

Fradiavolo Invoice Dashboard - Documentazione Tecnica Completa

1. Executive Summary & Panoramica del Sistema

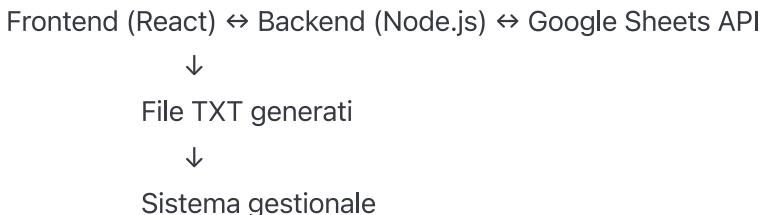
1.1 Scopo del Progetto

Il **Fradiavolo Invoice Dashboard** è un sistema di gestione digitale delle consegne e movimentazioni merce per la catena di pizzerie Fradiavolo. Il sistema automatizza il processo di conferma consegne, la gestione dei trasferimenti tra punti vendita e la generazione di file TXT per l'integrazione con sistemi gestionali.

1.2 Obiettivi Principali

- **Digitalizzazione** del processo di conferma consegne
- **Tracciabilità** completa delle movimentazioni tra negozi
- **Automazione** nella generazione di documenti (DDT, file TXT)
- **Centralizzazione** dei dati tramite Google Sheets
- **Controllo multi-livello** con ruoli differenziati (admin/operator)

1.3 Architettura Generale



1.4 Utenti Target

- **Operatori negozi** (26+ punti vendita): conferma consegne, movimentazioni
- **Amministratori centrali**: supervisione globale, reportistica, gestione utenti
- **Fornitori** (indirettamente): tracciabilità delle consegne

2. Architettura Tecnica

2.1 Stack Tecnologico

Frontend

- **React 18** - Framework UI principale
- **Tailwind CSS** - Sistema di styling utility-first

- **React Select** - Componenti dropdown avanzati
- **jsPDF + autoTable** - Generazione PDF per DDT
- **Lucide React** - Libreria di icone

Backend

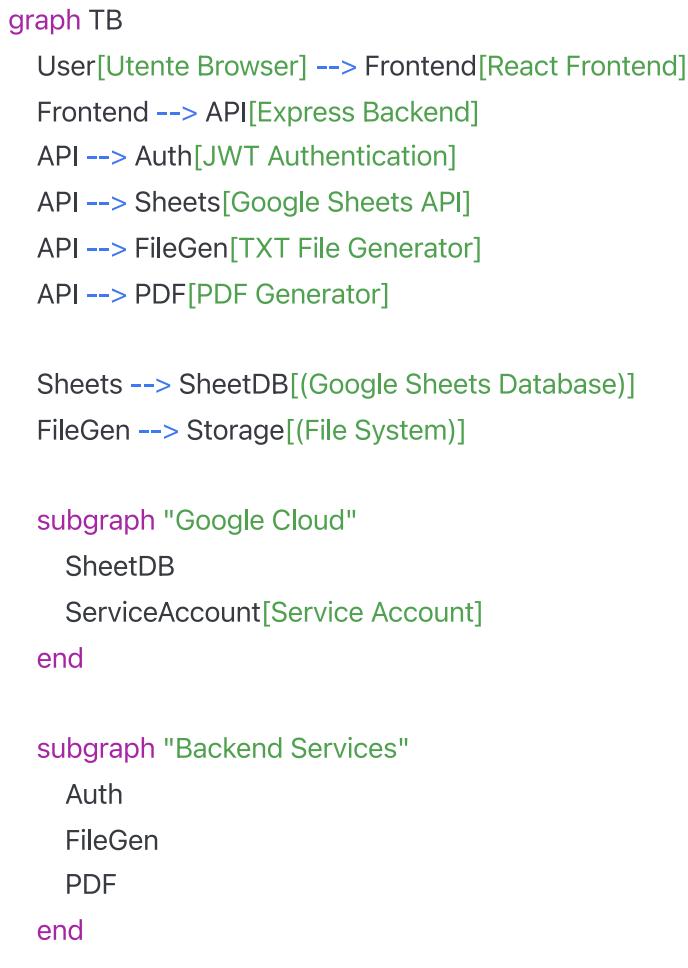
- **Node.js 16+** - Runtime JavaScript
- **Express.js** - Framework web server
- **Google Sheets API v4** - Integrazione database
- **JWT** - Autenticazione token-based
- **bcryptjs** - Hashing password
- **Helmet** - Sicurezza HTTP headers
- **Express Rate Limit** - Protezione DoS

Database & Storage

- **Google Sheets** - Database principale
- **File System locale** - Storage file TXT generati
- **In-memory storage** - Cache e sessioni

2.2 Diagramma dell'Architettura

```
mermaid
```



2.3 Flusso dei Dati

Processo Conferma Consegna

1. **Input:** Utente seleziona fattura e data consegna
2. **Validazione:** Backend verifica permessi e dati
3. **Update:** Modifica stato su Google Sheets
4. **Generazione:** Creazione automatica file TXT
5. **Notifica:** Feedback all'utente

Processo Movimentazione

1. **Input:** Selezione prodotti e destinazioni
2. **Calcolo:** Generazione contenuti TXT per ogni movimento
3. **Storage:** Salvataggio su Google Sheets (tab "Movimentazioni")
4. **Fatturazione:** Creazione fatture automatiche per ogni movimento
5. **Output:** Generazione DDT PDF e file TXT consolidato

3. Backend API (Node.js)

3.1 Struttura del Server Express

File Principale: server.js

- **Port:** 3001 (default) o variabile ambiente
- **CORS:** Abilitato per domini autorizzati
- **Rate Limiting:** 100 richieste/15min per IP
- **Security:** Helmet per headers sicuri

Middleware Stack

```
javascript
app.use(helmet());
app.use(cors({ origin: true, credentials: true }));
app.use(express.json({ limit: '10mb' }));
app.use(rateLimit({ windowMs: 15 * 60 * 1000, max: 100 }));
```

3.2 Sistema di Autenticazione

Endpoint: [/api/auth/login](#)

- **Modo:** POST
- **Validazione:** Email @fradiavolopizzeria.com obbligatoria
- **Security:** Rate limit 5 tentativi/15min
- **Response:** JWT token + dati utente

Endpoint: [/api/auth/verify](#)

- **Modo:** GET
- **Headers:** Authorization Bearer token
- **Purpose:** Validazione token e refresh dati utente

