality before upload. This is the only Phase 1 image processing step; no on-device OCR.

`expo-document-picker` handles the PDF/file import path from the share sheet (Phase 2, but the library is added to the Phase 1 scaffold so it does not trigger a native rebuild later).

# 1.7 Localization — expo-localization + i18n-js (mirror web message files)

The web app has five locale files: `bs.json`, `en.json`, `hr.json`, `sr-Cyrl.json`, `sr-Latn.json`. These files use `next-intl` conventions with namespaced keys.

Mobile should import the same message JSON files via a shared package (`packages/i18n/`) or by symlinking `apps/web/messages/` for Phase 1. The locale is detected via `expo-localization` (device locale) and can be overridden in Settings.

Library: `i18n-js` (maintained, small, works in RN) or `react-i18next` (heavier but matches the pattern if the team wants parity with web). Recommendation: `i18n-js` for Phase 1 to keep the bundle lean.

Market detection for feature gating (HR-only travel orders): driven by `organization.country` from `GET /auth/me`, not by device locale.

---

# 2. Project Structure

## 2.1 apps/mobile/ Directory Layout

```
apps/mobile/
├── app/                              # Expo Router routes (file = route)
│   ├── (auth)/
│   │   ├── _layout.tsx               # Auth stack layout (no tab bar)
│   │   └── login.tsx                 # Login screen
│   ├── (tabs)/
│   │   ├── _layout.tsx               # Tab bar layout (authenticated shell)
│   │   ├── today/
│   │   │   └── index.tsx             # Dashboard / Today screen
│   │   ├── invoices/
│   │   │   ├── index.tsx             # Invoice list
│   │   │   └── [id].tsx              # Invoice detail
│   │   ├── expenses/
│   │   │   ├── index.tsx             # Expense list / recent activity
│   │   │   └── scan.tsx              # Camera capture screen (modal)
│   │   └── more/
│   │       └── index.tsx             # Profile / Settings / Logout
│   ├── travel-order/
│   │   └── new.tsx                   # Travel order wizard (HR only — gated)
│   └── _layout.tsx                   # Root layout (font loading, splash guard)
├── src/
│   ├── api/
│   │   ├── client.ts                 # Base fetch wrapper (mirrors web api-base + api.ts)
│   │   ├── auth.ts                   # /auth/* endpoint methods
│   │   ├── dashboard.ts              # /reports/dashboard + /mobile/dashboard
│   │   ├── invoices.ts               # /invoices/* endpoint methods
│   │   ├── expenses.ts               # /expenses/* endpoint methods
│   │   └── travel-orders.ts         # /travel-orders/* (HR only)
│   ├── components/
│   │   ├── ui/                       # Bilko design-system primitives (Button, Card, Badge,
etc.)
│   │   ├── InvoiceCard.tsx
│   │   ├── ExpenseCard.tsx
│   │   ├── DashboardSummary.tsx
```

```
│   │   ├── CountryBadge.tsx         # Currency + country label display
│   │   └── SyncStatusBar.tsx        # Phase 2 — placeholder only in Phase 1
│   ├── hooks/
│   │   ├── useAuth.ts               # Read auth store + token helpers
│   │   ├── useDashboard.ts
│   │   ├── useInvoices.ts
│   │   ├── useExpenses.ts
│   │   └── useCountryConfig.ts      # Reads org.country → feature flags
│   ├── store/
│   │   ├── authStore.ts             # JWT tokens, user profile, org metadata
│   │   ├── dashboardStore.ts
│   │   ├── invoiceStore.ts
│   │   └── expenseStore.ts
│   ├── services/
│   │   ├── tokenService.ts          # SecureStore read/write/clear for tokens
│   │   ├── imageService.ts          # Resize + compress before upload
│   │   └── countryService.ts        # Country config → feature gate helpers
│   ├── i18n/
│   │   ├── index.ts                 # i18n-js setup + locale detection
│   │   └── messages/                # Symlink to apps/web/messages/ or copy
│   ├── types/
│   │   ├── api.ts                   # Shared API response types (copied from web where
stable)
│   │   └── country.ts               # CountryMobileConfig (from MOBILE-ARCHITECTURE.md)
│   └── utils/
│       ├── currency.ts              # Format currency by country (EUR/RSD/BAM)
│       ├── date.ts                  # Date formatting helpers
│       └── validation.ts            # Form field validators
├── assets/
│   ├── images/
│   │   ├── icon.png                 # 1024x1024 app icon (Bilko "B" logo)
│   │   └── splash.png               # 1284x2778 splash screen
│   └── fonts/
│       ├── WorkSans-*.ttf           # Body font (mirrors web)
│       └── NationalPark-*.otf       # Heading font (mirrors web)
├── app.config.ts                    # Expo config (bundle IDs, env vars, plugins)
├── eas.json                         # EAS Build profiles (dev/staging/production)
├── package.json
└── tsconfig.json                    # Extends @alai/tsconfig
```

Key structural choices:

- `app/` is Expo Router routes. It contains only thin screen files that import from `src/`.
- `src/` contains all business logic, components, hooks, stores, and API calls. Screens import from `src/`. This makes screens testable and composable without touching the router.
- No `src/screens/` directory. Screens live in `app/` as Expo Router requires. Components live in `src/components/`.

