Shopify Manager — Plan Tehnic Detaliat (Appendix)

Acest document este **appendix** la PDF-ul „Shopify Manager Roadmap – No Diacritics” pe care l-ai încărcat. Scopul lui: să ofere specificații tehnice complete, suficiente pentru a porni implementarea (MVP -> produs final) fără ambiguități.

---

1. Rezumat executiv

Scop: Construirea unei aplicații Android (Kotlin) tip manager / contabilitate pentru magazine Shopify, cu funcționare offline, sincronizare automată, export rapoarte, integrări curieri și OLX, plus asistent vocal. Arhitectură: modulară, MVVM + Repository Pattern, DI.

Livrabile: cod sursă modular, DB local (Room), sincronizare (WorkManager + Webhooks), UI (Jetpack Compose recomandat), testare automată, CI/CD.

---

2. Stack tehnologic recomandat

- Limbaj: **Kotlin**
- UI: **Jetpack Compose** (favorizează productivitate și modularizare)
- Arhitectură: **MVVM** + **Repository Pattern**
- DI: **Hilt**
- Networking: **Retrofit + OkHttp + Coroutines (suspend)**
- Local DB: **Room** (Entities + DAOs + Migrations)
- Background: **WorkManager** (periodic & one-off), **Foreground Service** pentru procese persistente dacă este nevoie
- Sync în timp real: **Webhooks** (backend microservice) + fallback polling
- Logging/Crash: **Timber** + **Firebase Crashlytics**
- CI/CD: **GitHub Actions** (build, lint, unit tests, deploy beta)
- Testing: **JUnit5**, **MockK**, **Espresso / Compose UI tests**
- Securitate: **EncryptedSharedPreferences / Android Keystore** pentru tokenuri
- Voice: Android **SpeechRecognizer**, **TextToSpeech**, eventual integrare cu un serviciu NLP extern (OpenAI / Dialogflow) pentru asistent avansat

---

3. Structura proiect (modulară)

Sugestie de structuri module (Gradle modules):

```
/ - settings.gradle
app/                  # modulul host (UI, navigation)
core/                 # utilitare comune, extensions
data/                  # implementari API, DB, mappers
domain/                # use-cases, interfețe repository
feature-products/      # ecrane + viewmodels produse
feature-orders/        # ecrane + viewmodels comenzi
feature-sync/          # sincronizare, WorkManager workers
feature-voice/         # asistent vocal
integrations/          # curieri, OLX adaptors
auth/                  # autentificare, roluri
```

Avantaj: fiecare feature poate fi testată și release-uită separat.

---

4. Modele de date esențiale (exemple concise)

Entities (Room)

```
@Entity(tableName = "products")
data class ProductEntity(
  @PrimaryKey val id: Long,
  val title: String,
  val sku: String?,
  val price: Double,
  val inventoryQuantity: Int,
  val updatedAt: Long,
  val synced: Boolean
)
```

DTO (Shopify API)

```
data class ProductDto(
  val id: Long,
  val title: String,
  val variants: List<VariantDto>,
  val updated_at: String
)
```

Mapper: `ProductDto` -> `ProductEntity`.

---

5. API: endpoint-uri și fluxuri (abstract)

- **GET /shopify/products** — preia lista produse (paginated)
- **GET /shopify/orders** — preia comenzi noi/actualizate
- **POST /shopify/inventory_adjust** — actualizare stoc
- **Webhooks**:
- `/webhook/orders/created`
- `/webhook/products/updated`

Notă: Folosește GraphQL pentru eficiență dacă ai nevoie de query-uri selective; REST e ok pentru MVP.

---

6. Sincronizare & strategie offline

Componente: 1. **Room DB** ca sursă de adevăr locală 2. **Repository** care oferă: `getProducts()`, `syncPendingUpdates()` 3. **Queue locală** (tabel `pending_actions`) pentru operații offline (ex: schimbare preț, update stoc) 4. **WorkManager**: - Worker periodic de sincronizare (ex: la 15 min sau configurabil) - Worker one-off la reconectare 5. **Foreground Service** (opțional) pentru sincronizări critice 6. **Webhook receiver** în backend care împinge update către app (via FCM) sau direct prin notificare

Conflict resolution: LWW (last write wins) cu timestamp-uri; arată utilizatorului conflictele importante.

