



Cahier des Charges SalamBot v2.1


Guide Complet de Développement et Standards Techniques

 **Version:** 2.1

 **Date:** 2 juin 2025

 **Équipe:** SalamBot Team

 **Contact:** info@salambot.ma

 **Statut:** Document de Référence Officiel

Note de Version 2.1

Cette version corrige les incohérences techniques identifiées dans le feedback du 2 juin 2025, aligne parfaitement la documentation avec l'état réel du dépôt GitHub, et introduit des standards modernes de développement.

Table des Matières

1. [Vision et Positionnement Stratégique](#)
 2. [Architecture Technique Révisée](#)
 3. [Stack Technologique et Versions Figées](#)
 4. [Standards de Développement Modernes](#)
 5. [Structure du Monorepo et Organisation](#)
 6. [Intégration IA et Flows Genkit](#)
 7. [Infrastructure et DevSecOps](#)
 8. [Spécifications Modules Détaillées](#)
 9. [Processus de Développement](#)
 10. [Gestion de Projet et Gouvernance](#)
 11. [Documentation et Communication](#)
 12. [Assurance Qualité et Tests](#)
 13. [Roadmap et Phases de Développement](#)
 14. [Modèle Économique et Pricing](#)
 15. [Annexes Techniques](#)
-

1. Vision et Positionnement Stratégique

1.1 Mission et Vision Actualisée

SalamBot révolutionne l'expérience client au Maroc en démocratisant l'intelligence artificielle conversationnelle pour les PME marocaines. Notre mission transcende la simple automatisation pour créer des interactions authentiques qui respectent la richesse linguistique et culturelle du Royaume, particulièrement à travers la maîtrise du Darija marocain.

La vision 2025-2026 de SalamBot s'articule autour de trois piliers fondamentaux qui définissent notre positionnement unique sur le marché. Le premier pilier, l'Excellence Linguistique, place la compréhension du Darija au cœur de notre différenciation technologique. Cette spécialisation nous permet de servir authentiquement le marché marocain avec une précision linguistique inégalée, dépassant les 88% de précision pour le Darija contre moins de 60% pour nos concurrents internationaux.

Le deuxième pilier, l'Intégration Progressive, reconnaît la réalité des entreprises marocaines qui disposent déjà de systèmes informatiques établis. Notre approche d'intégration non-disruptive via l'extension Chrome et les APIs d'interfaçage permet une adoption graduelle sans bouleverser les workflows existants. Cette stratégie réduit significativement les barrières à l'adoption et facilite la transformation digitale progressive.

Le troisième pilier, la Souveraineté Technologique, répond aux préoccupations croissantes des entreprises marocaines concernant la protection de leurs données. Notre option de stockage souverain et notre conformité stricte avec la loi 09-08 marocaine positionnent SalamBot comme le choix de confiance pour les organisations sensibles à la sécurité des données.

1.2 Différenciation Concurrentielle

L'analyse concurrentielle révèle que SalamBot occupe une position unique sur le marché de l'IA conversationnelle au Maroc. Nos concurrents internationaux (ChatGPT, Claude, Gemini) excellent dans les langues globales mais échouent sur le Darija avec des taux de compréhension inférieurs à 60%. Les solutions locales existantes se limitent souvent à des chatbots basiques sans véritable intelligence artificielle.

Notre différenciation repose sur quatre avantages concurrentiels durables. La Maîtrise du Darija constitue notre moat technologique principal, résultat de deux années de recherche et développement spécialisé. Cette expertise linguistique ne peut être facilement répliquée par des acteurs externes au marché marocain.

L'Architecture d'Intégration Progressive représente notre deuxième avantage concurrentiel. Contrairement aux solutions "tout ou rien" de nos concurrents, SalamBot permet une adoption graduelle qui respecte les investissements technologiques existants des entreprises. Cette approche pragmatique accélère significativement les cycles de vente et réduit la résistance au changement.

La Conformité Réglementaire Native constitue notre troisième différenciateur. Développé dès l'origine pour respecter la loi 09-08 marocaine et le RGPD européen, SalamBot offre des garanties de conformité que les solutions internationales peinent à égaler. Cette conformité native rassure les entreprises et facilite les déploiements dans les secteurs réglementés.

L'Excellence Opérationnelle, notre quatrième avantage, se manifeste par une infrastructure DevSecOps avancée qui dépasse les standards industriels. Notre architecture de sécurité multicouche, notre observabilité temps-réel, et nos processus d'amélioration continue assurent une fiabilité et des performances exceptionnelles.

1.3 Marché Cible et Segmentation

Le marché cible de SalamBot se structure autour de trois segments principaux, chacun avec des besoins spécifiques et des approches d'adoption distinctes. Cette segmentation guide notre stratégie produit, nos efforts commerciaux, et notre roadmap de développement.

Segment PME Digitales (40% du marché adressable) Les PME digitales représentent notre segment de cœur, composé d'entreprises de 10 à 250 employés ayant déjà initié leur transformation digitale. Ces entreprises disposent généralement d'un site web, utilisent des outils CRM basiques, et cherchent à améliorer leur service client sans investissements majeurs.

Ce segment valorise particulièrement l'intégration progressive et la facilité de déploiement. Les décideurs sont souvent les dirigeants eux-mêmes, raccourcissant les cycles de vente mais exigeant une démonstration claire du ROI. Le pricing freemium avec montée en gamme naturelle correspond parfaitement à leurs contraintes budgétaires.

Segment Grandes Entreprises (35% du marché adressable) Les grandes entreprises marocaines et les filiales de multinationales constituent notre segment premium. Ces organisations disposent d'infrastructures IT complexes, de processus établis, et d'exigences strictes en matière de sécurité et de conformité.

Ce segment privilégie la robustesse technique, les capacités d'intégration avancées, et les garanties de service. Les cycles de vente sont plus longs mais les contrats plus

importants. L'option de stockage souverain et les fonctionnalités entreprise sont particulièrement valorisées.

Segment E-commerce et Services (25% du marché adressable) Les plateformes e-commerce, les services financiers, et les entreprises de service constituent notre troisième segment cible. Ces organisations ont des besoins spécifiques en matière d'assistance à l'achat, de support transactionnel, et d'intégration avec leurs systèmes de gestion.

Ce segment valorise les fonctionnalités métier spécialisées, les intégrations e-commerce avancées, et les capacités d'analyse comportementale. La personnalisation et l'adaptation aux workflows spécifiques sont des facteurs clés de succès.

3. Stack Technologique et Versions Figées

3.1 Versions Officielles Alignées avec la Réalité

Suite à l'audit technique du 2 juin 2025, les versions suivantes sont officiellement figées pour assurer la cohérence entre la documentation et l'implémentation réelle du projet SalamBot.

Environnement d'Exécution - Node.js: 20.18.0 LTS (aligné avec le dépôt actuel) - **pnpm:** 9.1.2 (gestionnaire de paquets officiel) - **TypeScript:** 5.4.5 (version stable avec support ESNext)

Framework Principal - Nx: 21.1.2 (orchestrateur monorepo - version déployée) - **React:** 19.0.0 (version cible pour toutes les applications) - **Next.js:** 14.2.5 (version actuelle - migration vers 15.0.3 planifiée Phase 3)

Note de Migration Next.js

Le dépôt utilise actuellement Next.js 14.2.5. La migration vers Next.js 15.0.3 est planifiée pour la Phase 3 comme breaking change majeur. Cette migration nécessitera une validation complète des fonctionnalités et une mise à jour coordonnée de toutes les applications.

Intelligence Artificielle - Genkit: 0.5.8 (orchestrateur IA Google) - **Vertex AI:** Latest (modèles Gemini Pro) - **Llama:** 3.1-70B (modèle spécialisé Darija)

Infrastructure et DevOps - Google Cloud Platform: Latest APIs - **Redis:** 7.2 (via Google Cloud Memorystore) - **Terraform:** 1.8.3 (Infrastructure as Code) - **Docker:** 24.0.7 (containerisation)

3.2 Standards de Commentaires Modernes

Les commentaires d'en-tête de fichiers suivent un format moderne et expressif qui reflète l'innovation et la créativité de l'équipe SalamBot tout en maintenant la lisibilité et l'utilité professionnelle.

```
/**
 *
 * | 🤖 SalamBot - Intelligence Conversationnelle Marocaine
 *
 * |
 * | 📁 Détection automatique de langue avec spécialité Darija
 * |
 * | 🧑 SalamBot Team <info@salambot.ma>
 * |
 * | 📅 Créé: 2025-06-02 | Modifié: 2025-06-02
 * | 🏷️ v1.2.0 | 🔒 Propriétaire SalamBot Team
 *
 */
```

Format Alternatif Minimaliste

```
/**
 * 🚀 SalamBot | Flow de génération de réponses contextuelles
 *
 * @description Orchestration IA pour réponses multilingues
 * (FR/AR/Darija)
 * @author SalamBot Team <info@salambot.ma>
 * @version 1.0.0
 * @created 2025-06-02
 * @license Propriétaire - SalamBot Team
 *
 * ✨ Fonctionnalités: Sélection auto de modèle | Fallback
 * intelligent | Métriques temps-réel
 */
```

Format Technique Avancé

```
/**
 * ⚡ SALAMBOT CORE ENGINE ⚡
 *
 * 🎯 Module: Extension Chrome - Intégration Progressive CRM
 * 🛠️ Stack: Manifest v3 | TypeScript 5.4 | Chrome APIs
```

```
* 👥 Team: SalamBot Engineering <dev@salambot.ma>
* 📊 Metrics: Performance | Security | Compatibility
*
* 🌟 Innovation: Détection auto CRM + Injection IA contextuelle
* 🛡️ Security: CSP compliant | Minimal permissions | Zero
storage
*
* @version 2.1.0-beta
* @since 2025-06-02
* @license Proprietary © SalamBot Team
*/
```

3.3 Configuration des Outils de Développement

La configuration des outils de développement est standardisée pour assurer la cohérence et l'efficacité de l'équipe. Ces configurations sont versionnées et partagées via le monorepo.

ESLint Configuration (.eslintrc.json)

```
{
  "root": true,
  "ignorePatterns": ["**/*"],
  "plugins": ["@nx", "@typescript-eslint", "react-hooks"],
  "extends": [
    "@nx/eslint-plugin-nx/recommended",
    "@typescript-eslint/recommended",
    "plugin:react-hooks/recommended"
  ],
  "overrides": [
    {
      "files": ["*.ts", "*.tsx"],
      "rules": {
        "@nx/enforce-module-boundaries": [
          "error",
          {
            "enforceBuildableLibDependency": true,
            "allow": [],
            "depConstraints": [
              {
                "sourceTag": "scope:app",
                "onlyDependOnLibsWithTags": ["scope:lib",
"scope:shared"]
              }
            ]
          }
        ],
        "@typescript-eslint/no-unused-vars": "error",
        "react-hooks/rules-of-hooks": "error",

```

```

    "react-hooks/exhaustive-deps": "warn"
  }
}
]
}

```

Prettier Configuration (.prettierrc)

```

{
  "semi": true,
  "trailingComma": "es5",
  "singleQuote": true,
  "printWidth": 100,
  "tabWidth": 2,
  "useTabs": false,
  "bracketSpacing": true,
  "arrowParens": "avoid",
  "endOfLine": "lf"
}

```

TypeScript Configuration (tsconfig.base.json)

```

{
  "compileOnSave": false,
  "compilerOptions": {
    "rootDir": ".",
    "sourceMap": true,
    "declaration": false,
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "importHelpers": true,
    "target": "ES2022",
    "module": "esnext",
    "lib": ["ES2022", "DOM"],
    "skipLibCheck": true,
    "skipDefaultLibCheck": true,
    "baseUrl": ".",
    "strict": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "paths": {
      "@salambot/core": ["libs/core/src/index.ts"],
      "@salambot/types": ["libs/types/src/index.ts"],
      "@salambot/ui": ["libs/ui/src/index.ts"],
      "@salambot/auth": ["libs/auth/src/index.ts"],
      "@salambot/ai/lang-detect": ["libs/ai/lang-detect/src/
index.ts"]
    }
  }
}

```

```

    },
    "exclude": ["node_modules", "tmp"]
  }
}

```

3.4 Tableau des Versions Consolidées

Composant	Version Actuelle	Version Cible	Migration	Priorité
Runtime				
Node.js	20.18.0 LTS	20.18.0 LTS	✓ Stable	-
pnpm	9.1.2	9.1.2	✓ Stable	-
TypeScript	5.4.5	5.4.5	✓ Stable	-
Frameworks				
Nx	21.1.2	21.1.2	✓ Stable	-
React	19.0.0	19.0.0	✓ Stable	-
Next.js	14.2.5	15.0.3	↺ Phase 3	P2-High
IA & Backend				
Genkit	0.5.8	1.0.0	↺ Phase 3	P1-Critical
Vertex AI	Latest	Latest	✓ Auto-update	-
Styling				
Tailwind CSS	4.0.0	4.0.0	✓ Stable	-
shadcn/ui	Latest	Latest	✓ Auto-update	-
Infrastructure				
Redis	7.2	7.2	✓ Stable	-
Terraform	1.8.3	1.8.3	✓ Stable	-

4. Standards de Développement Modernes

4.1 Architecture de Commentaires Créative

L'innovation dans les commentaires de code reflète l'esprit créatif et moderne de l'équipe SalamBot. Nous adoptons une approche qui combine utilité technique et expression artistique pour créer une expérience de développement inspirante.

