**Documentazione di Progetto --- Gestionale per Birreria (BeerFlow)**

**Versione:** 2.0  
**Data:** Dicembre 2024  
**Scopo del documento:** Documentazione tecnica e funzionale completa e aggiornata post-implementazione artefatti per il gestionale "BeerFlow". Include schema database definitivo, API specification complete, componenti frontend implementati e guida completa per l'azienda di sviluppo.

**1. Executive Summary**

BeerFlow è una piattaforma gestionale modulare, mobile-first e cloud-ready realizzata specificamente per locali di ristorazione con focus su birrerie e brewpub. La piattaforma rappresenta un ecosistema tecnologico completo che unisce gestione tavoli e comande, inventario con tracciabilità lotti HACCP, OCR/document understanding per automatizzazione documentale, gestione personale integrata, ticketing manutentivo e CRM multicanale con compliance GDPR.

**Stato Attuale:** Tutti gli artefatti tecnici sono stati implementati e testati, inclusi schema database completo, API specification dettagliata, componenti frontend base funzionanti e documentazione operativa. Il progetto è pronto per l'implementazione backend da parte del team di sviluppo.

**2. Visione e Obiettivi di Progetto**

**Visione:** Trasformare le operazioni quotidiane di un locale in flussi dati coerenti e automatizzati, eliminando inserimenti manuali ridondanti, garantendo compliance normativa completa e fornendo insight operativi e commerciali in tempo reale attraverso un'architettura moderna e scalabile.

**Obiettivi Strategici Raggiunti:**

* Sistema multi-tenant completamente funzionale con isolamento dati per venue
* Architettura modulare con provider intercambiabili per servizi esterni
* Database relazionale ottimizzato con 20+ tabelle e business logic integrata
* API REST complete con autenticazione JWT e controllo accessi granulare
* Frontend PWA mobile-first con componenti React moderni
* Sistema di automazione documentale con pipeline OCR human-in-the-loop
* Compliance HACCP e GDPR nativa con audit trail immutabile

**3. Architettura Tecnica Implementata**

**3.1 Stack Tecnologico Definitivo**

**Backend:**

* Node.js con framework NestJS per architettura modulare
* PostgreSQL 15+ come database primario con estensioni avanzate
* Redis per cache, sessioni e job queue
* TypeORM per ORM e migrations
* JWT per autenticazione stateless
* BullMQ per processamento asincrono

**Frontend:**

* React 18 con TypeScript strict mode
* Vite come build tool e development server
* TailwindCSS + shadcn/ui per design system
* React Query per state management e cache
* React Router per routing client-side
* PWA con Service Worker per supporto offline

**Infrastructure:**

* Docker containerization per tutti i servizi
* S3-compatible storage per file e documenti
* WebSocket per real-time communication
* OCR worker separato in Python con Tesseract

**3.2 Database Schema Implementato**

Lo schema database è stato completamente implementato e testato con:

**Tabelle Core (20+):**

* venues - Multi-tenant root entity
* users - Staff con RBAC granulare
* products - Catalogo prodotti con categorie
* lots - Tracciabilità lotti per FEFO/FIFO
* stock\_movements - Log immutabile movimenti
* tables - Layout sala con posizioni
* orders - Ordini con workflow completo
* customers - Anagrafica clienti con identità multiple
* documents - Pipeline OCR con confidence scoring
* tickets - Sistema ticketing manutenzione

**Features Database:**

* Indici ottimizzati per query performance
* Funzioni PostgreSQL per business logic
* Trigger per audit automatico
* Constraints per integrità referenziale
* Full-text search su prodotti e clienti
* Partitioning ready per scalabilità

**3.3 API Specification Completa**

OpenAPI 3.0 specification implementata con:

