

Chatübergabe 11,12.09,13.09. zu 14.09



Übergabebericht – Advanced Persona System & Migration

(Stand: 12. September 2025)

1. Kontext

Dieser Chat dokumentiert die aktuelle Entwicklungsphase rund um **Matbakh.app** – insbesondere das **Advanced Persona System** und die vollständige Migration von **Supabase/Vercel** zu **AWS**.

Ziel: Das System auf AWS Cognito + RDS stabilisieren, Persona Detection mit AI-Integration produktionsreif machen und Frontend/Backend vollständig synchronisieren.

2. Was bisher geschah

2.1 Advanced Persona System

- **Task 9.0–9.8 abgeschlossen**
 - Advanced Persona Detection Engine
 - Behavioral Engine (AIDA + Psychology Trigger)
 - Claude Prompt Integration
 - Onboarding Integration
 - Frontend SafePersona Hooks
 - Database Schema & Analytics
 - Governance & DSGVO Testimonial Management
 - Teststrategie für Staging
- **Staging Test Infrastructure erstellt**

- Load-Tests
- Monitoring (CloudWatch)
- Staging Deployment Guide
- **Frontend Integration gestartet**
 - useSafePersona Hook
 - PersonaContext / SafePersonaLoader
 - PersonaAdaptiveUI und PersonaDebug Components
- **Build erfolgreich** (Vite React ShadCN)
 - Einheitliche Provider-Struktur
 - AuthContext auf AWS Cognito angepasst
 - i18n integriert

2.2 Migration Supabase → AWS

- Auth-Layer in Frontend bereits auf AWS Cognito umgestellt
- Supabase-Referenzen in Services, Env Variablen und UI noch teilweise vorhanden
- Vercel vollständig deaktiviert, aber Debug UI zeigt noch alte Strings
- Persona Detection Endpoint `/api/persona/detect` lokal nicht erreichbar, weil Login-Flow fehlt (kein Cognito-User Session)

2.3 Tests & Debugging

- Jest Tests liefern nach mehreren Fixes (getWidgetPriority, getCTAStyle, Typenbereinigung etc.)
- 6 Tests failing, 9 passing (Persona Detection/Manual Override/TrackConfidence noch nicht konsistent)
- Staging Tests 100% fehlgeschlagen (keine Verbindung zu API-Staging wegen DNS/Auth)

3. Warum es geschah

- **Projektziel:** Advanced Persona System produktionsreif machen, AI-gestützte Personalisierung, DSGVO-konform

- **Migration:** Supabase und Vercel abschalten, volle Kontrolle via AWS
- **Fehlerursache:** Migration nicht vollständig abgeschlossen → Supabase-Services & Env Variablen noch aktiv, fehlender Login-Flow → Persona Detection blockiert
- **Strategie:** Erst Advanced Persona System bauen & dokumentieren, dann Supabase-Reste beseitigen, dann Tests & Login finalisieren

4. Aktueller Status

Bereich	Status
Advanced Persona Backend	✅ Production-ready
Frontend Persona Hooks	🟡 Integration in Arbeit
AWS Auth Integration	✅ Cognito aktiv, Login-UI fehlt
Supabase Services	❌ Noch nicht bereinigt
Env Cleanup	❌ SUPABASE_* Variablen noch da
Tests	🟡 60% Pass, Persona Detection Tests failen
Staging	❌ Kein API-Zugriff, DNS/Auth Probleme

5. Was noch ansteht

5.1 Tasks.md fertigstellen

- Alle offenen Migrationstasks dokumentieren (ProfileService, ScoreHistory, BenchmarkComparison, Env Cleanup, UI Strings)
- Acceptance Criteria und Done-Definition je Task festlegen

5.2 Migration vervollständigen

- Supabase Services auf AWS RDS umstellen
- Env Variablen aufräumen
- UI Strings aktualisieren („AWS Auth“ statt „Supabase Auth“)
- Tests auf AWS umstellen

5.3 Login-Flow implementieren

- AWS Cognito Hosted UI oder Custom Login Form einbauen

- AuthContext Session setzen → Persona Detection aktivieren
- End-to-End Tests ausführen