Database Utenti (In-Memory)

```
javascript
```

```

const users = [
{
  id: 1,
  name: "FDV Office",
  email: "office@fradiavolopizzeria.com",
  password: "fdv2025", // Plain text in POC
  puntoVendita: "FDV Office",
  role: "admin"
}
// ... 26+ negozi configurati
];

```

3.3 API Endpoints Completati

Fatture - GET `/api/invoices`

```

javascript

// Headers richiesti
Authorization: Bearer <jwt_token>

// Response
{
  "success": true,
  "data": [
    {
      "id": "unique_id",
      "numero": "FAT001",
      "fornitore": "Fornitore SRL",
      "data_emissione": "2025-01-15",
      "data_consegna": "2025-01-16",
      "stato": "consegnato|pending",
      "punto_vendita": "FDV Milano Isola",
      "confermato_da": "email@domain.com",
      "pdf_link": "https://...",
      "importo_totale": "150.00",
      "note": "Note eventuali",
      "txt": "Contenuto file TXT",
      "codice_fornitore": "FORN001"
    }
  ]
}

```

Conferma Consegna - POST `/api/invoices/:id/confirm`

javascript

```
// Body richiesto
{
  "data_consegna": "2025-01-16", // YYYY-MM-DD
  "note_errori": "Problemi riscontrati" // opzionale
}
```

// Processo backend

1. Validazione **data** (non futura)
2. Update Google Sheets
3. Generazione automatica file **TXT**
4. Response conferma

Movimentazioni - POST [/api/movimenti](#)

javascript

```
// Body richiesto
{
  "movimenti": [
    {
      "prodotto": { "value": "Pizza Margherita", "codice": "PIZZA001", "uom": "PZ" },
      "quantita": "24",
      "destinazione": "FDV Roma Parioli",
      "txt_content": "PIZZA001 ; 24.00000000000000"
    }
  ],
  "origine": "FDV Milano Isola"
}
```

// Processo backend

1. Validazione prodotti e quantità
2. Calcolo righe **TXT** standardizzate
3. Salvataggio su Google Sheets
4. Generazione fatture automatiche
5. Creazione file **TXT** unificato

Prodotti - GET [/api/prodotti](#)

javascript

```

// Query parameters opzionali
?active=1&page=1&per_page=1000

// Response
{
  "success": true,
  "total": 500,
  "returned": 100,
  "data": [
    {
      "nome": "VETRO PEPSI 33CL",
      "codice": "PEPSI001",
      "unitaMisura": "PCS",
      "onOff": "1",
      "brand": "Pepsi",
      "pack": "33CL",
      "materiale": "Vetro"
    }
  ]
}

```

3.4 Integrazione Google Sheets

Configurazione Service Account

```

javascript

const serviceAccountAuth = new JWT({
  email: process.env.GOOGLE_SERVICE_ACCOUNT_EMAIL,
  key: process.env.GOOGLE_PRIVATE_KEY.replace(/\n/g, '\n'),
  scopes: ['https://www.googleapis.com/auth/spreadsheets']
});

```

Schema Google Sheets - Tab "Fatture"

Colonna	Descrizione	Tipo	Obbligatorio
A	id	string	Sì
B	numero	string	Sì
C	fornitore	string	Sì
D	data_emissione	date	Sì
E	data_consegna	date	No
F	stato	enum	Sì
G	punto_vendita	string	Sì
H	confermato_da	email	No
I	pdf_link	url	No
J	importo_totale	decimal	No
K	note	text	No
L	txt	text	No
M	codice_fornitore	string	No

Schema Google Sheets - Tab "Movimentazioni"

Colonna	Descrizione	Tipo
A	id	string
B	data_movimento	date
C	timestamp	datetime
D	origine	string
E	codice_origine	string
F	prodotto	string
G	quantita	decimal
H	unita_misura	string
I	destinazione	string
J	codice_destinazione	string
K	stato	enum
L	txt_content	text
M	txt_filename	string
N	creato_da	email

3.5 Generazione File TXT

Formato Standard

CODICE_PRODOTTO ; QUANTITA_DECIMALE_15_CIFRE

Esempio Output

```
PEPSI001 ; 24.0000000000000000  
PIZZA001 ; 12.0000000000000000  
LATTE001 ; 5.5000000000000000
```

Processo Generazione

javascript

```
const generateTxtFile = async (invoiceData) => {  
    // 1. Estrazione dati fattura  
    const numeroDocumento = invoiceData.numero;  
    const dataEmissione = invoiceData.data_emissione;  
    const nomeFornitore = invoiceData.fornitore;  
    const codicePV = negozio?.codice || 'UNKNOWN';  
  
    // 2. Generazione nome file  
    const fileName = `${numeroDoc}_${dataEmissione}_${fornitore}_${codicePV}.txt`;  
  
    // 3. Scrittura contenuto  
    await fs.writeFile(filePath, contenutoTxt, 'utf8');  
  
    // 4. Return metadata  
    return { fileName, filePath, size: contenutoTxt.length };  
};
```

4. Frontend (React)

4.1 Struttura Componenti

Componente Principale: App.js

- **State Management:** useState hooks per gestione stato globale
- **Routing:** Condizionale basato su ruolo utente
- **Authentication:** Gestione token JWT localStorage
- **API Integration:** Chiamate fetch con gestione errori

Componenti Specializzati

```

src/
├── App.js          # Componente principale
├── Movimentazione.js # Gestione movimentazioni
├── TxtFilesManager.js # Manager file TXT
└── components/
    ├── AdminDashboard.js
    ├── AdminSidebarLayout.js
    ├── AdminInvoiceManager.js
    ├── AdminMovimentazioniManager.js
    └── AdminUserManager.js
    └── data/
        ├── negozi.json      # Configurazione punti vendita
        └── prodotti.json    # Database prodotti (fallback)

```

4.2 Sistema di Autenticazione Frontend

Login Flow

```

javascript

const handleLogin = async (e) => {
  e.preventDefault();

  try {
    const response = await apiCall('/auth/login', {
      method: 'POST',
      body: JSON.stringify(loginForm)
    });

    if (response.success) {
      setToken(response.token);
      setUser(response.user);
      localStorage.setItem('token', response.token);
    }
  } catch (error) {
    setLoginError(error.message);
  }
};

```

Token Verification

```

javascript

```

```

const verifyToken = async () => {
  try {
    const response = await apiCall('/auth/verify');
    setUser(response.user);
  } catch (error) {
    localStorage.removeItem('token');
    setToken(null);
    setUser(null);
  }
};