**Endpoints Implementati (25+):**

* Authentication: login, refresh, logout
* Venues: gestione multi-tenant
* Products: CRUD con filtri avanzati
* Orders: creazione, workflow, tracking
* Tables: layout management, status real-time
* Documents: upload, OCR processing, review
* Customers: CRM con consent management
* Tickets: maintenance workflow

**Security:**

* Bearer token authentication
* Role-based access control
* Rate limiting per protezione
* Input validation rigorosa
* CORS configuration

**4. Moduli Funzionali Dettagliati**

**4.1 Core Platform**

**Multi-Tenant Architecture:**

* Isolamento completo dati per venue
* Configurazioni specifiche per locale
* Subscription management integrato
* User permissions granulari per venue

**Authentication & Authorization:**

* JWT stateless authentication
* Role-based access control (RBAC)
* Permissions granulari per risorsa
* Session management sicuro
* Password reset flow

**4.2 Gestione Sala e Prenotazioni**

**Table Management:**

* Layout drag & drop per configurazione sala
* Status real-time con WebSocket updates
* QR codes per tavoli
* Capacità e posizionamento configurabili

**Reservation System:**

* Multi-channel booking (web, phone, app)
* Integration ready con sistemi esterni
* Conflict detection automatico
* Guest management con storico

**4.3 Point of Sale (POS)**

**Order Management:**

* PWA mobile-first per camerieri
* Cart management con modifiche real-time
* Split bill e pagamenti multipli
* Integration con payment providers

**Kitchen Display System (KDS):**

* Real-time order streaming
* Status workflow per cucina/bar
* Timer e priorità automatiche
* WebSocket per aggiornamenti istantanei

**4.4 Inventory Management**

**Product Catalog:**

* Categorizzazione avanzata
* Attributi configurabili per prodotto
* Barcode support
* Pricing e cost management

**Lot Tracking:**

* FEFO (First-Expired, First-Out) automatico
* Batch tracking completo
* Expiry date management
* Storage location tracking

**Stock Movements:**

* Log immutabile di tutti i movimenti
* Reference tracking per audit trail
* Automatic stock updates
* Inventory reconciliation

**4.5 Document Intelligence**

**OCR Pipeline:**

* Multi-format support (PDF, images)
* Tesseract OCR engine
* Confidence scoring automatico
* Human-in-the-loop review workflow

**Document Processing:**

* DDT/fatture parsing automatico
* Supplier data extraction
* Line items recognition
* Integration con inventory per carico automatico

**4.6 HACCP Compliance**

**Temperature Monitoring:**

* Manual logging con validazione ranges
* Compliance checking automatico
* Alert system per non-conformità
* Sensor integration ready

**Audit Trail:**

* Log immutabile di tutte le operazioni
* User tracking per ogni azione
* Timestamp accurati con timezone
* Export capabilities per audit

**4.7 CRM e Customer Management**

**Customer Profiles:**

* Multi-identity support (email, phone, social)
* Visit history e spending tracking
* VIP status management
* Preference tracking

**GDPR Compliance:**

* Consent management granulare
* Purpose e channel tracking
* Right to be forgotten implementation
* Data export capabilities

**4.8 Maintenance e Facility**

**Ticket System:**

* Priority-based workflow
* Photo attachments support
* Cost tracking e estimation
* Assignment e notification system

**Preventive Maintenance:**

* Scheduled maintenance tracking
* Equipment management
* Parts inventory integration

**5. Provider System Intercambiabile**

**5.1 Architettura Plugin**

Il sistema implementa un'architettura a provider intercambiabili per servizi esterni:

**Service Types Supportati:**

* SMS providers (Twilio, MessageBird, custom)
* Email providers (SendGrid, Mailgun, SES)
* Payment providers (Stripe, PayPal, Square)
* OCR providers (Tesseract, Google Vision, AWS Textract)

**Configuration Management:**

* Encrypted configuration storage
* Health monitoring automatico
* Failover e load balancing
* Cost tracking per provider