# 2.2 Configuration and Environment Variables

Expo uses `EXPO_PUBLIC_*` prefix for variables baked into the JS bundle at build time.

```
EXPO_PUBLIC_API_URL=https://api.bilko.cloud/api/v1
EXPO_PUBLIC_APP_ENV=production
EXPO_PUBLIC_SENTRY_DSN=...
```

All three market environments (HR/BA/RS) use the same API URL (`api.bilko.cloud`). Market behavior is driven by `organization.country` returned from `/auth/me` after login — not by build-time environment variables. There is no per-market build variant at the API URL level.

Build variants (from `MOBILE-ARCHITECTURE.md` section 12):

| EAS Profile | Bundle ID | API URL | Analytics |
|---|---|---|---|
| development | no.alai.bilko.dev | http://localhost:4000/api/v1 (or staging) | off |
| staging | no.alai.bilko.staging | https://api-stage.bilko.cloud/api/v1 | limited |
| production | no.alai.bilko | https://api.bilko.cloud/api/v1 | consent-based |

Bundle IDs are defined in `app.config.ts` via `process.env.APP_ENV` switching. Secrets are injected via EAS Secrets (never committed).

# 2.3 Localization File Strategy

The web app ships five message files under `apps/web/messages/`:

- `en.json` — English (international fallback)
- `hr.json` — Croatian
- `bs.json` — Bosnian
- `sr-Latn.json` — Serbian Latin
- `sr-Cyrl.json` — Serbian Cyrillic

Phase 1 approach: copy these files into `src/i18n/messages/` at scaffold time and commit them alongside the mobile app. They will diverge from the web copies as mobile-specific strings are added (tab labels, camera permission prompts, etc.). A shared `packages/i18n/` extraction is planned for Phase 3 when the divergence cost justifies the shared package overhead.

Locale detection order: device locale via `expo-localization` → org language preference from `/auth/me` → `en` fallback.

---

# 3. Phase 1 Screens (MVP)

## Screen 1: Splash / Auto-login Check

Purpose: entry point, decides whether to route to Login or Today tab.

Behavior:

1. App launches, splash screen is visible (native splash via `expo-splash-screen`).
2. `tokenService.ts` reads Bilko/Entra token state from SecureStore.
3. If no valid session/refresh token: hide splash, navigate to `/(auth)/login`.
4. If a valid session/refresh token exists: refresh through the Entra/Bilko auth bridge. On success: store updated token state, hide splash, navigate to `/(tabs)/today`. On failure: clear all tokens, navigate to `/(auth)/login`.
5. Maximum wait: 5 seconds; show error and navigate to login on timeout.

Implementation note: this logic lives in `app/_layout.tsx` as a root-level effect, not a dedicated screen component. `expo-splash-screen` is kept visible until the auth check resolves.

## Screen 2: Login

Route: `/(auth)/login`

Fields:

- "Continue with Microsoft" / localized primary button.
- Optional environment label for staging builds only.

Behavior:

- Starts Microsoft Entra External ID OIDC Authorization Code + PKCE flow via `expo-auth-session`.
- Receives Entra authorization result, exchanges it per Entra config, then calls Bilko backend auth bridge to validate token and return Bilko `user`, `organization`, role, and

API/session token state.
- On success: store token state via `tokenService.ts`; store `user` and `organization` in `authStore`; navigate to `/(tabs)/today`.
- On failure: display inline error (not alert), localized where possible.
- No mobile registration flow in Phase 1 unless Entra self-service sign-up is explicitly enabled for Bilko.
- No mobile password reset UI in Phase 1; use Entra hosted flow or deep link to web/help.

Design: full-screen, centered card. Bilko plum `#8B6BBF` primary button. Logo at top. Work Sans body font.

# Screen 3: Dashboard (Today Tab)

Route: `/(tabs)/today/index`

Data sources:

- `GET /reports/dashboard` — revenue, expenses, invoice counts (mirrors `api.reports.dashboard()` in web)
- `GET /auth/me` — user name, org name, org country, org currency

Displayed sections:

1. Greeting header ("Dobar dan, [name]" / localized)
2. Revenue this month — formatted in org currency (EUR/RSD/BAM)
3. Expenses this month — formatted in org currency
4. Unpaid invoices — count + total amount
5. Top 3 unpaid invoices by amount — InvoiceCard component (tap → invoice detail)
6. "Capture expense" floating action button → navigates to `/(tabs)/expenses/scan`

Loading state: skeleton placeholders for all cards (no spinner). Error state: inline "Unable to load — tap to retry" per section. Offline state: show cached data with "Last updated [time]" indicator (Phase 1 cache is in-memory store only; full SQLite cache is Phase 2).