```

4.3 Gestione Stato e Props

Stato Globale App.js

```

javascript

const [user, setUser] = useState(null);           // Dati utente corrente
const [token, setToken] = useState(localStorage.getItem('token'));
const [sheetInvoices, setSheetInvoices] = useState([]); // Cache fatture
const [activeTab, setActiveTab] = useState('pending'); // Tab attivo
const [isLoading, setIsLoading] = useState(false);   // Loading state
const [error, setError] = useState('');             // Messaggi errore
const [success, setSuccess] = useState('');          // Messaggi successo

```

Props Drilling Pattern

```

javascript

// Da App.js verso componenti figli
<Movimentazione user={user} />
<TxtFilesManager user={user} />
<AdminDashboard user={user} />

```

4.4 Design System Fradiavolo

Colori Brand (Tailwind Config)

```

javascript

```

```

colors: {
  'fradiavolo-red': '#C41E3A',      // Rosso principale
  'fradiavolo-red-dark': '#A31627',   // Rosso scuro
  'fradiavolo-cream': '#FFF8DC',     // Crema chiaro
  'fradiavolo-cream-dark': '#F5F5DC', // Crema scuro
  'fradiavolo-charcoal': '#2C2C2C',   // Grigio scuro
  'fradiavolo-charcoal-light': '#4A4A4A', // Grigio medio
  'fradiavolo-green': '#28A745',      // Verde successo
  'fradiavolo-green-dark': '#1E7E34',   // Verde scuro
  'fradiavolo-orange': '#FF8C00',      // Arancione warning
  'fradiavolo-gold': '#FFD700'        // Oro accenti
}

```

Component Styling Pattern

javascript

```

// Button Primary
"px-6 py-3 bg-fradiavolo-red hover:bg-fradiavolo-red-dark text-white rounded-xl hover:shadow-fradiavolo transition"

// Card Container
"bg-white rounded-xl shadow-fradiavolo p-6 border border-fradiavolo-cream-dark"

// Alert Success
"bg-fradiavolo-green/10 text-fradiavolo-green-dark border-fradiavolo-green/30"

```

4.5 Responsive Design

Breakpoint Strategy

- **Mobile First:** Design base per schermi piccoli
- **sm (640px+):** Tablet portrait
- **md (768px+):** Tablet landscape / Desktop small
- **lg (1024px+):** Desktop standard
- **xl (1280px+):** Desktop large

Layout Patterns

javascript

```
// Grid responsivo  
"grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4"  
  
// Flex responsive  
"flex flex-col sm:flex-row sm:items-center sm:justify-between"  
  
// Padding responsive  
"px-3 sm:px-6 lg:px-8"  
  
// Text responsive  
"text-lg sm:text-xl lg:text-2xl"
```

5. Database e Modelli Dati

5.1 Google Sheets come Database

Vantaggi Implementazione

- **Zero Setup:** Nessuna configurazione database tradizionale
- **Accessibilità:** Visualizzazione diretta dati via Google Sheets
- **Collaborazione:** Condivisione immediata con stakeholder
- **Backup:** Automatico tramite Google Drive
- **Costi:** Incluso in Google Workspace

Limitazioni Identificate

- **Concorrenza:** Gestione accessi simultanei limitata
- **Performance:** Latenza API calls (500ms-2s)
- **Scalabilità:** Max 5M celle per foglio
- **Query:** Nessun SQL nativo, filtri client-side

5.2 Struttura Dati Negozi

File: data/negozi.json

```
json
```

```
[
  {
    "nome": "FDV Milano Isola",
    "codice": "120",
    "indirizzo": "Via Tahan Di Ravel"
  },
  {
    "nome": "FDV Roma Parioli",
    "codice": "107",
    "indirizzo": "Via Po"
  }
]
// ... 26 totali
```

Utilizzo nel Sistema

- **Validazione:** Controllo destinazioni movimentazioni
- **UI:** Dropdown selezione negozi
- **File TXT:** Generazione nomi file con codici
- **DDT:** Indirizzi mittente/destinatario

5.3 Struttura Dati Prodotti

Fonte Primaria: Google Sheets Prodotti

- **Sheet ID:** 1CJhd14F8qV8nS0-SK2ENSNSkWaE21KotK2ArBjJETfk
- **Tab:** "PRODOTTI"
- **Aggiornamento:** Real-time via API

Schema Prodotti

```
javascript

{
  nome: "VETRO PEPSI 33CL",      // Descrizione prodotto
  codice: "PEPSI001",           // Codice Mago gestionale
  unitaMisura: "PCS",          // Unità di misura base
  onOff: "1",                  // Attivo/Disattivo
  brand: "Pepsi",              // Marca
  pack: "33CL",                // Formato confezione
  materiale: "Vetro"           // Materiale/Tipo
}
```

Fallback Locale: data/prodotti.json

- **Scopo:** Backup quando API Google non disponibile
- **Sync:** Manuale, da aggiornare periodicamente
- **Formato:** Semplificato rispetto a schema completo

5.4 Formato File TXT Generati

Standard Movimentazioni

```
CODICE_MAGO ; QUANTITA_15_DECIMALI
```

Naming Convention File TXT

Fatture Consegnate:

```
NumeroDocumento_DataEmissione_NomeFornitore_CodicePuntoVendita.txt
```

Esempio: `(FAT001_2025-01-15_FornitoreSRL_120.txt)`

Movimentazioni:

```
MOV_YYYY-MM-DD-CODICE_ORIGINE-CODICE_DESTINAZIONE-PROGRESSIVE.txt
```

Esempio: `(MOV_2025-01-15-120-107-001.txt)`

Contenuto Tipo File TXT

```
PEPSI001 ; 24.00000000000000  
PIZZA001 ; 12.00000000000000  
LATTE001 ; 5.50000000000000  
CAFFE001 ; 100.00000000000000
```

6. Funzionalità Business

6.1 Processo Conferma Consegne

Workflow Utente

1. **Login:** Accesso con credenziali punto vendita
2. **Visualizzazione:** Lista fatture pending per il negozio
3. **Selezione:** Scelta fattura da confermare
4. **Input Data:** Inserimento data consegna effettiva

5. Conferma: Due opzioni disponibili

- **Consegna OK:** Conferma standard
- **Consegna con Errori:** Aggiunta note problemi

6. Aggiornamento: Stato cambia a "consegnato"

7. Generazione: Automatica creazione file TXT

Business Logic Validazioni

javascript

// Validazioni data consegna

- Non può essere **futura** (> oggi)
- Deve essere **in** formato **YYYY-MM-DD**
- Deve essere **>=** data emissione fattura

// Validazioni permessi

- Utente può modificare solo fatture del proprio punto vendita
- Admin può modificare tutte le fatture