---

7. Background & Notificări

- WorkManager pentru joburi programate
- Pentru notificări instant (ex: comandă nouă): **FCM** + handler local
- Notificări configurabile (sonor, badge, priorități)

---

8. Securitate și management token

- Stochează token-uri în **EncryptedSharedPreferences** sau Keystore
- Refresh token pattern: salvezi refresh token în backend şi foloseşti endpoint de refresh
- Rate limiting & retries (Exponential backoff) pentru API calls
- Permisiuni Android: STORAGE (dacă export CSV local), RECORD_AUDIO pentru asistent vocal (cerere runtime)

---

9. Integrare curieri & OLX

Curieri - Creează un adaptor per curier în modul `integrations/` care expune o interfață comună `CourierService`: - `createShipment(orderId, pickupAddress, items)` - `trackShipment(trackingId)` - Implementări: `SamedayAdapter`, `FanCourierAdapter` (folosesc API-urile publice ale curierilor)

OLX\n- Două opțiuni: 1. **Overlay / WebView** cu autentificarea userului și input direct (rapid) 2. **Integrare API** (dacă OLX oferă) — preferabil pentru automatizare

---

10. Asistent vocal

Faze - Stage 3: implementare locală simplă — comenzi vocale predefinite (ex: "Arată comenzile neprocesate") folosind `SpeechRecognizer` și intent matching manual - Stage 4: asistent avansat — NLP extern (Dialogflow / OpenAI) pentru înțelegere naturală, conversatii contextuale, generare răspunsuri dinamice

Intent examples: `get_unshipped_orders`, `update_inventory`, `create_label`.

---

11. Testare automată

- Unit tests pentru use-cases (domain)
- Repository tests cu Room in-memory
- Integration tests pentru endpoints (mock server)
- UI tests (Compose testing) pentru fluxuri critice
- Coverage target: 60–80% pentru modulele core

---

12. CI/CD

Pipeline GitHub Actions (exemplu): 1. `on: push` pentru branches `develop` / `main` 2. Steps: checkout -> setup JDK -> build -> run unit tests -> lint -> assemble debug 3. On success: deploy artifact to Firebase App Distribution / Play Internal 4. Optional: nightly build + automated UI tests

---

13. Securitate, GDPR & export date

- Asigură export CSV/PDF doar din datele locale criptate temporar
- Include funcționalitate de ștergere / anonimizare date (GDPR)
- Log retention policy + acces role-based

---

14. Roadmap granular (sprinturi recomandate)

Sprint 0 — Setup (3–5 zile) - Repo, modulare, CI pipeline minimal, dependințe - Config proiect, Hilt, Retrofit, Room

Sprint 1 — Stage 1 (MVP) (2–3 săptămâni) - Autentificare Shopify (token), preluare produse paginated - UI minimal listă produse + detaliu - Export CSV - Background sync trivial (WorkManager)

Sprint 2 — Stage 2 (offline + UX) (3–4 săptămâni) - Room cache + queue offline - Search & sort, backup local automat - Crashlytics, logging, optimize

Sprint 3 — Stage 3 (automations + voice basic) (4–6 săptămâni) - Pending actions queue sync robust - Webhooks / FCM integration - Simple voice intents + basic report export (PDF)

Sprint 4 — Stage 4 (integrations + security) (4–8 săptămâni) - Auth roles, multi-store - Courier adapters, OLX integration - Advanced voice (NLP), end-to-end tests

---

15. Task examples (ticket-ready)

- **TASK:** Implement ProductEntity + ProductDao + migrations
- **TASK:** Retrofit interface `ShopifyApi` + Auth Interceptor
- **TASK:** Worker `SyncProductsWorker` (WorkManager)
- **TASK:** PendingActionsRepository + processing logic
- **TASK:** Simple Compose screen: ProductList -> ProductDetail
- **TASK:** Implement `SamedayAdapter` skeleton
- **TASK:** Add EncryptedSharedPreferences wrapper

Fiecare task: descriere, definition of done, estimare story points.

---

16. Exemple de API call (Retrofit skeleton)

```
interface ShopifyApi {
  @GET("/admin/api/2024-10/products.json")
  suspend fun getProducts(@Query("page_info") pageInfo: String?):
ProductListResponse
}

class AuthInterceptor: Interceptor {
  override fun intercept(chain: Interceptor.Chain): Response {
    val request = chain.request().newBuilder()
      .addHeader("X-Shopify-Access-Token", token)
      .build()
    return chain.proceed(request)
  }
}
```

---

17. Observații & riscuri

- **Rate limits Shopify** — folosește GraphQL sau throttling cu backoff

- **Divergențe offline** — definește clar politica de conflict resolution
- **Dependențe curieri / OLX** — API-uri externe pot fi instabile; izolează adaptori
- **Permisiuni Android** — explică utilizatorului de ce cere app permisiuni critice (mic dialog UX)

---

18. Pași imediat următori (ce pot face eu acum)

1. Generez un **spec sight-to-code** (scaffold proiect Android + modules) — structură Git + Gradle files
2. Creem backlogul de taskuri în format Issues (GitHub) cu estimări
3. Generăm diagrame MVVM & flow (PNG / PDF) și le atașez la document

Spune-mi care dintre opțiunile de la punctul 18 vrei să fac imediat; pot începe instant cu scaffold-ul proiectului (structură Gradle + sample code pentru ProductEntity + Retrofit + Worker).

Document creat ca anexă tehnică. Conținutul de mai sus este menit să fie folosit împreună cu roadmap-ul din PDF-ul original.