Country-specific behavior:

- Currency symbol and format driven by `countryService.ts` → `CountryMobileConfig`
- VAT label ("PDV" for HR/RS/BA, same) from country config
- No travel order card on this screen for BA/RS — only if `supportsHRTravelOrders` in Phase 1

# Screen 4: Invoice Scan (Camera Capture)

Route: `/(tabs)/expenses/scan` (presented as a modal)

Flow:

1. Request camera permission via `expo-camera` (permission rationale: "Bilko treba kameru za snimanje računa").
2. Camera preview fills screen. Shutter button at bottom centre.
3. On capture: preview captured image + confirm/retake buttons.
4. On confirm: `imageService.ts` resizes to max 1920px longest side, JPEG quality 0.85.
5. Show "Add expense details" form: description (required), amount (numeric), date (defaults today), category (select).
6. "Save" calls `POST /api/v1/expenses` (JSON body) then `POST /api/v1/expenses/{id}/documents` (multipart with the compressed image).
7. On success: show brief success toast, dismiss modal, refresh expense list.
8. On upload failure: show error with "Retry" option. Do not discard the image or the draft expense ID.

Phase 1 note: this is capture-and-upload, not OCR extraction. The backend stores the image as `scan_pending`. Amount and description are entered manually by the user.

# Screen 5: Travel Order Quick-Add (HR Only)

Route: `/travel-order/new` (presented as a modal, not a tab)

Gate: only reachable if `useCountryConfig().country === 'HR'`. The "Travel Order" entry point is hidden in the More tab for RS/BA users. No navigation guard is sufficient — the screen itself also checks the gate and shows an error if reached by URL manipulation.

Three-step wizard:

Step 1 — Basic details:

- Destination (text, required)
- Purpose of travel (text, required)
- Departure date + return date (date pickers)

Step 2 — Allowances:

- Daily allowance rate (pre-filled from org HR config, editable)
- Number of days (auto-calculated from date range, user can override)
- Advance payment requested (currency input, optional)

Step 3 — Review + Submit:

- Summary of all fields
- "Submit" button → `POST /api/v1/travel-orders` with the collected payload
- On success: success screen with travel order number, "Done" closes the wizard

This mirrors the web "putni-nalozi" wizard reduced to 3 steps. Full expense attachment for receipts is Phase 2.

# Screen 6: Recent Activity (Expenses Tab)

Route: `/(tabs)/expenses/index`

Displays last 10 invoices and expenses interleaved by date (most recent first).

Data:

- `GET /api/v1/invoices?limit=10&sort=created_desc` via `invoiceStore`
- `GET /api/v1/expenses?limit=10&sort=created_desc` via `expenseStore`

Each row shows: type icon, description/contact name, amount in org currency, date, status badge.

Tap on invoice row → `/(tabs)/invoices/[id]` (invoice detail view, read-only in Phase 1). Tap on expense row → expense detail (read-only in Phase 1 — full edit is Phase 2).

FAB at bottom right: "Scan expense" → navigates to camera capture screen.

# Screen 7: Profile / Settings (More Tab)

Route: `/(tabs)/more/index`

Sections:

1. User info: full name, email, organization name, country flag + name
2. App info: app version (from `expo-constants`), environment (staging/production)
3. Language override: select from `[hr, bs, sr-Latn, sr-Cyrl, en]`
4. "Log out" button (destructive red)

Logout behavior:

1. Call `POST /api/v1/auth/logout` (best effort — do not block logout on API failure).
2. Clear `bilko_access_token`, `bilko_refresh_token`, `bilko_token_expires_at` from SecureStore.
3. Clear all Zustand stores.
4. Navigate to `/(auth)/login`.

---

# 4. API Integration

## 4.1 Auth Flow

```
Microsoft Entra External ID OIDC Authorization Code + PKCE
  app:     expo-auth-session starts hosted login
  success: Entra authorization code/token response

POST /api/v1/auth/entra/session   (Phase 0 backend bridge; see `docs/backend/MOBILE-ENTRA-AUTH-BRIDGE-SPEC.md`)
  body:    { idToken/accessToken or authorizationCode exchange result }
  success: { user, organization, tokens/session: { accessToken, refreshOrSessionToken?, expiresIn } }
  → tokenService.store(...)
  → authStore.setUser(user), authStore.setOrg(organization)

POST /api/v1/auth/entra/refresh or Entra SDK/session refresh
  body:    implementation-specific; no httpOnly browser cookie dependency
  success: { accessToken, expiresIn }
  → tokenService.updateAccessToken(accessToken)

POST /api/v1/auth/logout
  body:    {}  (access token in Authorization header)
  → tokenService.clear()
  → optionally revoke Bilko session / Entra session where supported
```

The current Kotlin/Ktor web auth flow is cookie-oriented: login/register set an httpOnly `refreshToken` cookie and `/auth/refresh` reads `call.request.cookies["refreshToken"]`. Mobile must not depend on that browser cookie path. Phase 0 must add/confirm the Entra External ID bridge and Bilko user/org mapping before Slice A is dispatched. Backend contract: `docs/backend/MOBILE-ENTRA-AUTH-BRIDGE-SPEC.md`.

# 4.2 Automatic 401 Retry

`client.ts` intercepts 401 responses identically to the web `api.ts`:

1. Call `tokenService.getRefreshToken()`.
2. If present: call `/auth/refresh`. On success: update stored access token, retry original request once.
3. On refresh failure: call `authStore.logout()` → clears tokens + navigates to login.
4. Guard: prevent infinite loop on `/auth/refresh` itself responding 401.