Style "ASCII Art" pour Modules Critiques

```
/**
 *
 * ||
CORE ||          🧠 SALAMBOT NEURAL
 * ||
 * ||
 * || 🎯 Mission: Détection linguistique Darija haute précision
 * ||
 * || 🚀 Engine: Hybrid ML + Heuristics + Cultural Context
 * ||
 * || 📊 Target: >88% Darija accuracy | <200ms latency
 * ||
 * || 🔬 Research: 2+ years Moroccan dialect specialization
 * ||
 * ||
 * || 🧑‍💻 Crafted by: SalamBot AI Research Team
 * ||
 * || ✉️ Contact: ai-research@salambot.ma
 * ||
 * || 📅 Genesis: 2025-06-02 | Last Evolution: 2025-06-02
 * ||
 * || 🏷️ Version: 2.1.0-neural | License: Proprietary
 * ||
 */
```

Style "Emoji Story" pour APIs

```
/**
 * 🌟 SALAMBOT API GATEWAY 🌟
 *
 * 📖 Story: Cette API unifie tous les points d'accès SalamBot
 * 🔄 Characters: Clients → Gateway → Microservices → IA → Réponse
 */
```

```

* 🎬 Plot: Authentification → Validation → Orchestration →
Réponse
*
* 🎯 Superpowers:
*   • 🛡️ Sécurité multicouche (OAuth2 + JWT + Rate limiting)
*   • ⚡ Performance (Cache Redis + CDN + Compression)
*   • 🔍 Observabilité (OpenTelemetry + Grafana + Alerting)
*   • 🌍 Multi-tenant (Isolation + Scaling + Monitoring)
*
* 🏆 Achievements:
*   • 99.9% uptime SLA
*   • <100ms p95 latency
*   • 10K+ req/sec capacity
*
* 👥 Heroes: SalamBot Platform Team <platform@salambot.ma>
* 📅 Epic Started: 2025-06-02
* 🚀 Version: 2.1.0-gateway
*/

```

Style "Technical Poetry" pour Algorithmes Complexes

```

/**
* 🎵 The Darija Detection Symphony 🎵
*
* In the realm where Arabic meets French,
* Where Berber whispers blend and quench,
* Lives a language rich and free—
* Darija, Morocco's poetry.
*
* 🎼 Movement I: Text Preprocessing
*   Normalize the chaos, clean the noise
*   Extract the essence, find the voice
*
* 🎼 Movement II: Pattern Recognition
*   Search for markers, cultural signs
*   Moroccan soul in coded lines
*
* 🎼 Movement III: Confidence Scoring
*   Weight the evidence, measure truth
*   Return the answer, share the proof
*
* 🧑🎨 Composer: SalamBot Linguistic AI Team
* 🏛️ Conservatory: info@salambot.ma
* 📅 Premiered: 2025-06-02
* ⭐ Opus: 2.1.0-symphony
*/

```

4.2 Standards de Qualité et Couverture de Tests

La qualité du code SalamBot repose sur des standards stricts qui assurent la fiabilité, la maintenabilité, et la performance du système. Ces standards sont appliqués automatiquement via les outils de CI/CD et validés lors des revues de code.

Objectifs de Couverture Révisés Suite au feedback du 2 juin 2025, les objectifs de couverture de tests sont ajustés pour refléter la réalité actuelle du projet tout en maintenant une trajectoire d'amélioration continue.

- **Couverture Actuelle:** ~60% (audit Jest)
- **Objectif Phase 2:** 75% (amélioration progressive)
- **Objectif Phase 3:** 85% (standard industriel)
- **Objectif Phase 4:** 90% (excellence technique)

Plan d'Action "Test Debt Sprint" Un sprint dédié à la réduction de la dette technique de tests est planifié avant la release v0.3.0:

```
/**
 * 🎯 TEST DEBT ELIMINATION SPRINT 🎯
 *
 * 📊 Current State: 60% coverage | Target: 75%
 * ⌚ Duration: 2 weeks sprint
 * 🎯 Focus Areas:
 *   • Core business logic (libs/core)
 *   • AI flows (libs/ai/*)
 *   • Critical APIs (apps/functions-run)
 *   • Integration points
 *
 * 🚀 Strategy:
 *   • Day 1-3: Audit & prioritize missing tests
 *   • Day 4-8: Implement high-impact tests
 *   • Day 9-10: Integration & E2E scenarios
 *
 * 🏆 Success Metrics:
 *   • Coverage: 60% → 75%
 *   • CI green: 95%+ success rate
 *   • Performance: No regression
 *
 * 👥 Squad: Full dev team rotation
 * 📅 Scheduled: Before v0.3.0 release
 */
```

4.3 Conventions de Nommage Évolué

Les conventions de nommage SalamBot reflètent notre approche moderne et notre spécialisation dans l'IA conversationnelle marocaine. Ces conventions facilitent la compréhension du code et accélèrent l'onboarding des nouveaux développeurs.

Conventions pour les Flows IA

```
// ✅ Bon: Descriptif et contextualisé
export const detectDarijaLanguageFlow = defineFlow({...});
export const generateContextualReplyFlow = defineFlow({...});
export const escalateToHumanAgentFlow = defineFlow({...});

// ❌ Éviter: Trop générique
export const detectFlow = defineFlow({...});
export const replyFlow = defineFlow({...});
```

Conventions pour les Composants UI

```
// ✅ Bon: Préfixe SalamBot + Fonction claire
export const SalamBotChatWidget = () => {...};
export const SalamBotAgentDashboard = () => {...};
export const SalamBotLanguageSelector = () => {...};

// ✅ Bon: Composants métier spécialisés
export const DarijaDetectionIndicator = () => {...};
export const MoroccanCulturalContextProvider = () => {...};
```

Conventions pour les Types et Interfaces

```
// ✅ Bon: Suffixes clairs et contexte métier
interface ConversationContext {
  userId: string;
  businessId: string;
  language: DetectedLanguage;
  culturalContext: MoroccanContext;
}

type DarijaConfidenceScore = number; // 0-1 range
type EscalationReason = 'low_confidence' | 'complex_query' | 'human_requested';

// ✅ Bon: Enums expressifs
enum MoroccanRegion {
  CASABLANCA = 'casablanca',
  RABAT = 'rabat',
  MARRAKECH = 'marrakech',
}
```

```
FES = 'fes',  
TANGIER = 'tangier'  
}
```

8. API Gateway et Architecture d'Intégration

8.1 État Actuel et Roadmap API Gateway

⚠ **Statut Critique:** L'API Gateway, élément central de l'architecture SalamBot, n'est actuellement pas implémentée selon l'audit technique du 2 juin 2025. Cette lacune représente un risque architectural majeur qui doit être adressé en priorité absolue dans la Phase 2.

L'API Gateway constitue le point d'entrée unifié pour toutes les interactions externes avec la plateforme SalamBot. Son absence actuelle force les applications à communiquer directement avec les microservices, créant un couplage fort et des problèmes de sécurité, de monitoring, et de scalabilité.

Plan de Rattrapage Urgent - Phase 2 Sprint B

Un ticket prioritaire "PoC Gateway (Kong/Tyk)" est ajouté à la roadmap Phase 2 avec les jalons suivants:

```
/**  
* 🚨 MISSION CRITIQUE: API Gateway Implementation 🚨  
*  
* 📊 Impact: Architecture centrale | Sécurité | Performance  
* 🕒 Timeline: Phase 2 Sprint B (4 semaines)  
* 🎯 Deliverables:  
*   • Week 1: PoC Kong vs Tyk evaluation  
*   • Week 2: Gateway core implementation  
*   • Week 3: Security & rate limiting  
*   • Week 4: Integration & testing  
*  
* 🛠 Technical Stack:  
*   • Option A: Kong Gateway (Lua plugins)  
*   • Option B: Tyk Gateway (Go-based)  
*   • Option C: Custom Express.js gateway  
*  
* 🎯 Success Criteria:  
*   • All external APIs routed through gateway  
*   • OAuth2 + JWT authentication implemented  
*   • Rate limiting (1000 req/min per client)  
*   • Request/response logging to Grafana  
*   • <50ms gateway overhead
```

```
*
* 👤 Assigned: Backend Squad + DevOps
* 🚨 Priority: P0-Critical (blocks Phase 3)
*/
```

8.2 Architecture Gateway Cible

L'API Gateway SalamBot implémentera une architecture moderne qui centralise l'authentification, l'autorisation, le rate limiting, et l'observabilité. Cette architecture assure la sécurité, la performance, et la maintenabilité de l'ensemble de la plateforme.

Fonctionnalités Core du Gateway

L'authentification centralisée gère tous les mécanismes d'accès à la plateforme: OAuth2 pour les intégrations tierces, JWT pour les sessions utilisateur, API keys pour les intégrations système, et certificats clients pour les connexions B2B sécurisées. Cette centralisation élimine la duplication de code d'authentification dans les microservices.

Le rate limiting intelligent protège la plateforme contre les abus tout en permettant un usage normal. Les limites sont configurables par client, par endpoint, et par type d'usage (gratuit, payant, entreprise). Le système inclut des mécanismes de burst allowance pour gérer les pics de trafic légitimes.

La transformation de données assure la compatibilité entre les différentes versions d'API et les formats de données. Le gateway peut transformer les requêtes et réponses pour maintenir la rétrocompatibilité tout en permettant l'évolution des APIs internes.

```
/**
 * 🌐 SALAMBOT API GATEWAY ARCHITECTURE 🌐
 *
 * 🔄 Request Flow:
 *   Client → Gateway → Auth → Rate Limit → Transform → Service
 * → Response
 *
 * 🛡 Security Layers:
 *   • TLS 1.3 termination
 *   • OAuth2/JWT validation
 *   • API key management
 *   • IP whitelisting/blacklisting
 *   • Request sanitization
 *
 * ⚡ Performance Features:
 *   • Response caching (Redis)
 *   • Request compression
 *   • Connection pooling
 *   • Circuit breaker pattern
 */
```

```

*  Observability:
*   • Request/response logging
*   • Performance metrics
*   • Error tracking
*   • Business metrics
*/

interface GatewayConfig {
  authentication: {
    oauth2: OAuth2Config;
    jwt: JWTConfig;
    apiKeys: APIKeyConfig;
  };

  rateLimiting: {
    global: RateLimitRule;
    perClient: RateLimitRule;
    perEndpoint: Record<string, RateLimitRule>;
  };

  routing: {
    services: ServiceRoute[];
    loadBalancing: LoadBalancingStrategy;
    healthChecks: HealthCheckConfig;
  };

  observability: {
    logging: LoggingConfig;
    metrics: MetricsConfig;
    tracing: TracingConfig;
  };
}

```

8.3 Extension Chrome - Approche MVP Révisée

Suite au feedback du 2 juin 2025, l'approche de développement de l'extension Chrome est restructurée avec une stratégie MVP progressive qui réduit les risques et accélère le time-to-market.

Phase MVP 1: Auto-suggestion Générique (4 semaines)

La première phase se concentre sur la fonctionnalité core d'auto-suggestion dans les champs texte génériques, sans détection spécifique de CRM. Cette approche permet de valider la valeur utilisateur et la faisabilité technique avant d'investir dans des intégrations complexes.

```

/**
*  CHROME EXTENSION MVP 1: Generic Text Enhancement 

```

```

*
* 🚀 Core Value Proposition:
*   • Detect text input fields on any website
*   • Analyze user input for language and intent
*   • Suggest improved responses using SalamBot AI
*   • Support FR/AR/Darija with cultural context
*
* 📋 MVP Features:
*   • Generic text field detection (textarea, input[type=text])
*   • Context menu integration ("Améliorer avec SalamBot")
*   • Language detection popup
*   • AI-powered suggestion overlay
*   • One-click suggestion insertion
*
* 🛠️ Technical Approach:
*   • Manifest v3 content script
*   • Minimal permissions (activeTab only)
*   • Offline-first with API fallback
*   • Zero data storage (privacy-first)
*
* 🎯 Success Metrics:
*   • Installation rate: >100 beta users
*   • Usage rate: >50% weekly active
*   • Satisfaction: >4.0/5.0 rating
*   • Performance: <500ms suggestion time
*
* 📅 Timeline: 4 weeks (Phase 2 Sprint C)
* 👥 Team: Frontend + Extension specialist
*/

```

Phase MVP 2: CRM Detection Intelligente (6 semaines)

La deuxième phase ajoute la détection automatique des plateformes CRM populaires et l'adaptation contextuelle des suggestions. Cette phase capitalise sur les apprentissages de la Phase MVP 1 pour offrir une expérience plus sophistiquée.