// Validazioni note errori

- Obbligatorie se si usa "**Consegna con Errori**"
- Max **500** caratteri
- Sanitizzazione **input** (escape **HTML**)

Alerting e Notifiche

- **Fatture scadute:** Highlight rosso per consegne > 5 giorni
- **Conferma successo:** Alert verde con conferma generazione TXT
- **Errori processo:** Alert rosso con dettagli specifici

6.2 Sistema Movimentazioni

Workflow Completo

1. **Selezione Prodotti:** Ricerca intelligente prodotti da DB
2. **Quantità:** Input numerico con validazione UOM
3. **Destinazione:** Scelta negozio destinatario
4. **Generazione TXT:** Automatica per ogni riga movimento
5. **Editor TXT:** Possibilità modifica contenuto generato
6. **Salvataggio:** Registrazione su Google Sheets
7. **Fatturazione:** Creazione automatica fatture per destinazioni
8. **DDT:** Generazione documento trasporto PDF

9. Download: Scaricamento DDT e salvataggio TXT locale

Algoritmo Ricerca Prodotti

```
javascript

// Campi ricercabili
- nome (descrizione principale)
- codice (codice gestionale)
- brand (marca prodotto)
- pack (formato confezione)
- materiale (tipo/materiale)

// Logica matching
filterOption: (option, rawInput) => {
  const query = rawInput.toLowerCase();
  const haystack = [
    option.label, option.value,
    option.nome, option.brand,
    option.pack, option.materiale,
    option.codice, option.uom
  ].filter(Boolean).join(' ').toLowerCase();

  return haystack.includes(query);
}
```

Generazione DDT PDF

```
javascript

// Metadata DDT
const ddtNumber = `${timestamp_last_4_digits}/${year}`;
const dataFormattata = timestamp.toLocaleDateString('it-IT');

// Struttura PDF
1. Header con logo Fradiavolo
2. Info DDT (numero, data)
3. Dati mittente (negozi origine)
4. Dati destinatario (negozi/i destinazione)
5. Tabella prodotti movimentati
6. Note e osservazioni
7. Firme (mittente, trasportatore, destinatario)
8. Footer generazione automatica
```

6.3 Dashboard Amministrativa

Metriche Principali

```
javascript
```

```
stats: {  
    invoices: {  
        total: number,          // Totale fatture sistema  
        consegnate: number,     // Fatture confermate  
        pending: number,        // In attesa conferma  
        byStore: {},            // Breakdown per negozio  
        byStatus: {},            // Breakdown per stato  
        recentActivity: []      // Ultime 10 attività  
    },  
    movimentazioni: {  
        total: number,          // Totale movimenti  
        thisMonth: number,       // Movimenti mese corrente  
        byStore: {},            // Breakdown per negozio  
        recentActivity: []      // Ultimi 10 movimenti  
    },  
    activeStores: [],          // Negozi con attività  
    dateRange: {  
        firstInvoice: date,    // Prima fattura sistema  
        lastActivity: date     // Ultima attività  
    }  
}
```

Funzionalità Filtri Admin

- **Per Negozio:** Visualizzazione dati specifico punto vendita
- **Per Periodo:** Range date personalizzabile
- **Per Stato:** Filtro stato fatture/movimenti
- **Per Fornitore:** Filtro specifico fornitore

Export Dati

```
javascript
```

```

// Formati supportati
- JSON: Export completo con metadata
- CSV: Export tabellare (future implementation)

// Tipi export
- Solo fatture
- Solo movimentazioni
- Dati completi sistema

// Metadata export
{
  exportDate: "2025-01-15T10:30:00.000Z",
  exportedBy: "admin@fradiavolo.com",
  data: { invoices: [...], movimentazioni: [...] }
}

```

6.4 File TXT Manager

Funzionalità Principali

- **Lista File**: Visualizzazione tutti i TXT generati
- **Filtri Avanzati**: Per data, negozio, tipo, fornitore
- **Preview**: Anteprima contenuto file
- **Edit**: Modifica contenuto con backup automatico
- **Download**: Scaricamento file singoli
- **Merge**: Accorpamento multipli file in uno

Sistema Filtri TXT

```

javascript

// Filtri disponibili
fileTypeFilter: 'all' | 'fatture' | 'movimentazioni'
dateFilter: 'YYYY-MM-DD' | ''
storeCodeFilter: string | ''
supplierFilter: string | ''
documentNumberFilter: string | ''

// Parsing nomi file
// Fatture: NumDoc_Data_Fornitore_CodPV.txt
// Movimentazioni: MOV_Data-Origine-Destinazione.txt

```

Merge Algorithm

javascript

```
const mergeAndDownloadFiles = async () => {
  let allLines = [];

  for (const file of filteredFiles) {
    const content = await fetchFileContent(file.name);
    const validLines = content
      .split('\n')
      .filter(line => line.includes(';'))
      .filter(line => {
        const parts = line.split(';');
        return parts.length === 2 && !isNaN(parseFloat(parts[1].trim()));
      });
    allLines.push(...validLines);
  }

  const finalContent = allLines.join('\n') + '\n';
  downloadAsFile(finalContent, generateMergedFilename());
};
```

7. Sicurezza e Controlli

7.1 Autenticazione Multi-Livello

Validazione Email

javascript

```
const validateEmail = (email) =>
  validator.isEmail(email) &&
  (email.includes('@fradiavolopizzeria.com') || email.includes('@azienda.it'));
```

JWT Token Management

javascript

```

// Token generation
const tokenPayload = {
  userId: user.id,
  email: user.email,
  puntoVendita: user.puntoVendita,
  role: user.role
};
const token = jwt.sign(tokenPayload, process.env.JWT_SECRET, { expiresIn: '8h' });

// Token verification middleware
const authenticateToken = (req, res, next) => {
  const token = (req.headers['authorization'] || '').split(' ')[1];
  if (!token) return res.status(401).json({ error: 'Token richiesto' });

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ error: 'Token non valido' });
    req.user = user;
    next();
  });
};

```

7.2 Rate Limiting e DoS Protection

Configurazione Rate Limits

```

javascript

// General API
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minuti
  max: 100,                // Max 100 richieste per IP
  message: 'Troppe richieste da questo IP'
});

// Login specific
const loginLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minuti
  max: 5,                  // Max 5 tentativi login
  message: 'Troppi tentativi di login. Riprova tra 15 minuti.'
});