# 4.3 User Profile and Org Info

```
GET /api/v1/auth/me
  Authorization: Bearer <accessToken>
  returns: {
    user: { id, email, fullName, role },
    organization: { id, name, country, baseCurrency, language, vatNumber }
  }
```

Called once after login and stored in `authStore`. `organization.country` drives all country-gating logic (HR-only features, currency display, PDV/VAT labels).

# 4.4 Invoice List

```
GET /api/v1/invoices?limit=10&status=&sort=created_desc
  Authorization: Bearer <accessToken>
  returns: { data: Invoice[], total: number, page: number }
```

Phase 1 uses the existing `/api/v1/invoices` endpoint (same as web). A dedicated `/mobile/invoices` BFF endpoint (listed in `MOBILE-ARCHITECTURE.md` section 6) is desirable for Phase 2 to return a mobile-optimized projection. Phase 1 maps the existing response shape in `src/types/api.ts`.

# 4.5 Expense Upload (Multipart)

```
Step 1: POST /api/v1/expenses
  body (JSON): { description, amount, date, category, currency }
  returns: { id, ... }

Step 2: POST /api/v1/expenses/{id}/documents
  body (multipart/form-data): file = <compressed JPEG>
  returns: { uploaded, documentId, url, fileName, message }
```

The web `expenses.uploadDocument()` already implements this exact two-step pattern. Mobile `src/api/expenses.ts` replicates it using `FormData` with `{ uri, name, type }` format for React Native's `fetch` multipart encoding.

Image size constraint: `imageService.ts` enforces max 1920px before upload. If the resulting JPEG exceeds 5 MB (unusual), a second compression pass at 0.7 quality is applied. The backend imposes a 10 MB limit on the documents endpoint.

# 4.6 Travel Orders (HR Only)

```
POST /api/v1/travel-orders
  Authorization: Bearer <accessToken>
  body: {
    destination: string,
    purpose: string,
    departureDate: string,   // ISO date
    returnDate: string,      // ISO date
    dailyAllowanceRate: number,
    numberOfDays: number,
    advancePayment?: number,
    currency: "EUR"           // HR always EUR
  }
  returns: { id, orderNumber, status }
```

If the `/travel-orders` endpoint does not yet exist in the Kotlin/Ktor backend, this is a backend dependency for Slice D. The slice cannot be completed until the backend endpoint is available. Backend team must confirm endpoint availability or create a stub before Slice D dispatch.

---

# 5. Camera and OCR Plan

## Phase 1 (this spec — Slices A–F)

- `expo-camera` captures a still photo.
- `expo-image-manipulator` resizes to max 1920px, JPEG 0.85.
- The compressed image is uploaded as a multipart document attachment to the expense record via `POST /api/v1/expenses/{id}/documents`.
- Backend sets `scanStatus: "scan_pending"` on the document record.
- No extraction data is returned to the mobile client in Phase 1. The user manually enters amount, description, and date.
- User-facing copy: "Slikajte račun — naš tim ili AI će ga procesirati" (localized per market). No claim of live OCR.

## Phase 2 (separate MC — not in this spec)

Backend-side OCR candidates:

- Google Document AI (Croatia: strong Latin script + EUR currency)
- Mindee (pre-built receipt extractor, SaaS, per-page pricing)
- Tesseract 5 + custom post-processing (self-hosted, lower cost, lower accuracy)

- Google ML Kit on-device (free, no data leaves device, but limited to common receipt formats)

The choice depends on data residency requirements (GDPR, BA data sovereignty), per-page cost budget, and accuracy threshold for HR/RS/BA merchant receipt formats. This decision requires a separate product + legal review. Defer to Phase 2 MC.

## Fiken pattern alignment

Fiken's "bare ta bilde med appen, vi foreslår hvordan det skal registreres" pattern is the target UX for Phase 2+. In Phase 1 the equivalent copy is: "Snimite račun odmah, unesite iznos, sinkroniziramo s računovođom" (capture now, enter amount, syncs to accountant).

---

# 6. Native Features

## Biometric Login (Phase 2)

`expo-local-authentication` provides `LocalAuthentication.authenticateAsync()` for FaceID / TouchID on iOS and BiometricPrompt on Android.

Phase 1: not implemented. The setting does not appear in the More tab. Phase 2: add "Enable biometric login" toggle in Settings. On enable: store a biometric-protected flag in SecureStore. On app foreground after lock timeout: prompt biometric before showing sensitive screens. Token is not re-issued — biometric unlocks the in-memory auth state only.

## Push Notifications (Deferred — Phase 3)

Phase 1 does **not** request notification permission, register push tokens, or wire notification handlers. There is no confirmed backend device-token endpoint yet, and prompting users before useful notifications exist is bad UX.

Phase 3 may use `expo-notifications` or direct FCM/APNs after product/security confirms provider choice and backend device-token storage.

## Offline Queue (Phase 2)

Phase 1 has no offline queue. If the user is offline:

- Dashboard shows the last in-memory cached data.
- Capture screen shows an error if the network call fails ("Nema veze — pokušajte ponovo kad budete online").

Phase 2 adds SQLite via `expo-sqlite` and the sync queue described in `MOBILE-ARCHITECTURE.md` sections 7 and 8.