5.4 Tests & QA

- Jest Tests aktualisieren (AWS statt Supabase)
- Staging Tests reaktivieren (DNS & Credentials fixen)
- Load Tests mit Cognito User Sessions laufen lassen

5.5 Deployment

- Nach erfolgreicher Migration → Staging Deployment
 - Health Monitoring & KPI Tracking aktivieren
 - Production Deployment vorbereiten
-

6. Wichtigste Hinweise für den neuen cChat

- **Fokus auf Roadmap:** Erst Tasks.md → dann Migration → dann Tests
 - **Nichts deployen ohne funktionierenden Login** → sonst Persona Detection blockiert
 - **Credentials:** AWS SSO verwenden (matbakh-dev), Token regelmäßig erneuern
 - **Check Supabase-Reste:** Alle Services auf AWS RDS umstellen
 - **Debug UI Strings:** Manuell auf AWS umstellen, um Verwirrung zu vermeiden
-

7. Quick Wins für den Start

- **Login UI einbauen** → sofortige sichtbare Verbesserung
 - **Env Cleanup** → Build transparenter
 - **Persona Detection Test-User** in useSafePersona Hook → schneller Test
 - **Supabase Services sukzessive ersetzen** → weniger Legacy
-

8. Zusammenfassung

Der Chat hat das **Advanced Persona System** gebaut, auf AWS Cognito umgestellt und eine Testinfrastruktur erstellt.

Was fehlt: **fertige Taskliste**, **Supabase-Reste entfernen**, **Login-Flow** bauen und **Tests stabilisieren**.

Wenn der neue cChat diese Schritte in der Reihenfolge abarbeitet, ist er nach kurzer Einarbeitung in der Lage, das Projekt **vollständig zu übernehmen und weiterzuführen**.

Willst du, dass ich **diesen Übergabebericht gleich in Markdown-Datei exportiere** (z. B. `handover-advanced-persona.md`), damit du ihn direkt in Git ablegen kannst?

ChatGPT can make mistakes. Check important info. See Cookie Preferences.



Übergabebericht an den nächsten cChat (Stand: 12.09.2025)

1. Überblick & Zielsetzung

Matbakh.app ist eine AWS-basierte Plattform, die Gastronomen Sichtbarkeits-Analysen und strategische Empfehlungen liefert (Google Business, Instagram, Benchmarks, Forecasting).

Wir haben im letzten Sprint einen **großen Umbruch** gemacht:

- **Supabase/Vercel Migration vollständig abgeschlossen**
- **Alle Services auf AWS RDS + Cognito** umgestellt
- **Test-Infrastruktur modernisiert (Jest statt Vitest)**
- **Mocks & Polyfills implementiert, sodass wir echte Businesslogik testen können**

Ziel jetzt: **Stabile Basis** für Features (Decoy Pricing System Task 10.1) und für KI-Agenten.

2. Infrastruktur – aktueller Status

Komponente	Status	Details
Frontend	React + Vite + Tailwind	Keine Supabase-Referenzen mehr, komplett AWS-basiert
Backend	AWS Lambda + AWS RDS (Postgres) + Cognito	Alle Services laufen über AWS SDK Clients
Auth	Cognito User Pools	Supabase Auth entfernt
Datenbank	AWS RDS Postgres	ProfileService, ScoreHistoryService, BenchmarkComparisonService migriert
Dateispeicher/CDN	S3 + CloudFront (matbakh.app + www.matbakh.app)	Alle Assets dort
AI/Bedrock	Claude 3.5 Sonnet (AWS Bedrock) + Prompt Templates	Claude-Integration für Visibility Check & Benchmarks
Tests	Jest 29.7.0 + ts-jest	Vitest komplett abgelöst
Polyfills	import.meta.env, TextEncoder/Decoder etc.	In setupTests.ts + polyfill-importmeta.js

Besondere IDs:

- AWS Account: 055062860590
- Region: eu-central-1
- S3 Bucket: matbakhvcstack-webbucket12880f5b-svct6cxfbip5
- CloudFront ID: E2W4JULEW8BXSD
- Supabase Projekt ref: uheksobnyedarrpgxhju (archiviert)