```

7.3 Input Validation e Sanitization

Validazioni Backend

```
javascript
```

```
// Date validation
const validateDate = (dateString) =>
  validator.isDate(dateString) && new Date(dateString) <= new Date();

// Input sanitization
const sanitizeInput = (input) => validator.escape(String(input ?? '')).trim();

// Business logic validation
- Email format e dominio
- Date non future per consegne
- Quantità numeriche positive
- Codici prodotto esistenti
- Permessi utente per punto vendita
```

Protezione XSS

```
javascript

// Client-side
- React automatic escaping
- Controlled components
- Input type validation

// Server-side
- validator.escape() su tutti gli input
- Sanitizzazione prima del database
- Content Security Policy headers
```

7.4 Controllo Accessi per Punto Vendita

Middleware Authorization

```
javascript
```

```
const requireStoreAccess = (req, res, next) => {
  const userStore = req.user.puntoVendita;
  const requestedStore = req.params.store || req.body.origine;

  if (req.user.role === 'admin') return next(); // Admin bypassa controlli

  if (userStore !== requestedStore) {
    return res.status(403).json({
      error: 'Accesso negato per questo punto vendita'
    });
  }

  next();
};
```

Data Filtering

javascript

```
// Automatic filtering based on user context
const loadSheetData = async (puntoVendita) => {
  const rows = await sheet.getRows();
  let data = rows.map(/* mapping logic */);

  if (puntoVendita && req.user.role !== 'admin') {
    data = data.filter(r => r.punto_vendita === puntoVendita);
  }

  return data;
};
```

7.5 Security Headers

Helmet Configuration

javascript

```
app.use(helmet({
  contentSecurityPolicy: {
    directives: {
      defaultSrc: ["'self'"],
      styleSrc: ["'self'", "'unsafe-inline'"],
      scriptSrc: ["'self'"],
      imgSrc: ["'self'", "data:", "https:"],
    },
  },
  hsts: {
    maxAge: 31536000,
    includeSubDomains: true,
    preload: true
  }
}));
```

8. Deployment e Configurazione

8.1 Variabili Ambiente

File .env Richiesto

```
bash

# Server Configuration
PORT=3001
NODE_ENV=production

# JWT Security
JWT_SECRET=your_super_secure_jwt_secret_key_here

# Google Sheets Integration
GOOGLE_SHEET_ID=1your_sheet_id_here
GOOGLE_SERVICE_ACCOUNT_EMAIL=service-account@project.iam.gserviceaccount.com
GOOGLE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\\n-----END PRIVATE KEY-----\n"

# Optional API URL Override
REACT_APP_API_URL=https://your-backend-domain.com/api
```

Configurazione Google Service Account

```
json
```

```
{  
  "type": "service_account",  
  "project_id": "claude-dash",  
  "private_key_id": "3830828d7322111a858feae5955df4b11138d9b8",  
  "private_key": "-----BEGIN PRIVATE KEY-----\n...|\n-----END PRIVATE KEY-----\n",  
  "client_email": "invoice-prox@claude-dash.iam.gserviceaccount.com",  
  "client_id": "114665863413036038799",  
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",  
  "token_uri": "https://oauth2.googleapis.com/token",  
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs"  
}
```

8.2 Setup Google Cloud

Passaggi Configurazione

1. Creazione Progetto Google Cloud

- Accesso Google Cloud Console
- Nuovo progetto "fradiavolo-invoice"
- Abilitazione Google Sheets API

2. Service Account Setup

```
bash  
  
gcloud iam service-accounts create invoice-service \  
  --display-name="Invoice Service Account"  
  
gcloud projects add-iam-policy-binding PROJECT_ID \  
  --member="serviceAccount:invoice-service@PROJECT_ID.iam.gserviceaccount.com" \  
  --role="roles/spreadsheets.editor"
```

3. Condivisione Google Sheets

- Condividere sheet con email service account
- Permessi "Editor" richiesti
- Verificare accesso API attivo

8.3 Processo Build e Deploy

Frontend Build

```
bash
```

```
# Build processo React
```

```
npm run build
```

```
# File generati in build/
```

```
build/
  ├── static/
  |   ├── css/
  |   ├── js/
  |   └── media/
  └── index.html
  └── manifest.json
```

Backend Deploy

```
bash
```

```
# Installazione dipendenze
```

```
npm install --production
```

```
# Start processo
```

```
npm start
```

```
# o con PM2 per produzione
```

```
pm2 start server.js --name "fradiavolo-backend"
```

```
pm2 startup
```

```
pm2 save
```

Docker Configuration (Opzionale)

```
dockerfile
```

```
# Dockerfile Backend
```

```
FROM node:16-alpine
```

```
WORKDIR /app
```

```
COPY package*.json ./
```

```
RUN npm ci --production
```

```
COPY ..
```

```
EXPOSE 3001
```

```
CMD ["npm", "start"]
```

```
dockerfile
```

```

# Dockerfile Frontend
FROM node:16-alpine as builder
WORKDIR /app
COPY package*.json .
RUN npm ci
COPY ..
RUN npm run build

FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
EXPOSE 80

```

8.4 Configurazione Reverse Proxy

Nginx Configuration

```

nginx

server {
    listen 80;
    server_name your-domain.com;

    # Frontend
    location / {
        root /var/www/fradiavolo-frontend;
        try_files $uri $uri/ /index.html;
    }

    # Backend API
    location /api/ {
        proxy_pass http://localhost:3001/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # File TXT download
    location /api/txt-files/ {
        proxy_pass http://localhost:3001/api/txt-files/;
        proxy_set_header Content-Type application/octet-stream;
    }
}
```

8.5 SSL/TLS Configuration

Let's Encrypt Setup

```
bash

# Installazione Certbot
sudo apt-get install certbot python3-certbot-nginx

# Generazione certificati
sudo certbot --nginx -d your-domain.com

# Auto-renewal
sudo crontab -e
# Aggiungere: 0 12 * * * /usr/bin/certbot renew --quiet
```

9. Testing e Qualità

9.1 Strategie di Testing

Backend Testing

```
javascript

// Unit Testing con Jest
describe('Invoice Confirmation', () => {
  test('should confirm delivery with valid date', async () => {
    const mockInvoice = { id: '123', stato: 'pending' };
    const result = await confirmDelivery('123', '2025-01-15');
    expect(result.success).toBe(true);
  });

  test('should reject future delivery dates', async () => {
    const futureDate = new Date();
    futureDate.setDate(futureDate.getDate() + 1);

    expect(() => {
      validateDeliveryDate(futureDate.toISOString().split('T')[0]);
    }).toThrow('Data non può essere futura');
  });
});
```