---

# 7. Deployment

## iOS — TestFlight via EAS Build

1. EAS Build runs on Expo infrastructure (no local Mac required).
2. `eas.json` defines three profiles: `development` (simulator build), `staging` (ad-hoc distribution for internal testers), `production` (App Store / TestFlight).
3. Code signing: Distribution Certificate + Provisioning Profile stored in EAS credentials manager. Alem provides Apple Developer account credentials once.
4. TestFlight submission: `eas submit --platform ios --profile staging` after a successful staging build.
5. Phase 1 does not auto-submit to TestFlight from CI. The developer triggers `eas submit` manually after reviewing the build artifact.

## Android — Internal Track via EAS Build + Play Console

1. EAS Build produces an AAB (Android App Bundle).
2. Upload artifact to Google Play Console Internal Testing track manually for Phase 1.
3. Phase 1 does not use `eas submit --platform android` automatically. Manual upload to Play Console.
4. Signing key: generated by EAS and stored in EAS credentials (not in the repo).

## CI — GitHub Actions

Phase 1 CI runs on every push to `feature/bilko-mobile-*` and `main` branches:

```
jobs:
  mobile-check:
    runs-on: ubuntu-latest
    steps:
      - TypeScript type check (tsc --noEmit)
      - ESLint
      - Vitest unit tests (src/services/, src/store/, src/utils/)
      - EAS Build trigger for staging profile (manual workflow_dispatch only in Phase 1)
```

No auto-submit to TestFlight or Play Console in Phase 1. CI produces a build artifact URL. Full auto-submit pipeline is Phase 3 (after E2E tests are in place).

---

# 8. Implementation Slices

All slices are independent dispatch units. Each has a clear done-state, measurable acceptance test, and a named API dependency that must be confirmed green before the slice starts.

## Slice A — Expo Scaffold + Login + JWT Auth

Scope:

- `apps/mobile/` directory created with Expo managed SDK 52+.
- `app.config.ts` with three build profiles.
- `src/api/client.ts` — base fetch wrapper with 401 auto-refresh.
- `src/services/tokenService.ts` — SecureStore read/write/clear.
- `src/store/authStore.ts` — user/org/session state; no raw tokens in Zustand.
- `/(auth)/login.tsx` — Entra "Continue with Microsoft" login, error display, loading state.
- `app/_layout.tsx` — splash guard (auto-login check on cold start).
- Localization scaffold: `i18n-js` setup, English + Croatian strings for login screen.

Done when: a developer can run the app on iOS Simulator, complete Entra External ID login with a real Bilko staging account, be taken to a blank `/(tabs)/today` placeholder screen, and log out successfully. Tokens/session material confirmed in SecureStore (not AsyncStorage/Zustand).

API dependencies: Entra tenant/app registration + Bilko backend auth bridge for token validation, user/org provisioning, role mapping, and session/API token issuance.

## Slice B — Dashboard + Invoice/Expense Read-Only Lists

Scope:

- `/(tabs)/today/index.tsx` — Dashboard screen (revenue, expenses, unpaid invoices, top 3 invoices).
- `/(tabs)/invoices/index.tsx` — Invoice list (last 10 by date).
- `/(tabs)/invoices/[id].tsx` — Invoice detail (read-only: contact, line items, total, status).
- `/(tabs)/expenses/index.tsx` — Expense list (last 10 by date, no scan button yet).
- `src/store/dashboardStore.ts`, `invoiceStore.ts`, `expenseStore.ts`.
- `src/components/DashboardSummary.tsx`, `InvoiceCard.tsx`, `ExpenseCard.tsx`, `CountryBadge.tsx`.
- Currency formatting by org country (EUR/RSD/BAM) in `src/utils/currency.ts`.

Done when: authenticated user sees live data from staging API on the dashboard and can scroll the invoice/expense lists. HR company shows EUR, RS shows RSD, BA shows BAM.

API dependencies: `GET /api/v1/reports/dashboard`, `GET /api/v1/invoices`, `GET /api/v1/expenses`, `GET /api/v1/auth/me`.

# Slice C — Invoice Scan (Camera + Upload as Expense)

Scope:

- `/(tabs)/expenses/scan.tsx` — camera capture modal.
- `src/services/imageService.ts` — resize + compress.
- FAB on expenses list and dashboard to trigger scan.
- Two-step: create expense (POST /expenses), then upload document (POST /expenses/{id}/documents).
- Error handling: retry on upload failure, preserve draft expense ID.
- Success toast + expense list refresh.

Done when: user can point camera at a paper invoice, confirm the photo, enter description + amount + date, save, and see the expense appear in the list with a document attachment. Verified on both iOS Simulator and Android Emulator.

API dependencies: `POST /api/v1/expenses`, `POST /api/v1/expenses/{id}/documents` (multipart). Both are live in the existing backend.

# Slice D — Travel Order Quick-Add (HR Only)

Scope:

- `/travel-order/new.tsx` — 3-step wizard.
- `src/api/travel-orders.ts` — `POST /api/v1/travel-orders`.
- `src/hooks/useCountryConfig.ts` — gates the entry point to HR only.
- Entry point: "Putni nalog" menu item in the More tab, visible only when `org.country === "HR"`.

Done when: HR-country test account can complete the 3-step wizard and receive a travel order number. RS/BA accounts do not see the entry point. Verified that navigating directly to the route on a non-HR account shows a clear "not available" message.