3. Test-Infrastruktur – aktueller Status

Erledigt (Tasks 1 & 2)

- Jest + ts-jest eingerichtet
- setupTests.ts bereinigt:
 - Polyfills für TextEncoder/TextDecoder
 - AWS SDK Mocks für RDS/CloudWatch/SNS
 - globale Test Utilities

- Alle Vitest → Jest Migration abgeschlossen
- ScoreHistoryService: 15/15 Tests erfolgreich

Offene Punkte

- **VC Service Tests (`vc.test.ts`):** `import.meta.env` wird noch vor Polyfill geparkt → SyntaxError. Lösung: **polyfill-importmeta.js** in `setupFiles` laden (wir haben es vorbereitet, nur noch committen).
 - **BenchmarkComparisonService Tests:** wirft `records undefined` → RDS-Mock muss für BenchmarkComparison angepasst werden.
 - **TrendFilters/TrendChart/EventAnnotations Tests:**
 - `vi.fn()` zu `jest.fn()`
 - Event-Datenstruktur angleichen (`label` fehlt)
 - TSX Transformation konfigurieren (transformIgnorePatterns erweitern)
-

4. Aktueller Stand bei den Services

4.1 ScoreHistoryService

- Vollständig auf AWS RDS umgestellt
- Tests laufen durch
- Mock korrekt implementiert (`executeQuery` , `mapRecord`)

4.2 BenchmarkComparisonService

- Service umgestellt, Tests aber noch auf alte Supabase Logik eingestellt
- Muss auf RDS-Mock umgestellt werden
- Tests erwarten Supabase `.from().select()` Calls → anpassen auf RDS-Mock

4.3 VC Service

- `vc.ts` verwendet `import.meta.env`
- Polyfill implementiert, aber zu spät geladen
- Lösung: Polyfill in `setupFiles` laden (vor allen Imports)

4.4 Frontend Komponenten

- TrendChart, EventAnnotations, TrendFilters nutzen teils noch vi.fn oder falsche Props
 - Eventdaten an neues Schema anpassen (wir haben `label` noch nicht überall ergänzt)
-

5. Polyfills & Mocks

Polyfill import.meta.env

- In `src/polyfill-importmeta.js` definiert (muss ins Repo)
- In `jest.config.cjs` unter `setupFiles` eintragen

AWS SDK Mocks

- `setupTests.ts` stellt Mocks für RDS, CloudWatch, SNS bereit
 - `afterEach` `clearAllMocks` & `resetMocks` aktiviert
-

6. Nächste Schritte (kurzfristig)

1. Polyfill vor Tests laden:

- `setupFiles: ['<rootDir>/src/polyfill-importmeta.js']` in `jest.config.cjs` ergänzen
- Test `vc.test.ts` erneut laufen lassen → `SyntaxError` sollte weg sein

2. BenchmarkComparisonService Tests umstellen:

- Mock für `rdsClient.executeQuery` so bauen, dass `records: [{...}]` zurückkommt
- Alle Tests auf `ScoreBenchmark` statt Supabase Rows anpassen

3. Frontend Tests korrigieren:

- `vi.fn()` → `jest.fn()`
- Eventdatenstruktur angleichen (`label` hinzufügen)
- `transformIgnorePatterns` für TSX erweitern, damit Jest TSX parsen kann

4. Task 10.1 (Decoy Effect Pricing) erst starten, wenn Tests stabil laufen

7. Wichtige Dateien

Datei	Zweck
<code>jest.config.cjs</code>	zentrale Jest-Konfiguration
<code>src/setupTests.ts</code>	Polyfills + globale Mocks (nach dem Laden)
<code>src/polyfill-importmeta.js</code>	Polyfill import.meta.env (vor dem Laden)
<code>src/services/vc.ts</code>	VC Service – nutzt import.meta.env
<code>src/services/__tests__/vc.test.ts</code>	Tests für VC Service
<code>src/services/score-history.ts</code>	ScoreHistory Service
<code>src/services/__tests__/score-history.test.ts</code>	Tests für ScoreHistory Service

8. Was der nächste cChat beachten muss

- **setupFiles vs setupFilesAfterEnv:**

Polyfills wie `import.meta.env` **müssen** in `setupFiles` stehen, sonst SyntaxError.