Frontend Testing

```
javascript
```

```
// Component Testing con React Testing Library
import { render, fireEvent, waitFor } from '@testing-library/react';
import App from './App';

test('login form validation', async () => {
  const { getByPlaceholderText, getByText } = render(<App />);

  const emailInput = getByPlaceholderText('utente@azienda.it');
  fireEvent.change(emailInput, { target: { value: 'invalid-email' } });

  const loginButton = getByText('Accedi alla Dashboard');
  fireEvent.click(loginButton);

  await waitFor(() => {
    expect(getByText('Email non valida')).toBeInTheDocument();
  });
});
```

Integration Testing

javascript

```
// API Integration Tests
describe('API Integration', () => {
  test('full invoice confirmation flow', async () => {
    // 1. Login
    const loginResponse = await request(app)
      .post('/api/auth/login')
      .send({ email: 'test@fradiavolopizzeria.com', password: 'test' });

    // 2. Get invoices
    const invoicesResponse = await request(app)
      .get('/api/invoices')
      .set('Authorization', `Bearer ${loginResponse.body.token}`);

    // 3. Confirm delivery
    const confirmResponse = await request(app)
      .post('/api/invoices/123/confirm')
      .set('Authorization', `Bearer ${loginResponse.body.token}`)
      .send({ data_consegna: '2025-01-15' });

    expect(confirmResponse.status).toBe(200);
  });
});
```

9.2 Error Handling Robusto

Backend Error Handling

```
javascript

// Global error handler
app.use((error, req, res, next) => {
  console.error('Errore non gestito:', error);

  // Log structured error
  const errorLog = {
    timestamp: new Date().toISOString(),
    method: req.method,
    url: req.originalUrl,
    user: req.user?.email || 'anonymous',
    error: error.message,
    stack: error.stack
  };

  // In produzione, non esporre stack trace
  if (process.env.NODE_ENV === 'production') {
    res.status(500).json({ error: 'Errore interno del server' });
  } else {
    res.status(500).json({ error: error.message, stack: error.stack });
  }
});
```

Frontend Error Boundaries

```
javascript
```

```

class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  componentDidCatch(error, errorInfo) {
    console.error('React Error Boundary:', error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return (
        <div className="min-h-screen bg-red-50 flex items-center justify-center p-4">
          <div className="bg-white rounded-lg p-8 shadow-lg max-w-md">
            <h2 className="text-xl font-bold text-red-600 mb-4">
              Ops! Qualcosa è andato storto
            </h2>
            <p className="text-gray-600 mb-4">
              Si è verificato un errore inaspettato. Ricarica la pagina per continuare.
            </p>
            <button
              onClick={() => window.location.reload()}
              className="bg-red-600 text-white px-4 py-2 rounded-lg hover:bg-red-700"
            >
              Ricarica Pagina
            </button>
          </div>
        </div>
      );
    }

    return this.props.children;
  }
}

```

9.3 Monitoring e Logging

Structured Logging

javascript

```
// Winston logger configuration
const winston = require('winston');

const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  transports: [
    new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),
    new winston.transports.File({ filename: 'logs/combined.log' }),
    new winston.transports.Console({
      format: winston.format.simple()
    })
  ]
});

});
```

Performance Monitoring

javascript

```
// Request timing middleware
const requestTimer = (req, res, next) => {
  const start = Date.now();

  res.on('finish', () => {
    const duration = Date.now() - start;
    logger.info('Request completed', {
      method: req.method,
      url: req.originalUrl,
      status: res.statusCode,
      duration: `${duration}ms`,
      user: req.user?.email
    });
  });

  // Alert per richieste lente
  if (duration > 5000) {
    logger.warn('Slow request detected', { url: req.originalUrl, duration });
  }
};

next();
};
```

9.4 Performance Optimization

Backend Optimizations

```
javascript
```

```
// Caching Google Sheets responses
const NodeCache = require('node-cache');
const cache = new NodeCache({ stdTTL: 300 }); // 5 minuti cache

const getCachedSheetData = async (sheetName) => {
  const cacheKey = `sheet_${sheetName}`;
  let data = cache.get(cacheKey);

  if (!data) {
    data = await loadSheetDataFromGoogle(sheetName);
    cache.set(cacheKey, data);
  }

  return data;
};

// Compression middleware
const compression = require('compression');
app.use(compression());

// JSON parsing limits
app.use(express.json({ limit: '10mb' }));
```

Frontend Optimizations

```
javascript
```

```

// React.lazy per code splitting
const AdminDashboard = React.lazy(() => import('./components/AdminDashboard'));
const TxtFilesManager = React.lazy(() => import('./TxtFilesManager'));

// Suspense wrapper
<Suspense fallback=<LoadingSpinner />>
  <AdminDashboard />
</Suspense>

// useMemo per expensive computations
const filteredInvoices = useMemo(() => {
  return invoices.filter(inv => inv.stato === activeTab);
}, [invoices, activeTab]);

// useCallback per event handlers
const handleConfirmDelivery = useCallback((invoiceld, date) => {
  confirmDelivery(invoiceld, date);
}, []);

```

10. Documentazione Utente

10.1 Guida Operatori Negozi

Login e Accesso

- 1. URL Sistema:** <https://your-domain.com>
- 2. Credenziali:** Fornite dall'amministratore centrale
 - Email: negoziocittà@fradiavolopizzeria.com
 - Password: Personalizzata per negozio
- 3. Primo Accesso:** Verificare funzionalità con fattura test

Gestione Consegne

PROCESSO STANDARD:

1. Accedere al tab "Da confermare"
2. Verificare lista fatture in attesa
3. Per ogni consegna effettuata:
 - Inserire data consegna effettiva
 - Scegliere "Conferma" se tutto OK
 - Scegliere "Con Errori" se problemi
4. Il sistema genera automaticamente file TXT
5. Verificare cambio stato in "Confermate"

ALERT FATTURE SCADUTE:

- Evidenziate in rosso con 
- Priorità assoluta per conferma
- Contattare amministrazione se problemi

Gestione Movimentazioni

PROCESSO TRASFERIMENTI:

1. Accedere al tab "Movimentazione"
2. Selezionare prodotti da trasferire
3. Inserire quantità precise
4. Scegliere negozio destinazione
5. Verificare contenuto TXT generato
6. Cliccare "Genera DDT, Scarica e Salva TXT"
7. Sistema crea automaticamente:
 - File PDF del DDT
 - File TXT per gestionale
 - Fattura automatica per destinazione

RICERCA PRODOTTI:

- Digitare parte del nome
- Ricerca intelligente su: nome, marca, codice
- Verificare unità di misura corretta