API dependencies: `POST /api/v1/travel-orders` — backend must confirm this endpoint exists or create a stub. This is the highest-risk dependency for Phase 1; flag to backend team immediately.

# Slice E — i18n + Market Detection + Settings Screen

Scope:

- `/(tabs)/more/index.tsx` — Profile, language selector, app version, logout.
- Complete i18n coverage for all Phase 1 screens in five locales: `hr`, `bs`, `sr-Latn`, `sr-Cyrl`, `en`.
- `src/i18n/` populated with mobile-specific strings not in the web messages (camera permission prompts, upload error copy, travel order wizard steps).
- Language override persisted in `expo-secure-store` as `bilko_locale_override`.
- Locale applied to number formatting (Intl.NumberFormat) and date formatting (Intl.DateTimeFormat) via country config.

Done when: switching language in Settings immediately re-renders all visible text without app restart. All five locales render without missing key warnings. Currency format is correct per org country independent of device locale.

API dependencies: none beyond `GET /api/v1/auth/me` (already in Slice B).

# Slice F — TestFlight Build + Internal Share

Scope:

- `eas.json` staging profile wired.
- GitHub Actions workflow: `mobile-check.yml` (lint + type-check + unit tests on every PR).
- `eas build --platform all --profile staging` produces valid IPA + AAB artifacts.
- IPA submitted to TestFlight internal group. AAB uploaded to Play Console internal track.
- Build metadata: bundle version, build number, changelog entry.

Done when: at least two internal testers (Alem + one developer) have installed the app via TestFlight on iPhone and via Play Console on Android, completed the login flow, and confirmed the dashboard loads.

API dependencies: staging API must be reachable from TestFlight/emulator devices.

---

# 9. Out of Scope — Phase 1

The following are explicitly deferred. They must not be partially implemented in Phase 1 slices:

- On-device OCR. No ML Kit, no Tesseract, no cloud OCR call from mobile. Upload raw image + manual entry only.

- Push notifications, permission prompts, device-token registration, handlers, and deep link routing. Entire push scope is Phase 3.
- Offline SQLite queue. No `expo-sqlite` installation in Phase 1. In-memory state only. Phase 2.
- Biometric auth (FaceID / TouchID). Not in Settings in Phase 1. Phase 2.
- Apple Pay / Google Pay. Not applicable to Phase 1 scope. Future.
- Calculator widget / home screen widgets. Not applicable.
- Invoice draft creation from mobile (simple invoice). Phase 4 per PRD.
- Approval actions (approve/reject expense or invoice). Phase 3 per PRD.
- SEF status views (RS). Phase 5 per PRD.
- eRačun/HR-FISK status views (HR). Phase 5 per PRD.
- Company switcher (multi-org accounts). Phase 2 (after single-org login is stable).
- Bank feed / Tok integration. Not in mobile scope for any Phase.

---

# 10. Risks and Mitigations

| Risk | Likelihood | Impact | Mitigation |
|---|---|---|---|
| Entra External ID bridge is not implemented in Bilko backend | High | Blocker for Slice A | Phase 0 backend task: Entra app registration, token validation, user/org mapping, role mapping, session/API token issuance, logout/revocation behavior. |
| `/travel-orders` endpoint does not exist in Kotlin/Ktor backend | High | Blocks Slice D | File backend ticket immediately. Slice D cannot start until endpoint is confirmed or stubbed. |
| Token accidentally written to AsyncStorage by a library (e.g., React Native MMKV default) | Low | Critical (security) | Audit all third-party library storage usage. No AsyncStorage import in `src/services/tokenService.ts`. CI lint rule to flag AsyncStorage imports outside explicitly allowed files. |
| Image upload exceeds backend/documented size limit on a high-res device | Medium | User-facing error | `imageService.ts` enforces max 1920px + second-pass compression if result > 5 MB. Align backend/docs limit before Slice C; current backend evidence showed 20 MB while older docs say 10 MB. |

| Risk | Likelihood | Impact | Mitigation |
|---|---|---|---|
| Travel order screen reachable by RS/BA users via direct URL | Low | UX confusion | Screen-level guard in `/travel-order/new.tsx` checks `org.country` and renders "not available" fallback. Navigation entry point also gated. |
| Expo managed workflow missing a required native capability | Low | Forces bare ejection | Maintain a list of required capabilities before Slice A scaffold. Current Phase 1 requirements (camera, secure store, image manipulator, localization) are all available in managed SDK 52. |
| `expo-secure-store` value size limit (2 KB on some Android versions) | Low | Truncated token | Access tokens (JWTs) are typically 600–900 bytes. Refresh tokens are similar. Both fit within the 2 KB limit. Monitor token size from backend. |
| Bilko design tokens (colors, fonts) not available as an npm-importable package | Medium | Visual inconsistency | Phase 1: copy tokens from `apps/web/tailwind.config.ts` into `src/components/ui/tokens.ts` manually. Phase 2: extract `packages/design-tokens/` shared package. |
| HR/RS/BA locale text not reviewed by native speakers | Medium | Credibility risk in demo | Dževad Jahić (Lexicon) must sign off on BS strings before Slice F / TestFlight build. HR strings reviewed by HR market advisor. |