- **RDS Mock:**

Jede Serviceklasse, die `AwsRdsClient` nutzt, braucht `executeQuery` + `mapRecord` Mock.

- **Frontend Tests:**

TSX + React Testing Library müssen in Jest korrekt transformiert werden.

- **KI/Claude/Bedrock:**

Noch nicht in Tests abgedeckt – später separate Mocks.

- **Versionsmanagement:**

Wir haben eine Datei geplant, in der Kiro alle Toolversionen kennt (noch anlegen).

9. Status Zusammenfassung

Bereich	Status	Kommentar
Supabase Migration	✅ abgeschlossen	Alles auf AWS RDS/Cognito
Jest Migration	✅ abgeschlossen (ScoreHistory)	VC/Benchmark/Frontend noch offen
Polyfills & Mocks	✅ implementiert (setupTests.ts)	import.meta muss in setupFiles

Bereich	Status	Kommentar
Decoy Pricing Feature	 blockiert	erst Tests stabilisieren
Kiro	läuft stabil	braucht Versionsdatei

10. Konkrete Empfehlung für den nächsten Sprint

1. Polyfill & Jest Config fixen → `vc.test.ts` lauffähig
2. BenchmarkComparisonService Tests umstellen
3. Frontend Tests angleichen
4. Danach Decoy Pricing Feature starten

11. Wichtigste Learnings für den nächsten cChat

- **Mocks VOR Imports laden** – Reihenfolge ist entscheidend
- **import.meta.env in Jest** immer polyfillen
- **Supabase Code in Tests nicht mehr erwarten**
- **TSX Dateien brauchen transformIgnorePatterns/ Babel**
- **erst Tests stabil, dann Features**

Damit hast du alle nötigen Infos, um **nahtlos weiterzumachen**.

Der nächste cChat kann sofort auf diese Basis aufbauen und muss sich nur an die Reihenfolge halten:

erst Polyfills/Mocks fixen → dann Tests stabilisieren → dann neue Features.

Willst du, dass ich aus diesem Bericht **gleich eine Markdown-Datei** mache (fertig zum Einchecken ins Repo, z. B. `docs/Übergabebericht-2025-09-12.md`)? (Dann hättest du alles versioniert.)



Übergabebericht für cChat (Stand: 14.09.2025 – 11:45 Uhr)











Rolle

Dieser Bericht übergibt dir die volle Verantwortung als **Product Owner (PO)** und **Chief Technology Officer (CTO)** von `matbakh.app`. Du übernimmst ein komplexes B2B2C-System, das auf Sichtbarkeitsanalysen für Restaurants spezialisiert ist, mit Fokus auf:

- **AI-generierte Empfehlungen** (Claude 3.5 via AWS Bedrock)
- **Google Business + Meta Integration**
- **DSGVO-konformer Data Layer mit Consent- und Audit-System**
- **Einheitlicher VC-Flow (Visibility Check)** inkl. Forecast, Benchmark, SWOT, etc.
- **Kiro-basierte Infrastruktur** auf AWS/Supabase/CloudFront


Aktueller Projektstatus

Migration & Setup (erfolgreich abgeschlossen)

Task	Status	Bemerkung
 Vitest → Jest Migration	 Abgeschlossen	87 Test-Dateien bereinigt, <code>jest.config.cjs</code> eingerichtet
 setupTests.ts konvertiert		Globale Mocks nun via Jest
 polyfill-importmeta.js		ImportMeta-Fehler beseitigt
 package.json aktualisiert		Jest, ts-jest, @jest/globals etc. installiert

Systemanalyse: Kritische Ergebnisse

1. Test-Coverage Schwächen

- Persona API Tests testen falsche Funktionen →  dringend überarbeiten
- VC Service (`vc.test.ts`) nur Platzhaltertest (wurde ersetzt)
- 5 Lambda-Funktionen bislang **ohne jegliche Tests**
- Authentifizierungssystem (Cognito/SimpleAuth) bislang **ungetestet**