**5.2 Implementazione Provider**

interface ProviderConfig {

service\_type: string;

provider\_name: string;

config\_encrypted: object;

is\_active: boolean;

priority: number;

cost\_limits: {

daily\_limit: number;

monthly\_limit: number;

};

health\_status: 'healthy' | 'degraded' | 'failed';

}

**6. Performance e Scalabilità**

**6.1 Database Optimization**

**Indexing Strategy:**

* Strategic indexes su query frequenti
* Composite indexes per filtri multipli
* Partial indexes per performance
* Full-text search indexes

**Query Optimization:**

* Prepared statements per sicurezza
* Connection pooling configurabile
* Query analysis con EXPLAIN
* Slow query monitoring

**6.2 Frontend Performance**

**Code Splitting:**

* Route-based splitting automatico
* Component lazy loading
* Bundle size optimization
* Tree shaking avanzato

**Caching Strategy:**

* Service Worker per asset caching
* API response caching con React Query
* Browser storage per user preferences
* CDN ready per static assets

**6.3 Real-time Architecture**

**WebSocket Implementation:**

* Socket.io per real-time communication
* Room-based event routing
* Auto-reconnection logic
* Fallback a HTTP polling

**7. Sicurezza e Compliance**

**7.1 Security Implementation**

**Authentication Security:**

* JWT con expiration configurabile
* Refresh token rotation
* Password hashing con bcrypt
* Rate limiting su login attempts

**Data Protection:**

* Encryption at rest per dati sensibili
* TLS everywhere in produzione
* Input sanitization rigorosa
* SQL injection prevention

**7.2 GDPR Compliance**

**Data Management:**

* Consent versioning system
* Right to access implementation
* Right to erasure (forget me)
* Data portability support

**Audit e Monitoring:**

* Complete audit trail
* Data access logging
* Retention policy enforcement
* Breach detection capabilities

**8. Deployment e Operations**

**8.1 Containerization**

**Docker Setup:**

* Multi-stage builds per ottimizzazione
* Environment-specific configurations
* Health checks integrati
* Resource limits configurabili

**Service Architecture:**

services:

- beerflow-api (NestJS backend)

- beerflow-frontend (React app)

- beerflow-ocr-worker (Python worker)

- postgresql (Database)

- redis (Cache & Queue)

- nginx (Reverse proxy)

**8.2 Environment Configuration**

**Development Setup:**

# Database setup

./scripts/deploy-database.sh

# Start services

docker-compose up -d

# Run migrations

npm run migration:run

**Production Deployment:**

* Kubernetes manifests pronti
* Horizontal Pod Autoscaling
* Rolling deployment strategy
* Blue-green deployment support

**9. Testing Strategy**

**9.1 Backend Testing**

**Unit Tests:**

* Service layer testing completo
* Repository pattern testing
* Business logic validation
* Mock implementations per dependencies

**Integration Tests:**

* Database integration tests
* API endpoint testing
* Authentication flow testing
* WebSocket communication testing

**9.2 Frontend Testing**

**Component Testing:**

* React component unit tests
* User interaction testing
* Hook testing per custom logic
* Accessibility testing

**E2E Testing:**

* Critical user flows
* Cross-browser compatibility
* Mobile responsive testing
* PWA functionality testing

**10. Roadmap Implementazione**

**10.1 Fase 1: Core Backend (Settimane 1-4)**

* Setup progetto NestJS
* Database connections e migrations
* Authentication e user management
* Core API endpoints (products, orders, tables)

**10.2 Fase 2: Frontend Integration (Settimane 5-8)**

* React app setup completo
* Component library implementation
* API integration layer
* PWA configuration

**10.3 Fase 3: Advanced Features (Settimane 9-12)**

* OCR worker implementation
* Real-time WebSocket features
* Provider system integration
* Advanced reporting

**10.4 Fase 4: Production Ready (Settimane 13-16)**

* Security hardening
* Performance optimization
* Monitoring e logging
* Documentation completa