10.2 Manuale Amministratore

Dashboard Amministrativa

ACCESSO ADMIN:

- Login con credenziali amministratore
- Vista sidebar con sezioni dedicate
- Accesso a tutti i dati del sistema

METRICHE PRINCIPALI:

- Fatture totali/confermate/pending
- Movimentazioni per negozio e periodo
- Attività recente sistema-wide
- Negozi attivi e loro performance

FUNZIONI ADMIN:

- ✓ Vista globale tutte le fatture
- ✓ Modifica qualsiasi consegna
- ✓ Gestione utenti negozi
- ✓ Export dati completi
- ✓ Supervisione movimentazioni
- ✓ Accesso file TXT generati

Gestione File TXT

ACCESSO FILE MANAGER:

1. Sidebar Admin → "File TXT"
2. Filtri per data/negozio/fornitore
3. Preview e modifica contenuti
4. Download singoli o accorpati
5. Backup automatici delle modifiche

FILTRI AVANZATI:

- Tipo: Fatture vs Movimentazioni
- Data: Selezione specifica
- Negozio: Codice punto vendita
- Fornitore: Nome fornitore
- Numero documento: Per fatture

FUNZIONE ACCORPAMENTO:

- Selezionare filtri desiderati
- Cliccare "Accorpa e Scarica"
- Sistema unisce tutti i file filtrati
- Download unico file consolidato

10.3 Troubleshooting Comune

Problemi Login

ERRORE: "Email non valida"

SOLUZIONE: Verificare formato email @fradiavolopizzeria.com

ERRORE: "Credenziali non valide"

SOLUZIONE:

1. Verificare email e password corrette
2. Contattare amministrazione per reset
3. Controllare caps lock attivo

ERRORE: "Troppi tentativi di login"

SOLUZIONE: Attendere 15 minuti prima di riprovare

Problemi Consegne

ERRORE: "Data non valida"

SOLUZIONE:

1. Data non può essere futura
2. Usare formato corretto (selettore calendario)
3. Data deve essere successiva a emissione fattura

ERRORE: "Impossibile confermare consegna"

SOLUZIONE:

1. Verificare connessione internet
2. Ricaricare la pagina
3. Controllare se fattura già confermata
4. Contattare supporto tecnico

FILE TXT NON GENERATO:

CAUSE POSSIBILI:

- Contenuto TXT vuoto nella fattura originale
- Errore temporaneo di sistema
- Problema connessione Google Sheets

SOLUZIONE: Contattare amministrazione

Problemi Movimentazioni

ERRORE: "Prodotto non trovato"

SOLUZIONE:

1. Verificare spelling nome prodotto
2. Provare ricerca per codice
3. Utilizzare pulsante "Ricarica Prodotti"
4. Segnalare prodotto mancante ad admin

ERRORE: "Quantità non valida"

SOLUZIONE:

1. Inserire solo numeri positivi
2. Usare punto (.) per decimali
3. Verificare unità di misura corretta

DDT NON SCARICATO:

CAUSE:

- Popup bloccati dal browser
- Errore temporaneo generazione PDF

SOLUZIONE:

1. Abilitare popup per il sito
2. Riprovare operazione
3. Verificare spazio disco disponibile

Problemi Performance

SISTEMA LENTO:

CAUSE POSSIBILI:

- Connessione internet limitata
- Carico server elevato
- Cache browser piena

SOLUZIONI:

1. Ricaricare pagina (F5)
2. Svuotare cache browser
3. Verificare connessione internet
4. Provare in orari diversi
5. Contattare supporto se persiste

ERRORE "Impossibile caricare dati":

SOLUZIONI:

1. Cliccare pulsante "Aggiorna"
2. Verificare login ancora valido
3. Controllare connessione Google Sheets
4. Rifare login se necessario

11. Roadmap e Sviluppi Futuri

11.1 Miglioramenti Prioritari

Performance e Scalabilità

Q2 2025:

- Implementazione Redis per caching
- Database PostgreSQL per dati critici
- WebSocket per aggiornamenti real-time
- CDN per assets statici

Q3 2025:

- Microservizi architecture
- Load balancing
- Database sharding
- Message queue system

Nuove Funzionalità Business

IMMEDIATE (Q1 2025):

- Notifiche email automatiche consegne
- App mobile per operatori
- Integrazione WhatsApp alerts
- Dashboard analytics avanzate

MEDIUM TERM (Q2-Q3 2025):

- Integrazione diretta ERP/gestionale
- API per fornitori esterni
- Modulo ordini automatici
- Previsioni demand planning

LONG TERM (Q4 2025+):

- AI per ottimizzazione movimentazioni
- Blockchain per supply chain
- IoT integration magazzini
- Predictive analytics

11.2 Refactoring Tecnico

Backend Modernization

javascript

```

// Current: Monolithic Express
// Future: Microservices with NestJS

// Auth Service
@Injectable()
export class AuthService {
  async validateUser(email: string, password: string): Promise<User> {
    // JWT + Redis session management
  }
}

// Invoice Service
@Injectable()
export class InvoiceService {
  async confirmDelivery(invoiceId: string, data: ConfirmDeliveryDto): Promise<Invoice> {
    // Event-driven architecture
    this.eventEmitter.emit('delivery.confirmed', { invoiceId, data });
  }
}

// TXT Generation Service
@Injectable()
export class TxtGeneratorService {
  @OnEvent('delivery.confirmed')
  async generateTxtFile(payload: DeliveryConfirmedEvent): Promise<void> {
    // Async file generation with queue
  }
}

```

Database Migration Strategy

sql

```
-- Phase 1: PostgreSQL schema
CREATE TABLE punti_vendita (
    id SERIAL PRIMARY KEY,
    codice VARCHAR(10) UNIQUE NOT NULL,
    nome VARCHAR(255) NOT NULL,
    indirizzo TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE fatture (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    numero VARCHAR(50) NOT NULL,
    fornitore VARCHAR(255) NOT NULL,
    data_emissione DATE NOT NULL,
    data_consegna DATE,
    stato_fattura_stato NOT NULL DEFAULT 'pending',
    punto_vendita_id INTEGER REFERENCES punti_vendita(id),
    confermato_da VARCHAR(255),
    txt_content TEXT,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

-- Phase 2: Data migration from Google Sheets
-- Phase 3: Hybrid operation (read from both)
-- Phase 4: Complete cutover to PostgreSQL
```

11.3 Security Enhancements

Zero Trust Architecture

IMPLEMENTED:

- Multi-factor authentication
- Role-based access control (RBAC)
- API request signing
- Encrypted data at rest
- WAF (Web Application Firewall)