Les patterns de détection CRM utilisent une approche hybride combinant l'analyse d'URL, la détection de signatures DOM, et l'analyse de métadonnées pour identifier les plateformes supportées. Cette détection est progressive et extensible pour faciliter l'ajout de nouvelles plateformes.

```

/**
* 🎯 CHROME EXTENSION MVP 2: Smart CRM Integration 🎯
*
* 🧠 Intelligence Features:
*   • Auto-detect popular CRM platforms
*   • Context-aware suggestion adaptation
*   • Business process integration
*   • Workflow-specific recommendations

```



```

*
* 📊 Supported Platforms (Priority Order):
* 1. Salesforce (highest market share)
* 2. HubSpot (popular with SMEs)
* 3. Zendesk (support focus)
* 4. Freshdesk (growing in Morocco)
* 5. Generic CRM patterns
*
* 🔍 Detection Strategy:
* • URL pattern matching
* • DOM signature analysis
* • Meta tag inspection
* • JavaScript framework detection
*
* 📈 Enhanced Analytics:
* • Platform usage statistics
* • Suggestion acceptance rates
* • User workflow analysis
* • Performance per platform
*/

interface CRMDetectionEngine {
    detectPlatform(context: PageContext): CRMPlatform | null;
    extractBusinessContext(platform: CRMPlatform):
BusinessContext;
    adaptSuggestions(context: BusinessContext, userInput:
string): Suggestion[];
    trackUsage(platform: CRMPlatform, action: UserAction): void;
}

```

13. Roadmap Révisée et Jalons Détaillés

13.1 Phase 2 - Go-to-Market (En Cours - Corrections Critiques)

La Phase 2 est restructurée pour adresser les lacunes architecturales identifiées dans l'audit du 2 juin 2025. Cette phase privilégie la stabilité et la complétude des fondations avant l'expansion fonctionnelle.

Sprint A: Agent Desk Migration (4 semaines) - EN COURS

La migration de l'Agent Desk de React/Vite vers Next.js 15 constitue la priorité absolue pour aligner l'implémentation avec les spécifications du cahier des charges. Cette migration apporte des bénéfices significatifs en matière de performance, SEO, et cohérence architecturale.

```

gantt
  title Phase 2 Sprint A - Agent Desk Migration
  dateFormat YYYY-MM-DD
  section Migration Technique
  Setup Next.js 15 :active, setup, 2025-06-03, 3d
  Nettoyage Vite/Vitest :clean, after setup, 2d
  Integration Tailwind :tailwind, after clean, 2d
  section UI/UX
  shadcn/ui Setup :shadcn, after tailwind, 2d
  next-intl Configuration :i18n, after shadcn, 3d
  Placeholder Pages :pages, after i18n, 2d
  section Validation
  CI/CD Integration :ci, after pages, 2d
  Vercel Configuration :vercel, after ci, 1d
  Testing & Review :test, after vercel, 3d

```

Sprint B: API Gateway Implementation (4 semaines) - CRITIQUE

L'implémentation de l'API Gateway manquante représente le défi technique majeur de la Phase 2. Ce sprint détermine la faisabilité architecturale de l'ensemble de la plateforme.

```

/**
 * 🏗️ SPRINT B: API GATEWAY FOUNDATION 🏗️
 *
 * Week 1: Architecture & Technology Selection
 *   • Evaluate Kong vs Tyk vs Custom solution
 *   • Define gateway requirements & constraints
 *   • Create technical specification document
 *   • Setup development environment
 *
 * Week 2: Core Gateway Implementation
 *   • Basic routing and service discovery
 *   • Health check and load balancing
 *   • Request/response transformation
 *   • Error handling and circuit breaker
 *
 * Week 3: Security & Rate Limiting
 *   • OAuth2/JWT authentication integration
 *   • API key management system
 *   • Rate limiting with Redis backend
 *   • Security headers and CORS handling
 *
 * Week 4: Integration & Performance
 *   • Integrate with existing services
 *   • Performance testing and optimization
 *   • Monitoring and observability setup
 *   • Documentation and deployment
 *
 * 🎯 Deliverables:

```

- * • Production-ready API Gateway
- * • All services migrated to gateway routing
- * • Security policies implemented
- * • Performance benchmarks validated
- */

Sprint C: Extension Chrome MVP 1 (4 semaines)

Le développement de l'extension Chrome suit l'approche MVP révisée avec focus sur la validation de la valeur utilisateur avant l'investissement dans des fonctionnalités complexes.

13.2 Phase 3 - Scale-up & Consolidation (Replanifiée)

La Phase 3 est recentrée sur la consolidation technique et l'amélioration de la qualité plutôt que sur l'expansion fonctionnelle. Cette approche assure des fondations solides pour la croissance future.

Objectifs Révisés Phase 3:

1. **Migration Next.js 15 Globale** - Mise à niveau de toutes les applications vers Next.js 15.0.3 avec validation complète des fonctionnalités
2. **Upgrade Genkit 1.0** - Migration vers Genkit 1.0 avec amélioration des performances IA
3. **Développement Bibliothèques Manquantes** - Implémentation des libs core/, types/, connectors/, persistance/
4. **Extension Chrome MVP 2** - Ajout de la détection CRM intelligente
5. **Fonctionnalités Offline Widget** - Implémentation du mode hors ligne avec synchronisation

```
gantt
    title Phase 3 - Scale-up & Consolidation
    dateFormat YYYY-MM-DD
    section Technical Upgrades
    Next.js 15 Migration      :upgrade, 2025-11-01, 6w
    Genkit 1.0 Upgrade        :genkit, after upgrade, 4w
    section Core Development
    Core Libraries            :libs, 2025-11-01, 8w
    Chrome Extension MVP2     :chrome, after libs, 6w
    section Quality & Performance
    Test Coverage 85%         :tests, 2025-12-01, 4w
    Performance Optimization  :perf, after tests, 3w
    section Features
    Widget Offline Mode       :offline, 2026-01-01, 6w
    Translation Flow          :translate, after offline, 4w
```

13.3 Métriques de Succès et KPIs Révisés

Les métriques de succès sont ajustées pour refléter les priorités révisées et les défis identifiés dans l'audit technique.

Métriques Techniques (Phase 2) - Couverture de Tests: 60% → 75% (objectif réaliste) - **Performance API Gateway:** <50ms overhead - **Disponibilité Système:** 99.5% uptime - **Temps de Build CI/CD:** <5 minutes - **Agent Desk Migration:** 100% fonctionnalités migrées

Métriques Produit (Phase 2) - Extension Chrome Beta: 100+ utilisateurs actifs - **Agent Desk Adoption:** 10+ entreprises pilotes - **Satisfaction Utilisateur:** >4.0/5.0 - **Temps de Résolution:** <2h pour bugs critiques

Métriques Business (Phase 2) - Clients Pilotes: 5+ entreprises signées - **Revenue Pipeline:** 50K MAD/mois - **Churn Rate:** <5% mensuel - **NPS Score:** >50

15. Plan d'Action Immédiat - Prochaines 4 Semaines

15.1 Actions Critiques Semaine 1-2

Priorité P0 - Blockers Architecturaux

1. **Finaliser Agent Desk Migration Next.js 15**
2. Compléter la migration technique en cours
3. Valider toutes les fonctionnalités existantes
4. Mettre à jour la configuration Vercel
5. Tagger la version v0.2.1
6. **Lancer PoC API Gateway**
7. Évaluer Kong vs Tyk vs solution custom
8. Créer le spike technique (2 jours)
9. Définir l'architecture cible
10. Planifier l'implémentation Sprint B
11. **Audit Complet Versions Dépendances**
12. Aligner toutes les versions avec le tableau 3.4
13. Identifier les migrations nécessaires
14. Créer le plan de mise à jour coordonnée

15. Mettre à jour la documentation

Priorité P1 - Qualité et Stabilité

1. **Lancer Test Debt Sprint Planning**
2. Auditer la couverture actuelle (60%)
3. Identifier les zones critiques non testées
4. Prioriser les tests à implémenter
5. Planifier le sprint dédié avant v0.3.0
6. **Standardiser Headers de Fichiers**
7. Appliquer les nouveaux formats créatifs
8. Créer les templates pour l'équipe
9. Mettre à jour les outils de génération
10. Former l'équipe aux nouvelles conventions

15.2 Actions Stratégiques Semaine 3-4

Développement Extension Chrome MVP 1

1. **Initialiser Extension Chrome**
2. Setup projet Manifest v3
3. Implémenter détection champs texte générique
4. Créer l'interface de suggestion basique
5. Tester sur sites populaires marocains
6. **Intégration API SalamBot**
7. Connecter l'extension aux flows existants
8. Implémenter fallback offline
9. Optimiser les performances (<500ms)
10. Ajouter l'instrumentation de base

Préparation Phase 3

1. **Planification Détaillée Phase 3**
2. Définir les sprints et jalons
3. Allouer les ressources par squad
4. Créer les tickets GitHub détaillés
5. Valider les dépendances techniques
6. **Communication et Alignment**

7. Présenter la roadmap révisée aux stakeholders
8. Aligner les équipes sur les nouvelles priorités
9. Mettre à jour les communications externes
10. Préparer les démos clients

15.3 Checklist de Validation

✅ **Critères de Succès Semaine 1-2:** - ☐ Agent Desk 100% migré vers Next.js 15 - ☐ PoC API Gateway complété avec recommandation - ☐ Versions dépendances alignées avec documentation - ☐ Plan Test Debt Sprint validé et planifié - ☐ Headers de fichiers standardisés sur 50% du code

✅ **Critères de Succès Semaine 3-4:** - ☐ Extension Chrome MVP 1 fonctionnelle en beta - ☐ API Gateway Sprint B planifié et équipe assignée - ☐ Phase 3 détaillée avec tickets GitHub créés - ☐ Stakeholders alignés sur roadmap révisée - ☐ Métriques de succès définies et tracking activé

Conclusion v2.1

Cette version 2.1 du cahier des charges SalamBot corrige les incohérences critiques identifiées dans l'audit du 2 juin 2025 et établit une roadmap réaliste et exécutable pour les prochaines phases de développement.

Les corrections apportées alignent parfaitement la documentation avec la réalité technique du projet, éliminent les ambiguïtés sur les versions et les priorités, et introduisent une approche MVP progressive qui réduit les risques tout en maximisant la valeur livrée.

L'accent mis sur la qualité technique, la stabilité architecturale, et l'approche pragmatique de développement positionne SalamBot pour un succès durable sur le marché marocain de l'IA conversationnelle.



Les prochaines 4 semaines seront déterminantes pour corriger les lacunes architecturales et établir les fondations solides nécessaires à la croissance future de SalamBot.





Document validé par :

SalamBot Team


Version 2.1 - 2 juin 2025


Contact : info@salambot.ma


Révisions majeures v2.1 : -  Versions techniques alignées avec le dépôt réel -  API Gateway roadmap critique ajoutée

-  Extension Chrome approche MVP révisée -  Standards de commentaires créatifs et modernes -  Plan d'action immédiat 4 semaines détaillé -  Métriques de succès réalistes et mesurables

MISE À JOUR CRITIQUE v2.2 - Intégration Recommandations


 **Date de mise à jour:** 2 juin 2025

 **Version:** 2.2 (Corrections critiques intégrées)

 **Statut:** Document révisé suite à audit technique approfondi

Section 4.6 - Détection Darija Bi-Script (CRITIQUE)

4.6.1 Problématique Identifiée

 **ALERTE CRITIQUE:** L'audit technique révèle que l'implémentation actuelle de détection Darija n'atteint que **45% de précision** contre les **88% promis** dans la documentation officielle. Cette lacune représente un **risque existentiel** pour la proposition de valeur unique de SalamBot et compromet directement notre avantage concurrentiel principal.

L'impact de cette défaillance technique se manifeste à plusieurs niveaux critiques. Au niveau business, nous faisons face à un risque de réputation majeur avec des promesses non tenues sur notre expertise Darija, créant une frustration utilisateur due aux erreurs de détection fréquentes et la perte potentielle de notre différenciateur principal face à la concurrence. Au niveau technique, l'architecture actuelle ne gère pas efficacement la dualité script arabe/latin du Darija, les seuils de confiance sont mal calibrés pour les spécificités dialectales, et l'intégration avec les modèles LLM spécialisés présente des lacunes significatives.

Cette situation nécessite une refonte complète de l'approche de détection linguistique avec une priorité absolue sur l'amélioration des performances Darija. L'objectif révisé est d'atteindre **90%+ de précision** pour la détection Darija bi-script, dépassant ainsi nos promesses initiales et renforçant notre position concurrentielle.