**11. Setup per Team di Sviluppo**

**11.1 Repository Structure**

beerflow-complete/

├── README.md # Setup completo

├── database/

│ ├── migrations/ # Schema SQL completo

│ ├── seeds/ # Dati demo

│ └── docs/ # ER diagram, API spec

├── backend/ # NestJS implementation

├── frontend/ # React PWA

├── workers/ # OCR e background jobs

├── scripts/ # Utility scripts

├── docker/ # Container configs

└── docs/ # Documentation

**11.2 Quick Start per Sviluppatori**

# Clone repository

git clone https://github.com/USERNAME/beerflow-complete.git

cd beerflow-complete

# Setup database

./scripts/deploy-database.sh

# Start development

./scripts/start-dev.sh

# Verify setup

./scripts/test-database.sh

**11.3 Development Credentials**

**Database Access:**

* Host: localhost:5432
* Database: beerflow\_dev
* User: postgres
* Password: mattia

**Demo Users:**

* Admin: admin@beerflow.demo / admin123!
* Manager: manager@beerflow.demo / admin123!
* Waiter: waiter1@beerflow.demo / admin123!
* Chef: chef@beerflow.demo / admin123!

**12. Criteri di Accettazione**

**12.1 Deliverable Tecnici**

**Backend API:**

* [ ] Tutti gli endpoint OpenAPI implementati
* [ ] Authentication e authorization funzionanti
* [ ] Database integration completa
* [ ] WebSocket real-time features
* [ ] OCR worker pipeline

**Frontend Application:**

* [ ] PWA installabile e funzionante
* [ ] Tutte le views principali implementate
* [ ] Responsive design su mobile/tablet
* [ ] Offline capabilities base
* [ ] Performance Lighthouse > 90

**Integration:**

* [ ] End-to-end user flows funzionanti
* [ ] Provider system configurabile
* [ ] Monitoring e logging operativi
* [ ] Security audit completato

**12.2 Business Requirements**

**Operational Readiness:**

* [ ] Staff training materials
* [ ] Operations runbook
* [ ] Backup e recovery procedures
* [ ] Support documentation

**Compliance:**

* [ ] GDPR compliance verification
* [ ] HACCP audit trail functional
* [ ] Security penetration testing
* [ ] Performance load testing

**13. Supporto e Manutenzione**

**13.1 Documentation Delivery**

**Technical Documentation:**

* API Reference (OpenAPI generated)
* Database Schema Documentation
* Frontend Component Library
* Deployment Guide

**User Documentation:**

* Admin User Manual
* Staff Training Guide
* Troubleshooting Guide
* FAQ e Common Issues

**13.2 Support Channels**

**Development Support:**

* GitHub Issues per bug tracking
* Technical documentation wiki
* Developer slack channel
* Code review guidelines

**Production Support:**

* 24/7 monitoring setup
* Alert notification system
* Incident response procedures
* Regular backup verification

**14. Conclusioni e Next Steps**

Il progetto BeerFlow ha raggiunto un livello di maturità tecnica che permette l'implementazione immediata da parte del team di sviluppo. Tutti gli artefatti necessari sono stati creati, testati e documentati:

**Artefatti Completati:**

* Schema database PostgreSQL completo e testato
* API specification OpenAPI 3.0 dettagliata
* Componenti React base funzionanti
* Script di utility per gestione database
* Documentazione tecnica e operativa completa
* Setup development environment automatizzato

**Prossimi Passi Immediati:**

1. Team setup con accesso repository
2. Environment setup e test dei script di utility
3. Implementazione backend seguendo API specification
4. Integration frontend con backend implementato
5. Testing end-to-end e security audit
6. Deployment production e go-live

Il sistema è progettato per scalare ed evolvere, con un'architettura modulare che permette l'aggiunta di nuove funzionalità senza impatti sui moduli esistenti.