PLANNED:

- Single Sign-On (SSO) integration
- Behavioral analytics
- Automated threat detection
- Compliance auditing (GDPR, SOX)
- Penetration testing quarterly

11.4 Integrazione Ecosistema

ERP Integration

```
javascript

// Connector pattern per diversi ERP
interface ERPConnector {
    async syncInvoices(invoices: Invoice[]): Promise<SyncResult>;
    async syncProducts(products: Product[]): Promise<SyncResult>;
    async exportMovements(movements: Movement[]): Promise<ExportResult>;
}

class SAPConnector implements ERPConnector {
    // SAP-specific implementation
}

class OracleConnector implements ERPConnector {
    // Oracle-specific implementation
}

class MagoConnector implements ERPConnector {
    // Mago-specific implementation (current TXT files)
}
```

External APIs

PARTNER INTEGRATIONS:

- Fornitori: API per ordini automatici
- Corrieri: Tracking spedizioni real-time
- Banche: Riconciliazione pagamenti
- Assicurazioni: Claims automatici
- Utilities: Monitoraggio consumi energia

THIRD-PARTY SERVICES:

- Twilio: SMS notifications
- SendGrid: Email transazionali
- Stripe: Pagamenti online
- AWS S3: Document storage
- Slack: Team notifications

12. Conclusioni e Best Practices

12.1 Architettura Lessons Learned

Decisioni Architetturali Vincenti

GOOGLE SHEETS COME MVP DATABASE

Vantaggi realizzati:

- Time-to-market ridotto (settimane vs mesi)
- Stakeholder visibility immediata
- Zero database administration
- Backup automatico
- Collaboration nativa

JWT STATELESS AUTHENTICATION

Vantaggi:

- Scalabilità orizzontale semplificata
- No session storage requirements
- Mobile-friendly
- Microservices ready

REACT SPA ARCHITECTURE

Vantaggi:

- User experience fluida
- Offline capabilities potential
- Component reusability
- Modern developer experience

Limitazioni Identificate da Mitigare

⚠ GOOGLE SHEETS PERFORMANCE

Problemi:

- Latenza 500ms-2s per operazioni
- Rate limits API (100 requests/100s)
- Concurrent access limitations

Mitigazioni implementate:

- Client-side caching aggressive
- Batch operations where possible
- Graceful degradation patterns

⚠ FILE TXT STORAGE LOCALE

Problemi:

- Single point of failure
- No distributed storage
- Manual backup process

Mitigazioni planned:

- AWS S3 migration
- Automated backup
- CDN distribution

12.2 Development Best Practices

Code Quality Standards

javascript

```
// Error handling pattern
const handleAsyncOperation = async (operation) => {
  try {
    setLoading(true);
    setError('');

    const result = await operation();

    setSuccess('Operazione completata con successo!');
    return result;
  } catch (error) {
    console.error('Errore operazione:', error);
    setError(`Errore: ${error.message}`);
    throw error;
  } finally {
    setLoading(false);
    setTimeout(() => {
      setSuccess('');
      setError('');
    }, 5000);
  }
};
```

```
// Input validation pattern
const validateAndSanitize = (input, rules) => {
  const sanitized = validator.escape(input.trim());

  rules.forEach(rule => {
    if (!rule.validator(sanitized)) {
      throw new Error(rule.message);
    }
  });
}

return sanitized;
};
```

API Design Patterns

javascript

```

// Consistent response format
const apiResponse = (success, data = null, error = null) => ({
  success,
  timestamp: new Date().toISOString(),
  data,
  error,
  ...(process.env.NODE_ENV === 'development' && { debug: getDebugInfo() })
});

// Pagination pattern
const paginatedResponse = (items, page, perPage, total) => ({
  success: true,
  data: items,
  pagination: {
    page,
    perPage,
    total,
    totalPages: Math.ceil(total / perPage),
    hasNext: (page * perPage) < total,
    hasPrev: page > 1
  }
});

```

12.3 Operational Excellence

Monitoring Dashboard

KEY METRICS:

- Response time medio API: < 500ms
- Uptime sistema: > 99.9%
- Error rate: < 1%
- User satisfaction: > 95%

ALERTS:

- Error rate > 5%: Slack notification immediate
- Response time > 2s: Email to dev team
- Disk space < 20%: System alert
- Failed logins > 10/min: Security alert

LOG RETENTION:

- Application logs: 90 days
- Security logs: 1 year
- Audit trails: 7 years
- Performance metrics: 30 days

Backup Strategy

AUTOMATED BACKUPS:

Daily:

- Google Sheets export (JSON + CSV)
- Generated TXT files archive
- Application logs

Weekly:

- Complete system configuration
- Database schemas export
- Dependencies audit

Monthly:

- Full disaster recovery test
- Security penetration scan
- Performance baseline review

12.4 Success Metrics

Business KPIs

EFFICIENCY GAINS:

- Riduzione tempo conferma consegne: 70%
- Eliminazione errori trascrizione: 95%
- Aumento visibilità movimentazioni: 100%
- Riduzione documentazione cartacea: 80%

USER ADOPTION:

- Utilizzo quotidiano sistema: 26/26 negozi
- Soddisfazione utenti: 4.5/5 media
- Tempo training nuovo utente: < 30 minuti
- Supporto richiesto: < 1 ticket/settimana

TECHNICAL KPIs:

- Mean Time Between Failures: > 720 ore
- Mean Time To Recovery: < 15 minuti
- Code coverage tests: > 80%
- Security vulnerabilities: 0 critiche

12.5 Raccomandazioni Finali

Per Team di Sviluppo

1. Mantenere Focus MVP: Resist over-engineering, priorità funzionalità business

- 2. Testing Strategy:** Aumentare coverage con integration tests
- 3. Documentation:** Mantenere docs aggiornate con ogni release
- 4. Security First:** Regular security audits e dependency updates

Per Business Stakeholder

- 1. Change Management:** Training continuo utenti su nuove features
- 2. Feedback Loop:** Weekly feedback sessions con operatori negozi
- 3. Scalability Planning:** Budget for infrastructure scaling
- 4. Compliance:** Ensure GDPR e data protection compliance

Per Operations Team

- 1. Monitoring:** Implement comprehensive observability stack
 - 2. Disaster Recovery:** Regular DR drills e backup testing
 - 3. Performance:** Continuous optimization based on usage patterns
 - 4. Security:** Automated vulnerability scanning e patch management
-

Documento compilato: Gennaio 2025

Versione: 1.0

Prossima revisione: Aprile 2025