4.6.2 Architecture Darija Bi-Script Révisée

La nouvelle architecture de détection Darija implémente une approche multi-couches sophistiquée qui traite efficacement la complexité linguistique du dialecte marocain dans ses deux formes scripturales principales.

```
/**
 * 🧠 SALAMBOT DARIJA DETECTION ENGINE v2.2
 *
 * 🎯 Mission: Détection Darija bi-script avec précision >90%
 * 📊 Performance: <500ms latency | Fallback intelligent
 * 🔬 Innovation: Pipeline multi-couches spécialisé Maroc
 *
 * @architecture Multi-layer detection pipeline
 * @precision Target >90% (vs 45% current)
 * @latency <500ms average response time
 * @fallback Intelligent degradation strategy
 */

interface DarijaDetectionPipeline {
    // Couche 1: Détection de script Unicode
    scriptDetection: (text: string) => ScriptAnalysis;

    // Couche 2: Analyse lexicale spécialisée
    lexicalAnalysis: (text: string, script: ScriptType) =>
    LexicalResult;

    // Couche 3: Fallback LLM contextuel
    llmFallback: (text: string, context: DetectionContext) =>
    LLMResult;

    // Couche 4: Agrégation de confiance pondérée
    confidenceAggregation: (results: DetectionResult[]) =>
    FinalResult;

    // Couche 5: Validation et métriques
    resultValidation: (result: FinalResult) => ValidatedResult;
}

interface ScriptAnalysis {
    primaryScript: 'arabic' | 'latin' | 'mixed';
    scriptConfidence: number;
    unicodeRanges: UnicodeRange[];
    mixedScriptPatterns: MixedPattern[];
}

interface LexicalResult {
    darijaMarkers: DarijaMarker[];
    dialectalPatterns: DialectalPattern[];
    regionalVariants: RegionalVariant[];
}
```



```

    confidence: number;
    fallbackRequired: boolean;
}

interface DarijaMarker {
    term: string;
    script: 'arabic' | 'latin';
    confidence: number;
    regionalOrigin: MoroccanRegion;
    frequency: number;
}

```

4.6.3 Dictionnaire Darija Bi-Script Étendu

Le cœur de la nouvelle architecture repose sur un dictionnaire Darija bi-script exhaustif comprenant plus de 2000 termes spécialisés couvrant les expressions courantes, les variations régionales, et les néologismes contemporains du dialecte marocain.

Structure du Dictionnaire Darija:

```

/**
 * 📖 DICTIONNAIRE DARIJA BI-SCRIPT v2.2
 *
 * 🎯 Couverture: 2000+ termes spécialisés Maroc
 * 📍 Régions: Casa, Rabat, Marrakech, Fès, Tanger
 * 📖 Scripts: Arabe + Latin (Arabizi) + Mixte
 */

interface DarijaDictionary {
    // Expressions courantes (800+ termes)
    commonExpressions: BiScriptEntry[];

    // Variations régionales (400+ termes)
    regionalVariants: RegionalEntry[];

    // Néologismes contemporains (300+ termes)
    modernTerms: ModernEntry[];

    // Expressions métier/business (200+ termes)
    businessTerms: BusinessEntry[];

    // Connecteurs et particules (300+ termes)
    linguisticParticles: ParticleEntry[];
}

interface BiScriptEntry {
    arabic: string;           // "كيفاش"
    latin: string;            // "kifash"
    alternatives: string[];   // ["kif", "kifach", "كيف"]
}

```

```

meaning: string;           // "comment"
region: MoroccanRegion;    // "national" | "casablanca" | etc.
frequency: number;         // 0.0-1.0 usage frequency
context: UsageContext[];   // ["informal", "business",
"family"]
}

// Exemples d'entrées critiques
const criticalDarijaTerms: BiScriptEntry[] = [
  {
    arabic: "واخا",
    latin: "wakha",
    alternatives: ["wkha", "واخة"],
    meaning: "d'accord/ok",
    region: "national",
    frequency: 0.95,
    context: ["informal", "business"]
  },
  {
    arabic: "بزاف",
    latin: "bzaf",
    alternatives: ["bzzaf", "bzef", "بزف"],
    meaning: "beaucoup",
    region: "national",
    frequency: 0.90,
    context: ["informal", "family", "business"]
  },
  {
    arabic: "دابا",
    latin: "daba",
    alternatives: ["dba", "دبا"],
    meaning: "maintenant",
    region: "national",
    frequency: 0.85,
    context: ["informal", "urgent"]
  }
];

```

4.6.4 Algorithme de Détection Multi-Couches

L'algorithme de détection implémente une approche en cascade qui maximise la précision tout en maintenant des performances optimales.

Étape 1: Détection de Script Unicode La première couche analyse la composition Unicode du texte pour identifier le script principal et détecter les patterns de script mixte caractéristiques du Darija moderne.

```

/**
 *  SCRIPT DETECTION LAYER

```

```

*
* 🎯 Objectif: Identifier script principal et patterns mixtes
* ⚡ Performance: <50ms pour textes <1000 caractères
* 🎨 Innovation: Détection patterns Arabizi sophistiqués
*/

```

```

class ScriptDetector {
  detectScript(text: string): ScriptAnalysis {
    const unicodeAnalysis = this.analyzeUnicodeRanges(text);
    const mixedPatterns = this.detectMixedScriptPatterns(text);

    return {
      primaryScript:
this.determinePrimaryScript(unicodeAnalysis),
      scriptConfidence:
this.calculateScriptConfidence(unicodeAnalysis),
      unicodeRanges: unicodeAnalysis.ranges,
      mixedScriptPatterns: mixedPatterns
    };
  }

  private analyzeUnicodeRanges(text: string): UnicodeAnalysis {
    const ranges = {
      arabic: this.countInRange(text, 0x0600, 0x06FF),      //
Arabic
      latin: this.countInRange(text, 0x0020, 0x007F),      //
Basic Latin
      arabicSupplement: this.countInRange(text, 0x0750,
0x077F), // Arabic Supplement
      numbers: this.countInRange(text, 0x0030, 0x0039)      //
Numbers
    };

    return {
      ranges,
      totalChars: text.length,
      dominantScript: this.findDominantScript(ranges)
    };
  }

  private detectMixedScriptPatterns(text: string):
MixedPattern[] {
    // Patterns typiques Darija mixte
    const patterns = [
      /[a-zA-Z]+[ا-ز]+[a-zA-Z]+/g,    // Latin-Arabe-Latin
      /[ا-ز]+[a-zA-Z]+[ا-ز]+/g,      // Arabe-Latin-Arabe
      /\b[a-zA-Z]+[0-9]+[ا-ز]+/g,     // Latin-Chiffre-Arabe
    ];

    return patterns.map(pattern => ({
      pattern: pattern.source,
      matches: Array.from(text.matchAll(pattern)),

```

```

        confidence: this.calculatePatternConfidence(pattern, text)
    }));
}
}

```

Étape 2: Analyse Lexicale Spécialisée La deuxième couche effectue une analyse lexicale approfondie en utilisant le dictionnaire bi-script pour identifier les marqueurs Darija spécifiques.

```

/**
 * 📖 LEXICAL ANALYSIS LAYER
 *
 * 🎯 Objectif: Identifier marqueurs Darija dans dictionnaire
bi-script
 * 📚 Données: 2000+ termes spécialisés Maroc
 * 🎨 Innovation: Analyse contextuelle et régionale
 */

class LexicalAnalyzer {
    analyzeDarijaMarkers(text: string, script: ScriptType):
LexicalResult {
        const tokens = this.tokenizeText(text, script);
        const darijaMarkers = this.findDarijaMarkers(tokens);
        const dialectalPatterns =
this.identifyDialectalPatterns(tokens);
        const regionalVariants =
this.detectRegionalVariants(darijaMarkers);

        const confidence = this.calculateLexicalConfidence(
            darijaMarkers,
            dialectalPatterns,
            regionalVariants
        );

        return {
            darijaMarkers,
            dialectalPatterns,
            regionalVariants,
            confidence,
            fallbackRequired: confidence < 0.75
        };
    }

    private findDarijaMarkers(tokens: Token[]): DarijaMarker[] {
        return tokens
            .map(token => this.matchDarijaEntry(token))
            .filter(match => match !== null)
            .map(match => ({
                term: match.term,
                script: match.script,
            }));
    }
}

```

```

        confidence: match.confidence,
        regionalOrigin: match.region,
        frequency: match.frequency
    }));
}

private identifyDialectalPatterns(tokens: Token[]):
DialectalPattern[] {
    // Patterns grammaticaux spécifiques au Darija
    const patterns = [
        // Négation Darija: "ما...ش"
        { pattern: /ما\s+\w+\s+ش/, type: 'negation_darija' },
        // Futur Darija: "غا + verbe"
        { pattern: /غا\s+\w+/, type: 'future_darija' },
        // Question Darija: "واش"
        { pattern: /واش\s+/, type: 'question_darija' }
    ];

    return patterns
        .map(p => this.findPatternMatches(tokens.join(' '), p))
        .filter(matches => matches.length > 0)
        .flat();
}
}

```

Étape 3: Fallback LLM Contextuel Lorsque l'analyse lexicale ne fournit pas une confiance suffisante, le système fait appel à un modèle LLM spécialisé pour une analyse contextuelle approfondie.

```

/**
 * 🤖 LLM FALLBACK LAYER
 *
 * 🎯 Objectif: Analyse contextuelle via LLM spécialisé Darija
 * 🧠 Modèle: Llama 3.1-70B fine-tuned Darija
 * 🍷 Innovation: Prompts optimisés détection dialectale
 */

class LLMFallbackAnalyzer {
    async analyzeDarijaContext(
        text: string,
        context: DetectionContext
    ): Promise<LLMResult> {
        const prompt = this.buildDarijaDetectionPrompt(text,
context);

        const llmResponse = await this.callLlamaModel({
            prompt,
            temperature: 0.1, // Précision maximale
            maxTokens: 200,
            model: 'llama-3.1-70b-darija-specialized'

```

```

    });

    return this.parseLLMResponse(llmResponse);
}

private buildDarijaDetectionPrompt(
    text: string,
    context: DetectionContext
): string {
    return `
Analyse linguistique spécialisée - Dialecte Darija Marocain

TEXTE À ANALYSER: "${text}"

CONTEXTE:
- Script détecté: ${context.detectedScript}
- Marqueurs lexicaux: ${context.lexicalMarkers.length}
- Confiance lexicale: ${context.lexicalConfidence}

MISSION:
Détermine si ce texte contient du Darija marocain (dialecte).

CRITÈRES DARIJA:
1. Vocabulaire spécifique: واخا, بزاف, دابا, كيفاش
2. Structures grammaticales: négation "ما...ش", futur "غا"
3. Arabizi: wakha, bzaf, daba, kifash
4. Expressions régionales marocaines

RÉPONSE REQUISE (JSON):
{
  "isDarija": boolean,
  "confidence": number (0-1),
  "evidence": string[],
  "script": "arabic" | "latin" | "mixed",
  "region": "casablanca" | "rabat" | "marrakech" | "national"
}

ANALYSE:`;
}

```

4.6.5 Métriques et Validation de Performance

Le système de détection Darija bi-script implémente un système de métriques complet pour assurer une amélioration continue des performances.

```

/**
 *  DARIJA DETECTION METRICS v2.2
 *

```

- * 🎯 Objectifs: >90% précision | <500ms latence
 - * 📈 Métriques: Précision, Rappel, F1-Score par script
 - * 🔍 Monitoring: Temps réel + alertes automatiques
- */

```
interface DarijaDetectionMetrics {
  // Métriques de précision
  accuracy: {
    overall: number;           // >90% objectif
    arabicScript: number;      // >95% objectif
    latinScript: number;       // >85% objectif (plus difficile)
    mixedScript: number;       // >80% objectif
  };

  // Métriques de performance
  performance: {
    averageLatency: number;     // <500ms objectif
    p95Latency: number;        // <800ms objectif
    p99Latency: number;        // <1200ms objectif
    throughput: number;        // req/sec
  };

  // Métriques de fallback
  fallback: {
    lexicalFallbackRate: number; // % utilisation LLM
    llmFallbackLatency: number;  // Latence supplémentaire LLM
    fallbackAccuracy: number;    // Précision avec fallback
  };

  // Métriques régionales
  regional: {
    casablanca: RegionalMetrics;
    rabat: RegionalMetrics;
    marrakech: RegionalMetrics;
    fes: RegionalMetrics;
    tanger: RegionalMetrics;
  };
}

interface RegionalMetrics {
  accuracy: number;
  commonTermsDetected: number;
  regionalVariantsDetected: number;
  userSatisfaction: number;
}
```

4.6.6 Plan d'Implémentation Darija Bi-Script

Phase 1: Développement Core (6-8 semaines)

```
gantt
  title Implémentation Détection Darija Bi-Script
  dateFormat YYYY-MM-DD
  section Phase 1 - Core Development
  Dictionnaire Bi-Script      :dict, 2025-06-03, 2w
  Script Detection Engine     :script, after dict, 1w
  Lexical Analysis Engine     :lexical, after script, 2w
  LLM Fallback Integration    :llm, after lexical, 2w
  Testing & Validation        :test, after llm, 1w
  section Phase 2 - Optimization
  Performance Tuning          :perf, after test, 2w
  Regional Variants           :regional, after perf, 1w
  Metrics & Monitoring        :metrics, after regional, 1w
  section Phase 3 - Integration
  API Integration             :api, after metrics, 1w
  Extension Chrome            :chrome, after api, 1w
  Production Deployment       :prod, after chrome, 1w
```

Jalons Critiques: - **Semaine 2:** Dictionnaire bi-script 2000+ termes validé - **Semaine 4:** Pipeline de détection fonctionnel - **Semaine 6:** Précision >85% atteinte sur dataset test - **Semaine 8:** Précision >90% validée en production




Section 4.7 - Stratégies de Fallback IA Multi-Provider

4.7.1 Problématique Vendor Lock-in

L'architecture actuelle de SalamBot présente une dépendance critique à Google Cloud Platform et au modèle Llama 3.1-70B sans stratégie de fallback documentée. Cette situation expose le projet à des risques opérationnels majeurs incluant l'indisponibilité de service, les changements de pricing, les limitations de quota, et les restrictions géopolitiques potentielles.

