Novi - Technical Documentation

Novi
Smart Commerce Suite

Generated on: 28/06/2025

Novi Technical Documentation

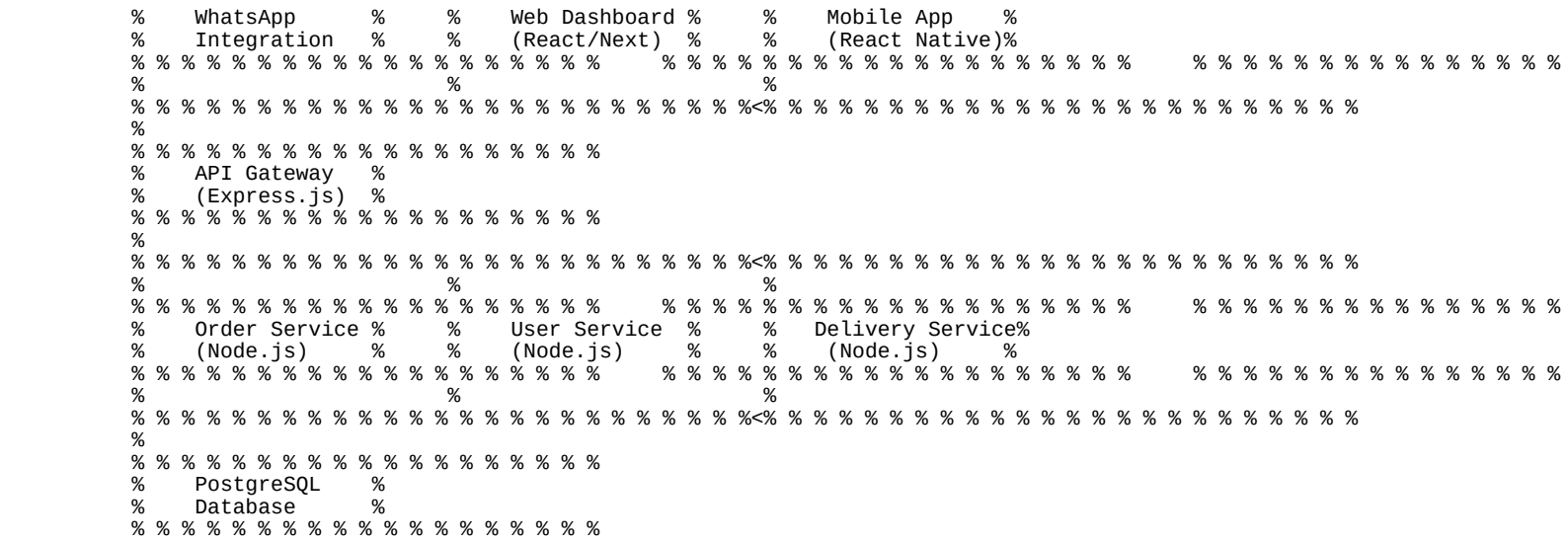
Technical architecture and implementation guide for Novi Smart Commerce Suite

Ø<ß×þ System Architecture

Overview

Novi is built as a modern, scalable web application designed to handle WhatsApp-native commerce operations. The system architecture follows microservices principles with a focus on reliability, performance, and ease of maintenance.

Core Components



Technology Stack

Frontend

- Framework: React.js with Next.js
- Styling: Tailwind CSS + Custom Components
- State Management: React Context + Hooks
- Charts: Chart.js for analytics

- UI Components: Custom design system

Backend

- Runtime: Node.js
- Framework: Express.js
- Database: PostgreSQL with Knex.js ORM
- Authentication: JWT tokens
- File Storage: Local filesystem (configurable for cloud)

WhatsApp Integration

- Library: WhatsApp Web.js
- Protocol: WebSocket connections
- Message Processing: Custom NLP pipeline
- Media Handling: Image and document support

Infrastructure

- Hosting: Railway (primary), Heroku (backup)
- Process Management: PM2
- Monitoring: Custom logging + error tracking
- Backup: Automated database backups