---

# 11. Backend Dependencies (Pre-Dispatch Checklist)

Before any slice is dispatched, the backend team must confirm the following:

| # | Dependency | Required for | Status |
|---|---|---|---|
| B1 | Microsoft Entra External ID tenant/app registration for Bilko mobile/web customer login | Slice A | Required Phase 0 |

| # | Dependency | Required for | Status |
|---|---|---|---|
| B2 | Bilko backend auth bridge validates Entra token and maps/creates Bilko user + organization + role | Slice A | Required Phase 0 |
| B3 | Bilko API/session token issuance works without browser httpOnly refresh-cookie dependency for React Native | Slice A | Required Phase 0 |
| B4 | `GET /auth/me` returns `organization.country` and `organization.baseCurrency` | Slice B | Likely exists — confirm field names |
| B5 | `GET /reports/dashboard` field names match mobile dashboard mapping ( `cashBalance`, `revenueMTD`, etc.) | Slice B | Existing backend uses web field names — adapt mobile |
| B6 | `POST /expenses/{id}/documents` accepts multipart `{ uri, name, type }` format from React Native fetch | Slice C | Likely exists (web uses it) — confirm RN FormData compat |
| B7 | `POST /travel-orders` endpoint exists with the field schema in section 4.6 | Slice D | Exists in Kotlin/Ktor; confirm mobile payload shape |

---

# 12. Open Architecture Questions (Carry Forward to Phase 2)

These are not blockers for Phase 1 but must be resolved before Phase 2 dispatch:

1. Should the mobile app consume the existing `/api/v1/*` endpoints or new `/mobile/*` BFF endpoints? The architecture doc recommends `/mobile/*` for projection efficiency. Phase 1 uses existing endpoints as a pragmatic shortcut. A decision is needed before Phase 2 to avoid building on the wrong base.
2. Which OCR provider for Phase 2? Requires product + legal + cost review (GDPR, BA data residency, per-page pricing).
3. Local database encryption approach under Expo constraints (`expo-sqlite` + SQLCipher or plaintext with field-level encryption for sensitive columns).
4. Maximum local retention period for cached financial data and attachment files (legal/data retention policy).

5. Expo Notifications vs direct FCM/APNs for Phase 3. Expo Notifications is faster but less flexible for compliance markets that may require local notification routing.

# ADR-037 — Bilko Native Mobile + Entra External ID Auth

# ADR-037 — Bilko Native Mobile Companion and Entra External ID Auth

> **Status:** Accepted for mobile direction / Phase 0 backend work required
> **Date:** 2026-06-04
> **Scope:** Bilko customer-facing mobile companion app for iPhone and Samsung/Android

## Context

Bilko has existing web-first documentation that previously assumed a PWA path for MVP mobile support. Dedicated mobile docs later proposed React Native + Expo, but the architecture doc still marked the decision as recommended/not final and the implementation spec was marked ready for dispatch despite unresolved backend/auth blockers.

The product direction is now native mobile delivery for iPhone and Samsung/Android, not PWA as the primary mobile companion path.

Current backend auth evidence reviewed before this ADR:

- `apps/api/src/main/kotlin/no/alai/bilko/routes/AuthRoutes.kt` uses httpOnly refresh-cookie behavior for web.
- `apps/web/lib/api.ts` uses browser-oriented `credentials: include` refresh handling.
- React Native must not depend on browser httpOnly cookie refresh semantics.

## Decision

1. Bilko Mobile Companion will use **React Native + Expo** for iOS and Android.
2. The previous **PWA-for-MVP** assumption is superseded for the mobile companion workstream. Responsive/PWA web may still exist as web capability.
3. Customer login for mobile should use **Microsoft Entra External ID** with **OIDC Authorization Code + PKCE**.
4. Bilko backend must provide an auth bridge that validates Entra identity, maps/creates Bilko user/org/role context, and issues/refreshes Bilko API/session tokens without relying on browser-only cookies.
5. Phase 1 mobile build is not dispatch-ready until Phase 0 auth/backend prerequisites are closed.

# Phase 0 Backend Prerequisites

- Entra External ID tenant/app registration for Bilko customer login.
- Redirect URI configuration for Expo development, staging, and production builds.
- Backend token validation against Entra issuer/audience/JWKS.
- Bilko user, organization, and role mapping/provisioning.
- Mobile-safe session/API token issuance and refresh path.
- Logout/revocation behavior definition.
- Document upload size limit aligned between backend and docs.

# Phase 1 Scope Boundary

Phase 1 includes:

- Entra/Bilko login and logout.
- Today dashboard and read-only invoice/expense lists.
- Receipt camera capture and expense attachment upload.
- Basic HR travel-order quick-add only if backend payload is confirmed.

Phase 1 excludes:

- Durable offline queue / SQLite sync.
- OCR extraction.
- Push notifications or push-token registration.
- Full `/mobile/*` BFF implementation unless separately scoped.
- Mobile regulatory e-invoice submission logic.

# Consequences

- Mobile docs and the tech-stack spec must treat PWA notes as legacy/superseded for this workstream.
- Web Zustand stores can be mirrored/adapted, but not reused verbatim because mobile auth/storage differs.
- Existing `/api/v1/*` endpoints are preferred for Phase 1 where live; `/mobile/*` BFF is a Phase 2+ decision.
- Security review is required before production rollout because customer identity and financial data are involved.