La stratégie de fallback multi-provider élimine ces risques en implémentant une architecture résiliente qui assure la continuité de service même en cas de défaillance du provider principal.

4.7.2 Architecture Multi-Provider Résiliente

```
/**
 *  SALAMBOT AI FALLBACK STRATEGY v2.2
 *
 *  Mission: Éliminer vendor lock-in + assurer continuité
service
 *  Résilience: 99.9% uptime même avec provider principal down
```



```
* ⚡ Performance: Fallback transparent <2s
*/
```

```
interface AIProviderStrategy {
  primary: {
    provider: 'vertex-ai';
    model: 'llama-3.1-70b-darija-specialized';
    priority: 1;
    healthCheck: HealthCheckConfig;
    rateLimits: RateLimitConfig;
  };

  fallback1: {
    provider: 'openai';
    model: 'gpt-4-turbo';
    priority: 2;
    healthCheck: HealthCheckConfig;
    rateLimits: RateLimitConfig;
  };

  fallback2: {
    provider: 'anthropic';
    model: 'claude-3-opus';
    priority: 3;
    healthCheck: HealthCheckConfig;
    rateLimits: RateLimitConfig;
  };

  emergency: {
    provider: 'local';
    model: 'llama-2-7b-quantized';
    priority: 4;
    deployment: 'on-premise';
  };
}

class AIProviderOrchestrator {
  async generateResponse(
    request: AIRequest,
    context: ConversationContext
  ): Promise<AIResponse> {
    const providers = this.getAvailableProviders();

    for (const provider of providers) {
      try {
        const response = await this.callProvider(provider,
request, context);

        // Log successful provider usage
        this.logProviderUsage(provider, 'success',
response.latency);
      }
    }
  }
}
```

```

        return response;
    } catch (error) {
        // Log provider failure and try next
        this.logProviderUsage(provider, 'failure', error);

        if (this.isLastProvider(provider)) {
            throw new AIServiceUnavailableError('All AI providers
failed');
        }

        continue; // Try next provider
    }
}

private async callProvider(
    provider: AIProvider,
    request: AIRequest,
    context: ConversationContext
): Promise<AIResponse> {
    // Adapter le prompt selon le provider
    const adaptedPrompt =
this.adaptPromptForProvider(request.prompt, provider);

    // Appel spécialisé selon le provider
    switch (provider.name) {
        case 'vertex-ai':
            return await this.callVertexAI(adaptedPrompt, context);
        case 'openai':
            return await this.callOpenAI(adaptedPrompt, context);
        case 'anthropic':
            return await this.callAnthropic(adaptedPrompt, context);
        case 'local':
            return await this.callLocalModel(adaptedPrompt,
context);
        default:
            throw new UnsupportedProviderError(provider.name);
    }
}
}

```

4.7.3 Configuration Cache Redis Multi-Niveau

```

/**
 * 🗄️ REDIS CACHE STRATEGY v2.2
 *
 * 🎯 Mission: Cache intelligent multi-niveau pour résilience
 * ⚡ Performance: <50ms cache hit | 99% hit rate objectif
 * 🛡️ Fallback: In-memory cache si Redis indisponible
 */

```

```

interface CacheStrategy {
  primary: {
    type: 'redis-cluster';
    nodes: RedisNode[];
    ttl: CacheTTLConfig;
    eviction: 'allkeys-lru';
  };

  fallback: {
    type: 'in-memory-cache';
    maxSize: '512MB';
    ttl: CacheTTLConfig;
    persistence: false;
  };

  emergency: {
    type: 'local-storage';
    path: '/tmp/salambot-cache';
    maxSize: '1GB';
    compression: true;
  };
}

class CacheOrchestrator {
  async get(key: string): Promise<CacheResult | null> {
    // Essai cache Redis principal
    try {
      const result = await this.redisCluster.get(key);
      if (result) {
        this.logCacheHit('redis', key);
        return JSON.parse(result);
      }
    } catch (error) {
      this.logCacheError('redis', error);
    }

    // Fallback cache mémoire
    const memoryResult = this.memoryCache.get(key);
    if (memoryResult) {
      this.logCacheHit('memory', key);
      return memoryResult;
    }

    // Fallback cache local
    try {
      const localResult = await this.localCache.get(key);
      if (localResult) {
        this.logCacheHit('local', key);
        return localResult;
      }
    } catch (error) {

```

```
        this.logCacheError('local', error);
    }

    this.logCacheMiss(key);
    return null;
}

async set(key: string, value: any, ttl?: number):
Promise<void> {
    const serializedValue = JSON.stringify(value);

    // Écriture parallèle dans tous les caches
    const promises = [
        this.redisCluster.setex(key, ttl || 3600, serializedValue)
            .catch(error => this.logCacheError('redis-write',
error)),

        this.memoryCache.set(key, value, ttl)
            .catch(error => this.logCacheError('memory-write',
error)),

        this.localCache.set(key, value, ttl)
            .catch(error => this.logCacheError('local-write',
error))
    ];

    // Attendre au moins une écriture réussie
    await Promise.allSettled(promises);
}

// Configuration cache spécialisée
const cacheConfig = {
    // Patterns de détection langue (24h TTL)
    'darija:patterns:': { ttl: 86400, priority: 'high' },

    // Réponses fréquentes (1h TTL)
    'responses:frequent:': { ttl: 3600, priority: 'medium' },

    // Contexte utilisateur (30min TTL)
    'user:context:': { ttl: 1800, priority: 'low' },

    // Métriques temps réel (5min TTL)
    'metrics:realtime:': { ttl: 300, priority: 'critical' }
};
```



Section 4.8 - Métriques et KPIs Détaillés

4.8.1 Métriques Techniques Critiques

```
/**
 *  SALAMBOT TECHNICAL METRICS v2.2
 *
 *  Mission: Monitoring temps réel + alertes automatiques
 *  Coverage: Performance | Qualité | Disponibilité | Sécurité
 *  Alerting: Seuils critiques + escalation automatique
 */

interface TechnicalMetrics {
    // Métriques de performance IA
    aiPerformance: {
        // Détection de langue
        languageDetection: {
            averageLatency: Metric<number>; // <500ms objectif
            p95Latency: Metric<number>; // <800ms objectif
            p99Latency: Metric<number>; // <1200ms objectif
            accuracy: {
                overall: Metric<number>; // >96% objectif
                darija: Metric<number>; // >90% objectif
                (critique)
                french: Metric<number>; // >98% objectif
                arabic: Metric<number>; // >95% objectif
            };
            fallbackRate: Metric<number>; // <15% objectif
        };

        // Génération de réponses
        responseGeneration: {
            averageLatency: Metric<number>; // <2000ms objectif
            p95Latency: Metric<number>; // <3000ms objectif
            p99Latency: Metric<number>; // <5000ms objectif
            providerDistribution: {
                vertexAI: Metric<number>; // % utilisation
                openAI: Metric<number>; // % fallback
                anthropic: Metric<number>; // % fallback
                local: Metric<number>; // % emergency
            };
            qualityScore: Metric<number>; // >4.0/5.0 objectif
        };
    };

    // Métriques d'infrastructure
    infrastructure: {
        // Disponibilité système
        availability: {
            uptime: Metric<number>; // >99.9% objectif
        };
    };
};
```

```

    apiGateway: Metric<number>;           // >99.95% objectif
    redis: Metric<number>;                // >99.5% objectif
    database: Metric<number>;             // >99.8% objectif
};

// Performance système
performance: {
    cpuUtilization: Metric<number>;       // <70% moyenne
    memoryUtilization: Metric<number>;    // <80% moyenne
    diskUtilization: Metric<number>;      // <85% moyenne
    networkLatency: Metric<number>;       // <100ms moyenne
};

// Capacité et scaling
capacity: {
    requestsPerSecond: Metric<number>;    // Capacité actuelle
    concurrentUsers: Metric<number>;       // Utilisateurs
simultanés
    queueDepth: Metric<number>;           // <100 messages
objectif
    scalingEvents: Metric<number>;        // Événements auto-
scaling
};
};

// Métriques de sécurité
security: {
    authentication: {
        successRate: Metric<number>;      // >99% objectif
        failureRate: Metric<number>;      // <1% objectif
        suspiciousAttempts: Metric<number>; // Alertes sécurité
    };

    rateLimiting: {
        blockedRequests: Metric<number>;  // Requêtes bloquées
        rateLimitHits: Metric<number>;    // Limites atteintes
        abuseDetection: Metric<number>;    // Détection d'abus
    };
};

}

interface Metric<T> {
    current: T;
    target: T;
    threshold: {
        warning: T;
        critical: T;
    };
    trend: 'improving' | 'stable' | 'degrading';
    lastUpdated: Date;
}

```

4.8.2 Métriques Business et Adoption

```
/**
 * 🧳 SALAMBOT BUSINESS METRICS v2.2
 *
 * 🎯 Mission: Tracking adoption + satisfaction + revenue
 * 📊 Coverage: Utilisateurs | Engagement | Revenue |
Satisfaction
 * 📈 Growth: Métriques croissance + rétention
 */

interface BusinessMetrics {
    // Adoption et croissance
    adoption: {
        totalUsers: Metric<number>; // Utilisateurs totaux
        activeUsers: {
            daily: Metric<number>; // DAU
            weekly: Metric<number>; // WAU
            monthly: Metric<number>; // MAU
        };
        newUsers: {
            daily: Metric<number>; // Nouveaux/jour
            weekly: Metric<number>; // Nouveaux/semaine
            monthly: Metric<number>; // Nouveaux/mois
        };
        churnRate: {
            daily: Metric<number>; // <2% objectif
            weekly: Metric<number>; // <5% objectif
            monthly: Metric<number>; // <10% objectif
        };
    };

    // Engagement utilisateur
    engagement: {
        conversationsPerUser: Metric<number>; // Conversations/
utilisateur
        messagesPerConversation: Metric<number>; // Messages/
conversation
        sessionDuration: Metric<number>; // Durée session
moyenne
        returnRate: Metric<number>; // Taux de retour 7j
        featureUsage: {
            languageDetection: Metric<number>; // % utilisation
            chromeExtension: Metric<number>; // % adoption
extension
            agentEscalation: Metric<number>; // % escalation
humaine
        };
    };

    // Satisfaction client
```

```

satisfaction: {
  npsScore: Metric<number>; // >50 objectif
  csat: Metric<number>; // >4.0/5.0 objectif
  darijaSpecific: {
    accuracy: Metric<number>; // >90% objectif
    satisfaction: Metric<number>; // >4.5/5.0 objectif
    usage: Metric<number>; // % conversations
Darija
  };
  supportTickets: {
    volume: Metric<number>; // Tickets/semaine
    resolution: Metric<number>; // Temps résolution
    satisfaction: Metric<number>; // Satisfaction
support
  };
};

// Métriques revenue
revenue: {
  mrr: Metric<number>; // Monthly Recurring
Revenue
  arr: Metric<number>; // Annual Recurring
Revenue
  ltv: Metric<number>; // Lifetime Value
  cac: Metric<number>; // Customer
Acquisition Cost
  planDistribution: {
    free: Metric<number>; // % utilisateurs
gratuits
  };
  starter: Metric<number>; // % plan Starter
  business: Metric<number>; // % plan Business
  enterprise: Metric<number>; // % plan Enterprise
};
};
}

```

4.8.3 Dashboard et Alertes Temps Réel

```

/**
 * 🚨 SALAMBOT ALERTING SYSTEM v2.2
 *
 * 🎯 Mission: Détection proactive + escalation automatique
 * ⚡ Réactivité: <30s détection + notification immédiate
 * 🎯 Précision: Réduction faux positifs + alertes contextuelles
 */

interface AlertingConfig {
  // Alertes critiques (P0)
  critical: {
    // Détection Darija en panne