2. Legacy-Strukturen in Frontend

- 3+ unterschiedliche **Landing Pages**
- 2+ **Dashboard-Generationen** (Lovable, Vercel, Supabase)
- Mehrere verwaiste Login-Prozesse, Routen und Seiten
- Aktuelle Produktions-UI basiert **noch nicht vollständig auf Kiro-Architektur**

3. Infrastruktur-Verwirrung

- Unterschiedliche Branches (dev, main, legacy-ui)
- Verschiedene Deploys in GitHub, Supabase & Vercel (vor AWS-Migration)
- Rest-APIs teilweise aus Altstruktur noch eingebunden

Neue Komponenten von Kiro (seit Übernahme)

Datei	Inhalt
<code>jest.config.cjs</code>	Neue Jest-Konfiguration
<code>polyfill-importmeta.js</code>	Unterstützung für Jest+ESM
<code>vc.test.ts</code>	Neu geschriebener Test für VC-Logik
<code>report/pre-run-deep-causality.md</code>	Risikobericht vor Testausführung
<code>report/test-rewrite-suggestions.md</code>	Empfehlungen zur Testreparatur
<code>report/source-test-coverage-map.json</code>	Causality Map aller Testverknüpfungen
<code>tasks.md</code>	Task-Dokumentation: aktuell Phase 1 in progress

Aktuelle Risiken

Bereich	Problem	Empfehlung
! Persona API	Interface-Mismatch	Test-Datei ersetzen oder korrekt refactoren
! Frontend API	Mehrfachsystem aktiv	Legacy löschen, neue Kiro-UI konsolidieren
! Lambda Tests	5 ohne Tests	Tests dringend nachrüsten
! Alt-System	Noch aktiv in Branches	Cleanup & Branch Freeze empfohlen
? Google/Meta Auth	Noch nicht final implementiert	Frage: MCP aktivieren oder Auth-only bleiben?

CTO/PO Empfehlung (Next Steps)

Phase 1 – Mapping & Cleanup

- Mapping aller `/api/` , `/services/` , `/pages/` , `/dashboard/` , `/auth/`
- Ermitteln: Nutzung + Ursprung (Lovable, Supabase, Kiro)
- Ziel: Nur Kiro-System weiterführen

Phase 2 – Validierte Test-Ausführung

- Nur getestete, **tatsächlich genutzte** Services ausführen
- `persona-api.test.ts` **ausschließen**

Phase 3 – Branch Cleanup & Frontend-Neustart

- Alte Dashboards, Logins, Auth-Komponenten **löschen**
- Nur Kiro-basierte Views behalten
- UI konsolidieren → "Invisible UI" nach Rāmāni-Vorbild

Zusatzfragen (noch offen)

- **MCP aktivieren?** (für Google + Meta API Management)
 - | Alt-Prompt rät davon ab, Rabieb stellt die Frage zur Prüfung erneut.
- **API Routing finalisieren:** `/track-consent` , `/upload` , `/vc` , `/reports` , etc. → stehen teils in Konflikt mit Alt-Routing

Nächste Schritte für dich (cChat)

1. **Analysiere test coverage map:** `report/source-test-coverage-map.json`
2. **Checke `persona-api.test.ts`**, lösche oder refactor sauber
3. **Starte mit Phase 1 Mapping** (du kannst Kiro anweisen, dies automatisch zu tun)
4. **Leite Legacy-Komponenten aus** `/pages` , `/auth` , `/components` → markiere deprecated

5. Gib grünes Licht für Phase 2 Test Execution, aber nur auf validierten Pfaden

Zusammenarbeit

Rabieb ist als **Business Developer** strategisch und visionär sehr stark. Sie erwartet:

- **klare Tests, verständlich erklärt**
 - **dokumentierte Architekturentscheidungen**
 - **MVP-Ziel: funktionierender Visibility Check, DSGVO-konform, visuell hochwertig**
-