# Related Documents

- `docs/mobile/MOBILE-PRD.md`
- `docs/mobile/MOBILE-ARCHITECTURE.md`
- `docs/mobile/MOBILE-IMPL-SPEC-PHASE1.md`
- `/Users/makinja/system/specs/bilko-tech-stack.md`

# Bilko Mobile Entra Auth Bridge Spec

# Bilko Mobile — Entra External ID Auth Bridge Spec

> **Status:** Phase 0 backend prerequisite
> **Date:** 2026-06-04
> **Related:** `docs/architecture/ADR-037-BILKO-MOBILE-NATIVE-ENTRA-AUTH.md`, `docs/mobile/MOBILE-IMPL-SPEC-PHASE1.md`

## Purpose

Enable Bilko native mobile login for iPhone and Samsung/Android via Microsoft Entra External ID without relying on browser httpOnly refresh cookies.

The existing web cookie flow must remain intact until a separate web migration is explicitly approved.

## Target Flow

1. Mobile app starts Microsoft Entra External ID login using OIDC Authorization Code + PKCE (`expo-auth-session`).
2. Mobile receives an Entra ID token / auth result.
3. Mobile calls Bilko backend auth bridge with the Entra token.
4. Backend validates Entra token issuer, audience, expiry, signature, and required claims using Entra JWKS.
5. Backend maps the external identity to Bilko `user`, `organization`, and role.
6. Backend issues Bilko API/session tokens suitable for React Native secure storage.
7. Mobile stores token/session material only in `expo-secure-store`.

# Required Endpoints

## `POST /api/v1/auth/entra/session`

Creates or resumes a Bilko session from a validated Entra identity.

Request:

```
{
  "idToken": "<entra-id-token>",
  "client": "mobile",
  "device": {
    "platform": "ios|android",
    "appVersion": "1.0.0"
  }
}
```

Response:

```
{
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "fullName": "Full Name",
    "role": "owner|admin|accountant|viewer"
  },
  "organization": {
    "id": "uuid",
    "name": "Company Name",
    "country": "HR|RS|BA",
    "baseCurrency": "EUR|RSD|BAM",
    "language": "hr|sr-Latn|sr-Cyrl|bs|en"
  },
  "tokens": {
    "accessToken": "<bilko-api-jwt>",
    "refreshToken": "<bilko-mobile-refresh-token>",
    "expiresIn": 900
  }
}
```

Rules:

- Do not set or require browser cookies for `client: mobile`.
- Do not accept unsigned/unverified JWTs.
- Do not trust email alone as identity key; store Entra issuer + subject/object ID mapping.
- If a matching Bilko user/org cannot be resolved, return an explicit onboarding/not-authorized error; do not silently create an organization.

## `POST /api/v1/auth/mobile/refresh`

Refreshes Bilko API access token from a mobile refresh token stored in Keychain/Keystore.

Request:

```
{
  "refreshToken": "<bilko-mobile-refresh-token>"
}
```

Response:

```
{
  "accessToken": "<bilko-api-jwt>",
  "refreshToken": "<rotated-refresh-token>",
  "expiresIn": 900
}
```

Rules:

- Rotate refresh tokens.
- Reject reused/revoked refresh tokens.
- Bind refresh token to user/session/device metadata where feasible.
- Existing `POST /api/v1/auth/refresh` cookie flow may remain for web.

## `POST /api/v1/auth/logout`

For mobile, revoke the current Bilko mobile refresh/session token and return success. Entra hosted session logout can be added later if required.

# Data Model Requirements

At minimum, persist:

- Entra issuer (`iss`).
- Entra subject/object ID (`sub` / `oid`, exact claim depends on tenant config).
- Bilko user ID.
- Organization membership and role.
- Mobile refresh token hash, expiry, rotation family/session ID, device metadata, revoked timestamp.

Never store plaintext refresh tokens.

# Configuration Requirements

Environment/config values:

- `ENTRA_EXTERNAL_ID_ISSUER`
- `ENTRA_EXTERNAL_ID_AUDIENCE`
- `ENTRA_EXTERNAL_ID_JWKS_URL`
- `ENTRA_EXTERNAL_ID_TENANT_ID` or equivalent tenant/domain identifier
- Allowed mobile redirect URIs for Expo dev/staging/production

No secrets should be committed to the repo.

# Acceptance Criteria

Phase 0 is complete when:

- A real Entra test user can log in from an Expo iOS simulator build and receive Bilko `user`, `organization`, and tokens.
- The same flow works on Android emulator.
- Refresh works without browser cookies.
- Invalid issuer/audience/signature/expired token tests fail closed.
- A user with no Bilko org mapping receives a controlled not-authorized/onboarding response.
- Backend tests cover token validation, org/role mapping, refresh rotation, and logout/revocation.
- Web cookie-based login still passes existing tests.

# Security Review Items

- Confirm Entra External ID tenant policy and MFA/passwordless settings.
- Confirm JWKS caching and key rotation handling.
- Confirm token TTLs and refresh-token max lifetime.
- Confirm logging does not include tokens or PII-heavy claims.

- Confirm Sentry/telemetry scrubbing for auth errors.