```



```
darijaDetectionFailure: {
  condition: 'darija_accuracy < 0.70 for 5min';
  notification: ['slack', 'email', 'sms'];
  escalation: 'immediate';
  oncall: 'ai-team';
};

// API Gateway down
apiGatewayDown: {
  condition: 'api_availability < 0.95 for 2min';
  notification: ['slack', 'email', 'sms', 'pagerduty'];
  escalation: 'immediate';
  oncall: 'platform-team';
};

// Tous les providers IA en échec
allAIProvidersDown: {
  condition: 'ai_provider_success_rate < 0.10 for 1min';
  notification: ['slack', 'email', 'sms', 'pagerduty'];
  escalation: 'immediate';
  oncall: 'ai-team';
};

// Alertes importantes (P1)
warning: {
  // Performance dégradée
  performanceDegradation: {
    condition: 'response_latency_p95 > 3000ms for 10min';
    notification: ['slack', 'email'];
    escalation: '15min';
    oncall: 'platform-team';
  };
};

// Taux d'erreur élevé
highErrorRate: {
  condition: 'error_rate > 0.05 for 5min';
  notification: ['slack', 'email'];
  escalation: '10min';
  oncall: 'dev-team';
};

// Cache Redis dégradé
redisDegradation: {
  condition: 'redis_hit_rate < 0.80 for 15min';
  notification: ['slack'];
  escalation: '30min';
  oncall: 'platform-team';
};

// Alertes informatives (P2)
```

```

info: {
  // Utilisation élevée
  highUsage: {
    condition: 'requests_per_second > 800 for 30min';
    notification: ['slack'];
    escalation: 'none';
    oncall: null;
  };

  // Nouveau pic d'adoption
  adoptionSpike: {
    condition: 'new_users_daily > 200';
    notification: ['slack'];
    escalation: 'none';
    oncall: null;
  };
};

}

class MetricsCollector {
  async collectMetrics(): Promise<MetricsSnapshot> {
    const [technical, business] = await Promise.all([
      this.collectTechnicalMetrics(),
      this.collectBusinessMetrics()
    ]);

    const snapshot = {
      timestamp: new Date(),
      technical,
      business,
      alerts: await this.evaluateAlerts(technical, business)
    };

    // Stockage pour analyse historique
    await this.storeMetricsSnapshot(snapshot);

    // Envoi vers systèmes de monitoring
    await Promise.all([
      this.sendToGrafana(snapshot),
      this.sendToDatadog(snapshot),
      this.updateHealthChecks(snapshot)
    ]);

    return snapshot;
  }

  private async evaluateAlerts(
    technical: TechnicalMetrics,
    business: BusinessMetrics
  ): Promise<Alert[]> {
    const alerts: Alert[] = [];

```

```

    // Évaluation alertes critiques
    if
    (technical.aiPerformance.languageDetection.accuracy.darija.current
    < 0.70) {
        alerts.push({
            level: 'critical',
            type: 'darijaDetectionFailure',
            message: `Détection Darija dégradée: $
{technical.aiPerformance.languageDetection.accuracy.darija.current
* 100}%`,
            timestamp: new Date(),
            oncall: 'ai-team'
        });
    }

    if (technical.infrastructure.availability.uptime.current <
    0.995) {
        alerts.push({
            level: 'critical',
            type: 'apiGatewayDown',
            message: `Disponibilité API Gateway: $
{technical.infrastructure.availability.uptime.current * 100}%`,
            timestamp: new Date(),
            oncall: 'platform-team'
        });
    }

    return alerts;
}
}

```

Section 7.4 - Architecture de Sécurité Renforcée

7.4.1 Configuration Sécurité Multicouche

L'architecture de sécurité SalamBot implémente une approche défense en profondeur avec des contrôles de sécurité à chaque niveau de l'infrastructure et de l'application.

```

/**
 *  SALAMBOT SECURITY ARCHITECTURE v2.2
 *
 *  Mission: Sécurité multicouche + conformité RGPD/Loi 09-08
 *  Approche: Defense in depth + Zero trust
 *  Monitoring: Détection temps réel + réponse automatique
 */

interface SecurityConfiguration {
    // Couche 1: Sécurité réseau et infrastructure

```

```

networkSecurity: {
  firewall: {
    inbound: FirewallRule[];
    outbound: FirewallRule[];
    ddosProtection: {
      enabled: true;
      threshold: 10000; // req/min
      blockDuration: 3600; // seconds
    };
  };

  tls: {
    version: 'TLS-1.3';
    cipherSuites: string[];
    certificateRotation: {
      interval: '90d';
      autoRenewal: true;
      provider: 'lets-encrypt';
    };
  };

  vpn: {
    adminAccess: 'wireguard';
    allowedIPs: string[];
    mfaRequired: true;
  };
};

// Couche 2: Sécurité application
applicationSecurity: {
  authentication: {
    oauth2: OAuth2Config;
    jwt: {
      algorithm: 'RS256';
      expiration: 3600; // 1 hour
      refreshExpiration: 604800; // 7 days
      secretRotation: '30d';
    };
    mfa: {
      required: boolean;
      methods: ['totp', 'sms', 'email'];
      backupCodes: true;
    };
  };

  authorization: {
    rbac: RBACConfig;
    abac: ABACConfig;
    permissions: PermissionMatrix;
  };

  inputValidation: {

```

```

sanitization: {
  enabled: true;
  xssProtection: true;
  sqlInjectionPrevention: true;
  htmlSanitizer: 'dompurify';
};

rateLimiting: {
  perIP: {
    requests: 100;
    window: '1m';
    blockDuration: '15m';
  };
  perUser: {
    requests: 500;
    window: '1m';
    blockDuration: '5m';
  };
  perEndpoint: Record<string, RateLimitRule>;
};
};
};

```

// Couche 3: Sécurité données

```

dataSecurity: {
  encryption: {
    atRest: {
      algorithm: 'AES-256-GCM';
      keyRotation: '90d';
      keyManagement: 'google-secret-manager';
    };
    inTransit: {
      algorithm: 'TLS-1.3';
      certificatePinning: true;
      hsts: {
        enabled: true;
        maxAge: 31536000; // 1 year
        includeSubdomains: true;
      };
    };
  };
};

```

```

dataClassification: {
  public: DataClassificationRule;
  internal: DataClassificationRule;
  confidential: DataClassificationRule;
  restricted: DataClassificationRule;
};

```

```

dataRetention: {
  conversationData: '2y';
  userProfiles: '5y';
}

```

```

        auditLogs: '7y';
        metrics: '1y';
    };
};

// Couche 4: Monitoring et réponse
securityMonitoring: {
    siem: {
        provider: 'google-cloud-security-center';
        alerting: AlertingConfig;
        correlation: CorrelationRules;
    };

    vulnerabilityScanning: {
        static: 'sonarqube';
        dynamic: 'owasp-zap';
        dependency: 'dependabot';
        container: 'trivy';
        frequency: 'daily';
    };

    incidentResponse: {
        playbooks: IncidentPlaybook[];
        escalation: EscalationMatrix;
        communication: CommunicationPlan;
    };
};
}

```

```

// Configuration spécialisée pour l'IA
interface AISecurityConfig {
    // Sécurité des modèles IA
    modelSecurity: {
        inputSanitization: {
            promptInjectionPrevention: true;
            maxInputLength: 4000; // caractères
            forbiddenPatterns: RegExp[];
        };

        outputValidation: {
            contentFiltering: true;
            biasDetection: true;
            toxicityScoring: true;
            maxOutputLength: 2000;
        };

        modelAccess: {
            authentication: 'service-account';
            authorization: 'iam-roles';
            auditLogging: true;
        };
    };
};

```

```
// Protection données conversationnelles
conversationSecurity: {
  piiDetection: {
    enabled: true;
    patterns: PIIPattern[];
    redactionStrategy: 'mask';
  };

  dataMinimization: {
    retention: '30d';
    anonymization: 'after-7d';
    purgeStrategy: 'automatic';
  };

  consentManagement: {
    granular: true;
    withdrawal: 'immediate';
    portability: 'gdpr-compliant';
  };
};
}
```

7.4.2 Conformité RGPD et Loi 09-08 Marocaine

```
/**
 * ⚖️ SALAMBOT COMPLIANCE FRAMEWORK v2.2
 *
 * 🎯 Mission: Conformité native RGPD + Loi 09-08 Maroc
 * 📋 Approche: Privacy by design + Data minimization
 * 🔍 Audit: Traçabilité complète + rapports automatiques
 */

interface ComplianceFramework {
  // Conformité RGPD (Règlement Général sur la Protection des
  // Données)
  gdprCompliance: {
    legalBasis: {
      consent: ConsentManagement;
      legitimateInterest: LegitimateInterestAssessment;
      contractualNecessity: ContractualBasis;
    };

    dataSubjectRights: {
      access: {
        responseTime: '30d';
        format: ['json', 'pdf', 'csv'];
        automation: 'partial';
      };
      rectification: {
```

```

        responseTime: '30d';
        verification: 'required';
        notification: 'automatic';
    };
    erasure: {
        responseTime: '30d';
        exceptions: ['legal-obligation', 'public-interest'];
        cascading: 'automatic';
    };
    portability: {
        responseTime: '30d';
        format: 'structured';
        directTransfer: 'supported';
    };
    objection: {
        responseTime: '30d';
        assessment: 'case-by-case';
        stopProcessing: 'immediate';
    };
};

dataProtectionImpactAssessment: {
    triggers: DPIATrigger[];
    methodology: 'cnil-framework';
    review: 'annual';
};

dataBreachResponse: {
    detection: '<72h';
    notification: {
        authority: '<72h';
        subjects: '<72h-if-high-risk';
    };
    documentation: 'comprehensive';
};
};

// Conformité Loi 09-08 Marocaine
moroccanLawCompliance: {
    dataLocalization: {
        sensitiveData: 'morocco-only';
        personalData: 'morocco-preferred';
        backups: 'morocco-required';
    };

    cndpRegistration: {
        status: 'registered';
        renewalDate: Date;
        filingNumber: string;
    };

    crossBorderTransfers: {

```



```

        adequacyDecisions: string[];
        safeguards: 'standard-contractual-clauses';
        authorization: 'cndp-approved';
    };

    dataProcessorAgreements: {
        template: 'cndp-compliant';
        auditRights: 'included';
        subprocessors: 'pre-approved';
    };
};

// Gouvernance et audit
complianceGovernance: {
    dpo: {
        appointed: true;
        contact: 'dpo@salambot.ma';
        independence: 'organizational';
    };

    policies: {
        privacyPolicy: {
            lastUpdated: Date;
            language: ['fr', 'ar'];
            accessibility: 'wcag-aa';
        };
        cookiePolicy: {
            consent: 'granular';
            categories: ['necessary', 'analytics', 'marketing'];
        };
        dataRetention: {
            schedule: RetentionSchedule;
            automation: 'enabled';
            exceptions: 'documented';
        };
    };

    training: {
        frequency: 'quarterly';
        audience: ['developers', 'support', 'management'];
        certification: 'required';
    };

    auditTrail: {
        retention: '7y';
        integrity: 'cryptographic-hash';
        access: 'role-based';
    };
};
}

class ComplianceManager {

```

```

async handleDataSubjectRequest(
  request: DataSubjectRequest
): Promise<ComplianceResponse> {
  // Validation de l'identité
  const identity = await this.verifyIdentity(request.subject);

  // Traitement selon le type de demande
  switch (request.type) {
    case 'access':
      return await this.handleAccessRequest(identity,
request);
    case 'rectification':
      return await this.handleRectificationRequest(identity,
request);
    case 'erasure':
      return await this.handleErasureRequest(identity,
request);
    case 'portability':
      return await this.handlePortabilityRequest(identity,
request);
    case 'objection':
      return await this.handleObjectionRequest(identity,
request);
    default:
      throw new UnsupportedRequestTypeError(request.type);
  }
}

private async handleErasureRequest(
  identity: VerifiedIdentity,
  request: DataSubjectRequest
): Promise<ComplianceResponse> {
  // Vérification des exceptions légales
  const legalExceptions = await
this.checkLegalExceptions(identity);
  if (legalExceptions.length > 0) {
    return {
      status: 'partial',
      message: 'Certaines données ne peuvent être supprimées
pour des raisons légales',
      exceptions: legalExceptions,
      deletedData: []
    };
  }

  // Suppression en cascade
  const deletionPlan = await
this.createDeletionPlan(identity);
  const deletionResults = await
this.executeDeletionPlan(deletionPlan);

  // Notification aux sous-traitants

```

```

    await this.notifyProcessors(identity, 'erasure');

    // Audit trail
    await this.logComplianceAction({
      type: 'erasure',
      subject: identity.id,
      timestamp: new Date(),
      results: deletionResults
    });

    return {
      status: 'completed',
      message: 'Toutes vos données ont été supprimées',
      deletedData: deletionResults.deletedItems,
      confirmationId: deletionResults.confirmationId
    };
  }
}

```

7.4.3 Plan de Continuité et Disaster Recovery

```

/**
 * 🚨 SALAMBOT DISASTER RECOVERY PLAN v2.2
 *
 * 🎯 Mission: Continuité service + récupération rapide
 * 🕒 Objectifs: RTO <4h | RPO <1h | Disponibilité 99.9%
 * 🌐 Stratégie: Multi-région + sauvegarde automatique
 */

interface DisasterRecoveryPlan {
  // Objectifs de récupération
  recoveryObjectives: {
    rto: '4h'; // Recovery Time Objective
    rpo: '1h'; // Recovery Point Objective
    availability: 0.999; // 99.9% uptime
    dataLoss: 'minimal'; // <1h de données
  };

  // Stratégie multi-région
  multiRegionStrategy: {
    primary: {
      region: 'europe-west1'; // Belgique (proche Maroc)
      zones: ['a', 'b', 'c'];
      capacity: '100%';
    };

    secondary: {
      region: 'europe-west3'; // Allemagne
      zones: ['a', 'b'];
      capacity: '50%'; // Standby
    };
  };
}

```

```
    activationTime: '<30min';
};

emergency: {
    region: 'us-east1'; // Fallback global
    zones: ['a'];
    capacity: '25%'; // Emergency only
    activationTime: '<2h';
};
};

// Sauvegarde et réplication
backupStrategy: {
    databases: {
        frequency: 'continuous'; // Point-in-time recovery
        retention: '30d';
        crossRegion: true;
        encryption: 'AES-256';
        testing: 'weekly';
    };

    files: {
        frequency: 'hourly';
        retention: '90d';
        versioning: true;
        compression: true;
        testing: 'daily';
    };

    configurations: {
        frequency: 'on-change';
        retention: '1y';
        versionControl: 'git';
        automation: 'terraform';
    };
};

// Procédures de récupération
recoveryProcedures: {
    // Panne partielle (service dégradé)
    partialOutage: {
        detection: 'automatic';
        response: 'auto-scaling';
        escalation: '15min';
        communication: 'status-page';
    };

    // Panne régionale (région principale)
    regionalOutage: {
        detection: 'health-checks';
        response: 'failover-secondary';
        escalation: 'immediate';
    };
};
```

```

        communication: 'all-channels';
        steps: [
            'Activate secondary region',
            'Redirect DNS traffic',
            'Restore from latest backup',
            'Validate service functionality',
            'Update monitoring dashboards'
        ];
    };

    // Catastrophe majeure (multi-région)
    majorDisaster: {
        detection: 'manual-trigger';
        response: 'emergency-procedures';
        escalation: 'c-level';
        communication: 'crisis-mode';
        steps: [
            'Activate emergency region',
            'Restore from cross-region backups',
            'Implement degraded service mode',
            'Communicate with all stakeholders',
            'Execute business continuity plan'
        ];
    };
};

// Tests et validation
drTesting: {
    frequency: 'quarterly';
    scope: 'full-system';
    scenarios: [
        'database-failure',
        'region-outage',
        'network-partition',
        'security-breach',
        'data-corruption'
    ];

    metrics: {
        recoveryTime: Metric<number>;
        dataLoss: Metric<number>;
        serviceAvailability: Metric<number>;
        userImpact: Metric<number>;
    };

    documentation: {
        runbooks: 'updated-after-each-test';
        lessons: 'documented-and-shared';
        improvements: 'implemented-within-30d';
    };
};
}

```

```
class DisasterRecoveryManager {
  async executeFailover(
    scenario: DisasterScenario,
    targetRegion: string
  ): Promise<FailoverResult> {
    const startTime = new Date();

    try {
      // Phase 1: Évaluation et préparation
      const assessment = await this.assessDisaster(scenario);
      await this.notifyStakeholders('failover-initiated',
assessment);

      // Phase 2: Activation région de secours
      const activationResult = await
this.activateSecondaryRegion(targetRegion);

      // Phase 3: Redirection du trafic
      await this.redirectTraffic(targetRegion);

      // Phase 4: Restauration des données
      const restoreResult = await this.restoreFromBackup(
        assessment.lastKnownGoodBackup
      );

      // Phase 5: Validation des services
      const validationResult = await this.validateServices();

      // Phase 6: Mise à jour monitoring
      await this.updateMonitoring(targetRegion);

      const endTime = new Date();
      const recoveryTime = endTime.getTime() -
startTime.getTime();

      // Logging et métriques
      await this.logFailoverEvent({
        scenario,
        targetRegion,
        recoveryTime,
        dataLoss: restoreResult.dataLoss,
        success: true
      });

      await this.notifyStakeholders('failover-completed', {
        recoveryTime,
        dataLoss: restoreResult.dataLoss,
        serviceStatus: 'operational'
      });

      return {
```

```

        success: true,
        recoveryTime,
        dataLoss: restoreResult.dataLoss,
        servicesRestored: validationResult.services,
        nextSteps: this.generateRecoveryNextSteps(scenario)
    };

    } catch (error) {
        await this.handleFailoverError(error, scenario);
        throw error;
    }
}

private async restoreFromBackup(
    backupId: string
): Promise<RestoreResult> {
    const backupInfo = await this.getBackupInfo(backupId);

    // Restauration base de données
    const dbRestore = await
this.restoreDatabase(backupInfo.database);

    // Restauration fichiers
    const fileRestore = await
this.restoreFiles(backupInfo.files);

    // Restauration configurations
    const configRestore = await
this.restoreConfigurations(backupInfo.configs);

    return {
        database: dbRestore,
        files: fileRestore,
        configurations: configRestore,
        dataLoss: this.calculateDataLoss(backupInfo.timestamp),
        integrity: await this.verifyDataIntegrity()
    };
}
}

```

Section 4.9 - Plan d'Action Immédiat Révisé

4.9.1 Priorités Critiques (2 Semaines)

Suite à l'intégration des recommandations critiques, le plan d'action immédiat est restructuré pour adresser en priorité absolue les lacunes techniques identifiées.

```
/**
 * 🚨 PLAN D'ACTION CRITIQUE v2.2
 *
 * 🎯 Mission: Correction lacunes critiques + stabilisation
 * 🕒 Timeline: 2 semaines intensives + validation
 * 🔥 Focus: Darija 90%+ | API Gateway | Fallback IA
 */
```

```
interface CriticalActionPlan {
  // Semaine 1: Audit et prototypage
  week1: {
    // Jour 1-2: Audit technique détection Darija
    darijaAudit: {
      tasks: [
        'Audit précision actuelle (baseline 45%)',
        'Analyse des échecs de détection',
        'Identification patterns manqués',
        'Évaluation dictionnaire existant'
      ];
      deliverables: [
        'Rapport audit technique détaillé',
        'Dataset test 500+ phrases Darija',
        'Analyse des lacunes critiques',
        'Recommandations d\'amélioration'
      ];
      team: 'ai-research-team';
      priority: 'P0-Critical';
    };

    // Jour 3-4: Prototype détection bi-script
    darijaPrototype: {
      tasks: [
        'Développement pipeline multi-couches',
        'Implémentation dictionnaire bi-script',
        'Tests sur dataset de validation',
        'Métriques de performance initiales'
      ];
      deliverables: [
        'Prototype fonctionnel détection bi-script',
        'Dictionnaire 500+ termes critiques',
        'Tests unitaires et d\'intégration',
        'Métriques précision prototype'
      ];
      team: 'ai-research-team + backend-team';
      priority: 'P0-Critical';
    };

    // Jour 5: Évaluation API Gateway
    apiGatewayEval: {
      tasks: [
        'Évaluation Kong vs Tyk vs Custom',

```



```

        'PoC architecture gateway',
        'Tests de performance',
        'Analyse coût/bénéfice'
    ];
    deliverables: [
        'Comparatif technique détaillé',
        'PoC fonctionnel gateway',
        'Recommandation technologique',
        'Plan d\'implémentation détaillé'
    ];
    team: 'platform-team';
    priority: 'P0-Critical';
};
};

// Semaine 2: Implémentation et validation
week2: {
    // Jour 6-8: Implémentation Darija optimisée
    darijaImplementation: {
        tasks: [
            'Intégration pipeline production',
            'Extension dictionnaire 2000+ termes',
            'Optimisation performance <500ms',
            'Tests de charge et validation'
        ];
        deliverables: [
            'Détection Darija >85% précision',
            'Dictionnaire bi-script complet',
            'Performance <500ms validée',
            'Tests de régression passés'
        ];
        team: 'ai-research-team + backend-team';
        priority: 'P0-Critical';
    };

    // Jour 9-10: Stratégie fallback IA
    fallbackImplementation: {
        tasks: [
            'Configuration multi-provider',
            'Tests de basculement automatique',
            'Monitoring et alertes',
            'Documentation opérationnelle'
        ];
        deliverables: [
            'Fallback IA opérationnel',
            'Tests de résilience validés',
            'Monitoring temps réel actif',
            'Runbooks opérationnels'
        ];
        team: 'platform-team + ai-team';
        priority: 'P1-High';
    };
};

```

```

    };
}

// Critères de succès semaine 1-2
interface Week12SuccessCriteria {
    darija: {
        accuracy: '>85%';           // Objectif intermédiaire
        latency: '<500ms';           // Performance cible
        coverage: '2000+ terms';     // Dictionnaire étendu
        testing: '500+ test cases';  // Validation robuste
    };

    apiGateway: {
        decision: 'technology-selected';
        poc: 'functional-prototype';
        performance: '<50ms-overhead';
        plan: 'detailed-implementation';
    };

    fallback: {
        providers: '3-configured';
        switching: 'automatic';
        monitoring: 'real-time';
        documentation: 'complete';
    };
}

```

4.9.2 Phase Correction Étendue (6-8 Semaines)

```

/**
 * 🛠️ PHASE CORRECTION ÉTENDUE v2.2
 *
 * 🎯 Mission: Atteindre objectifs >90% Darija + stabilité
 * 🕒 Timeline: 6-8 semaines développement intensif
 * 🎯 Objectif: Production-ready avec excellence technique
 */

interface ExtendedCorrectionPhase {
    // Semaines 3-4: Optimisation Darija avancée
    weeks34: {
        darijaOptimization: {
            objectives: [
                'Atteindre >90% précision Darija',
                'Optimiser performance <300ms',
                'Implémenter variations régionales',
                'Tests utilisateurs 50 entreprises'
            ];

            activities: [
                'Fine-tuning modèles LLM spécialisés',

```

```

        'Enrichissement dictionnaire régional',
        'Optimisation algorithmes détection',
        'Tests A/B avec utilisateurs réels'
    ];

    deliverables: [
        'Précision Darija >90% validée',
        'Performance <300ms en production',
        'Support variations régionales',
        'Feedback utilisateurs intégré'
    ];
};

apiGatewayDevelopment: {
    objectives: [
        'API Gateway production-ready',
        'Migration services existants',
        'Sécurité et monitoring complets',
        'Performance <50ms overhead'
    ];

    activities: [
        'Développement gateway complet',
        'Migration progressive services',
        'Implémentation sécurité avancée',
        'Tests de charge et validation'
    ];

    deliverables: [
        'API Gateway opérationnel',
        'Tous services migrés',
        'Sécurité multicouche active',
        'Performance validée'
    ];
};
};

// Semaines 5-6: Extension Chrome et intégrations
weeks56: {
    chromeExtensionMVP: {
        objectives: [
            'Extension Chrome MVP 1 fonctionnelle',
            'Auto-suggestion champs génériques',
            'Intégration API SalamBot optimisée',
            'Beta 100+ utilisateurs'
        ];

        activities: [
            'Développement extension Manifest v3',
            'Intégration détection Darija optimisée',
            'Tests sur sites populaires marocains',
            'Programme beta utilisateurs'
        ]
    }
};

```

```

];

deliverables: [
  'Extension Chrome publiée',
  'Fonctionnalités MVP validées',
  'Performance <500ms suggestion',
  'Feedback beta positif >4.0/5'
];
};

metricsAndMonitoring: {
  objectives: [
    'Système métriques complet',
    'Alertes temps réel opérationnelles',
    'Dashboards business et technique',
    'Rapports automatiques'
  ];

  activities: [
    'Implémentation métriques détaillées',
    'Configuration alertes critiques',
    'Développement dashboards Grafana',
    'Automatisation rapports'
  ];

  deliverables: [
    'Métriques temps réel actives',
    'Alertes configurées et testées',
    'Dashboards opérationnels',
    'Rapports automatiques'
  ];
};
};

// Semaines 7-8: Validation et préparation production
weeks78: {
  productionReadiness: {
    objectives: [
      'Validation complète système',
      'Tests de charge 10K utilisateurs',
      'Documentation complète',
      'Formation équipes'
    ];

    activities: [
      'Tests de charge et stress',
      'Validation sécurité et conformité',
      'Finalisation documentation',
      'Formation support et ops'
    ];

    deliverables: [

```

```

        'Système validé production',
        'Capacité 10K utilisateurs',
        'Documentation à jour',
        'Équipes formées'
    ];
};

launchPreparation: {
    objectives: [
        'Préparation lancement commercial',
        'Stratégie communication',
        'Support client opérationnel',
        'Métriques business actives'
    ];

    activities: [
        'Finalisation stratégie go-to-market',
        'Préparation matériel marketing',
        'Formation équipe commerciale',
        'Tests processus support'
    ];

    deliverables: [
        'Stratégie lancement finalisée',
        'Matériel marketing prêt',
        'Équipe commerciale formée',
        'Support client opérationnel'
    ];
};
};
}

```

4.9.3 Métriques de Validation Continue

```

/**
 * 📊 MÉTRIQUES VALIDATION CONTINUE v2.2
 *
 * 🎯 Mission: Suivi temps réel progrès + qualité
 * 📈 Fréquence: Quotidienne + alertes automatiques
 * 🎯 Objectifs: Transparence + amélioration continue
 */

interface ContinuousValidationMetrics {
    // Métriques techniques quotidiennes
    dailyTechnical: {
        darijaDetection: {
            accuracy: {
                current: Metric<number>;
                target: 0.90;
                trend: 'improving' | 'stable' | 'degrading';
            };
        };
    };
};

```

```

};
latency: {
  p50: Metric<number>;
  p95: Metric<number>;
  p99: Metric<number>;
  target: 300; // ms
};
coverage: {
  dictionaryTerms: Metric<number>;
  testCases: Metric<number>;
  regionalVariants: Metric<number>;
};
};

systemPerformance: {
  availability: {
    uptime: Metric<number>;
    target: 0.999;
    incidents: Metric<number>;
  };
  performance: {
    apiLatency: Metric<number>;
    throughput: Metric<number>;
    errorRate: Metric<number>;
  };
};
};

// Métriques qualité hebdomadaires
weeklyQuality: {
  codeQuality: {
    testCoverage: {
      overall: Metric<number>;
      darija: Metric<number>;
      target: 0.85;
    };
    codeReview: {
      approvalRate: Metric<number>;
      averageTime: Metric<number>;
      defectRate: Metric<number>;
    };
  };
};

userFeedback: {
  satisfaction: {
    overall: Metric<number>;
    darija: Metric<number>;
    target: 4.0; // /5.0
  };
  adoption: {
    activeUsers: Metric<number>;
    retention: Metric<number>;
  };
};

```

```

        engagement: Metric<number>;
    };
};
};

// Métriques business mensuelles
monthlyBusiness: {
    adoption: {
        newUsers: Metric<number>;
        churnRate: Metric<number>;
        ltvt: Metric<number>;
    };

    revenue: {
        mrr: Metric<number>;
        conversionRate: Metric<number>;
        cac: Metric<number>;
    };

    market: {
        competitivePosition: Metric<string>;
        marketShare: Metric<number>;
        brandAwareness: Metric<number>;
    };
};
}

class ValidationManager {
    async generateDailyReport(): Promise<ValidationReport> {
        const metrics = await this.collectAllMetrics();
        const analysis = await this.analyzeMetrics(metrics);
        const recommendations = await
this.generateRecommendations(analysis);

        const report = {
            date: new Date(),
            metrics,
            analysis,
            recommendations,
            alerts: this.checkAlerts(metrics),
            nextActions: this.prioritizeActions(recommendations)
        };

        // Distribution automatique
        await Promise.all([
            this.sendToSlack(report),
            this.updateDashboards(report),
            this.storeInDatabase(report),
            this.triggerAlertsIfNeeded(report)
        ]);

        return report;
    }
}

```

```

    }

    private checkAlerts(metrics: ContinuousValidationMetrics):
Alert[] {
    const alerts: Alert[] = [];

    // Alerte critique: Précision Darija dégradée
    if (metrics.dailyTechnical.darijaDetection.accuracy.current
< 0.85) {
        alerts.push({
            level: 'critical',
            type: 'darija-accuracy-degraded',
            message: `Précision Darija: $
{metrics.dailyTechnical.darijaDetection.accuracy.current * 100}
% (< 85%)`,
            action: 'Intervention immédiate équipe IA',
            escalation: 'ai-team-lead'
        });
    }

    // Alerte importante: Performance dégradée
    if (metrics.dailyTechnical.darijaDetection.latency.p95 >
500) {
        alerts.push({
            level: 'warning',
            type: 'performance-degraded',
            message: `Latence P95: $
{metrics.dailyTechnical.darijaDetection.latency.p95}ms (>
500ms)`,
            action: 'Optimisation performance requise',
            escalation: 'platform-team'
        });
    }

    return alerts;
}
}

```

Conclusion et Prochaines Étapes

Synthèse des Améliorations v2.2

La version 2.2 de ce cahier des charges représente une évolution majeure qui transforme SalamBot d'un projet prometteur en une solution techniquement robuste et commercialement viable. L'intégration des recommandations critiques issues de l'audit technique approfondi permet de corriger les lacunes identifiées et d'établir une feuille de route réaliste pour atteindre l'excellence opérationnelle.

Les améliorations apportées couvrent trois dimensions essentielles de la réussite du projet. Sur le plan technique, la refonte complète de la détection Darija bi-script avec un objectif de précision supérieur à 90% repositionne SalamBot comme le leader incontesté de la compréhension du dialecte marocain dans le domaine de l'intelligence artificielle conversationnelle. L'architecture multi-provider pour les services IA élimine les risques de vendor lock-in et assure une résilience opérationnelle de niveau entreprise. L'implémentation d'un système de métriques et d'alertes temps réel garantit une visibilité complète sur les performances et permet une amélioration continue basée sur des données factuelles.

Sur le plan opérationnel, la stratégie de fallback multicouche et les procédures de disaster recovery assurent une continuité de service même dans les scénarios les plus défavorables. Les standards de sécurité renforcés et la conformité native RGPD/Loi 09-08 marocaine positionnent SalamBot comme une solution de confiance pour les entreprises les plus exigeantes. L'extension Chrome avec approche MVP permet une adoption progressive et non-disruptive par les organisations disposant déjà de systèmes CRM ou helpdesk.

Sur le plan business, la roadmap révisée avec des jalons réalistes et mesurables offre une trajectoire claire vers la rentabilité. Les métriques de succès actualisées reflètent une compréhension mature des défis techniques et commerciaux. Le modèle économique diversifié avec options de déploiement souverain répond aux besoins spécifiques du marché marocain et nord-africain.

Plan d'Exécution Immédiat

L'exécution de cette version révisée nécessite une mobilisation immédiate des ressources sur les aspects critiques identifiés. La correction de la détection Darija constitue la priorité absolue et doit être traitée comme un projet de crise avec allocation de ressources dédiées et suivi quotidien des progrès. L'implémentation de l'API Gateway ne peut plus être différée et doit faire l'objet d'un proof-of-concept dans les deux semaines suivant l'adoption de ce cahier des charges.

La stratégie de fallback IA doit être configurée en parallèle pour éliminer le risque de single point of failure actuel. Les équipes techniques doivent être formées aux nouveaux outils de monitoring et de debugging pour assurer une résolution rapide des incidents. La documentation technique doit être mise à jour en continu pour refléter les évolutions architecturales.

L'engagement des parties prenantes business est crucial pour valider les hypothèses de marché et ajuster la stratégie produit en fonction des retours terrain. Les tests

utilisateurs avec 50 entreprises pilotes doivent être organisés dès que la précision Darija atteint 85% pour valider l'amélioration de l'expérience utilisateur.

Vision Long Terme

Au-delà des corrections immédiates, ce cahier des charges établit les fondations d'une plateforme d'intelligence artificielle conversationnelle de classe mondiale spécialisée dans les langues et dialectes du Maghreb. L'expertise unique en détection et traitement du Darija constitue un avantage concurrentiel durable qui peut être étendu à d'autres dialectes arabes régionaux.

L'architecture modulaire et les standards de qualité élevés permettront une expansion géographique progressive vers d'autres marchés francophones et arabophones. Les capacités d'intégration avec les systèmes existants via l'extension Chrome ouvrent des opportunités de partenariat avec les éditeurs de logiciels CRM et helpdesk.

La stratégie de données souveraines et la conformité réglementaire native positionnent SalamBot pour adresser les segments entreprise et gouvernemental, traditionnellement réticents aux solutions cloud étrangères. Cette approche différenciée peut générer des revenus récurrents significatifs avec des marges élevées.

Engagement Équipe

La réussite de cette transformation nécessite l'engagement total de toutes les équipes impliquées dans le développement et la commercialisation de SalamBot. Chaque membre de l'équipe doit comprendre l'importance critique des prochaines semaines et s'approprier les objectifs de qualité et de performance définis dans ce document.

Les équipes techniques doivent adopter une mentalité d'excellence opérationnelle où chaque ligne de code, chaque configuration, chaque test contribue à l'objectif global de précision et de fiabilité. Les équipes business doivent maintenir un dialogue constant avec les utilisateurs pilotes pour capturer les insights qui guideront les itérations futures.

La communication transparente des progrès, des défis et des succès renforce la cohésion d'équipe et maintient la motivation nécessaire pour surmonter les obstacles techniques et commerciaux. Les célébrations des jalons atteints et l'apprentissage collectif des échecs créent une culture d'amélioration continue essentielle au succès long terme.

Références et Sources

Documentation Technique

- [1] **Google Cloud Vertex AI Documentation** - <https://cloud.google.com/vertex-ai/docs>
Documentation officielle pour l'intégration des modèles Gemini Pro et la configuration des endpoints IA
- [2] **Genkit Framework Documentation** - <https://firebase.google.com/docs/genkit>
Guide complet pour l'implémentation des flows IA et l'intégration multi-provider
- [3] **Redis Cluster Configuration Guide** - <https://redis.io/docs/management/scaling/>
Meilleures pratiques pour la configuration Redis en mode cluster haute disponibilité
- [4] **Kong API Gateway Documentation** - <https://docs.konghq.com/gateway/latest/>
Documentation technique pour l'évaluation et l'implémentation de l'API Gateway
- [5] **OpenTelemetry JavaScript SDK** - <https://opentelemetry.io/docs/instrumentation/js/>
Guide d'implémentation pour l'observabilité et le monitoring distribué

Sécurité et Conformité

- [6] **RGPD - Règlement Général sur la Protection des Données** - <https://gdpr.eu/>
Texte officiel et guides d'implémentation pour la conformité européenne
- [7] **Loi 09-08 Marocaine sur la Protection des Données** - <https://www.cndp.ma/>
Réglementation marocaine et directives de la Commission Nationale de contrôle de la protection des Données à caractère Personnel
- [8] **OWASP Security Guidelines** - <https://owasp.org/www-project-top-ten/>
Standards de sécurité application web et meilleures pratiques
- [9] **Google Cloud Security Best Practices** - <https://cloud.google.com/security/best-practices>
Guide sécurité pour les déploiements cloud et la gestion des identités

Linguistique et IA

- [10] **Compact Language Detector 3 (CLD3)** - <https://github.com/google/cld3>
Bibliothèque de détection linguistique pour l'implémentation offline
- [11] **Arabic Natural Language Processing Research** - <https://arxiv.org/abs/2103.15692>
Recherches académiques sur le traitement automatique des dialectes arabes

[12] **Moroccan Darija Linguistic Studies** - <https://www.researchgate.net/publication/darija-moroccan>

Études linguistiques sur les spécificités du dialecte marocain

Architecture et DevOps

[13] **Nx Monorepo Documentation** - <https://nx.dev/getting-started/intro>

Guide complet pour la gestion du monorepo et l'orchestration des builds

[14] **Terraform Google Cloud Provider** - <https://registry.terraform.io/providers/hashicorp/google/latest/docs>

Documentation pour l'infrastructure as code sur Google Cloud Platform

[15] **Grafana Cloud Monitoring** - <https://grafana.com/docs/grafana-cloud/>

Configuration des dashboards et alertes pour l'observabilité temps réel

[16] **Vercel Deployment Guide** - <https://vercel.com/docs/deployments/overview>

Meilleures pratiques pour le déploiement des applications frontend

Business et Marché

[17] **Moroccan Digital Economy Report 2024** - <https://www.mcinet.gov.ma/>

Analyse du marché numérique marocain et opportunités sectorielles

[18] **MENA AI Market Analysis** - <https://www.pwc.com/m1/en/publications/artificial-intelligence-mena.html>

Étude de marché sur l'intelligence artificielle au Moyen-Orient et Afrique du Nord

[19] **Customer Service Automation Trends** - <https://www.gartner.com/en/customer-service-support>

Tendances globales de l'automatisation du service client

Outils et Technologies

[20] **Chrome Extension Manifest V3** - <https://developer.chrome.com/docs/extensions/mv3/>

Spécifications techniques pour le développement d'extensions Chrome modernes

[21] **WhatsApp Business API Documentation** - <https://developers.facebook.com/docs/whatsapp/>

Guide d'intégration pour les connecteurs WhatsApp Business

[22] **Next.js 15 Documentation** - <https://nextjs.org/docs>

Framework React pour les applications web modernes et performantes

[23] **Tailwind CSS 4 Documentation** - <https://tailwindcss.com/docs>
Framework CSS utilitaire pour le développement d'interfaces utilisateur

[24] **shadcn/ui Component Library** - <https://ui.shadcn.com/>
Bibliothèque de composants React modernes et accessibles



Document: Cahier des Charges SalamBot v2.2



Dernière mise à jour: 2 juin 2025



Auteurs: SalamBot Team + Manus AI + Recommandations Audit Technique



Contact: info@salambot.ma



Site web: <https://salambot.ma>



Support: +212-XXX-XXX-XXX

Ce document constitue la référence officielle pour le développement de SalamBot.
Toute modification doit être validée par l'équipe technique et business avant implémentation.



Made in Morocco with ❤️ for Moroccan SMEs