Ø=Ý' Development Setup

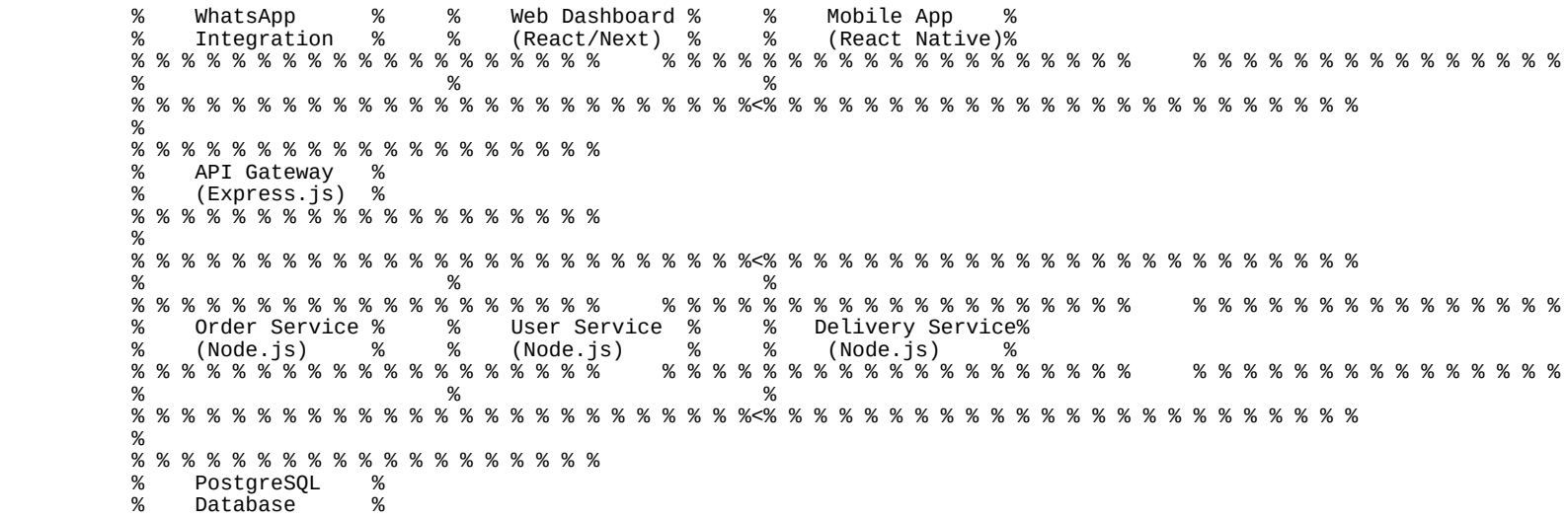
Prerequisites

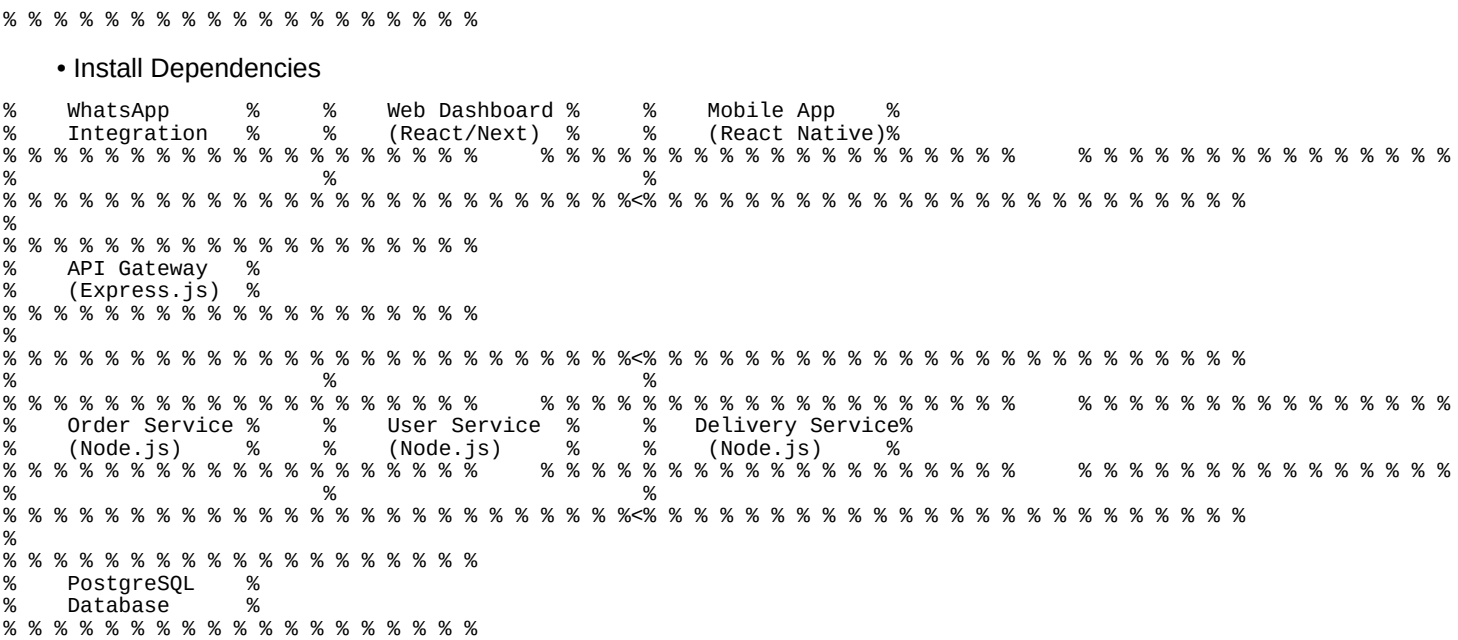
- Node.js 18+
- PostgreSQL 14+
- Git
- WhatsApp Business API access (optional for development)

Environment Setup

- Clone Repository

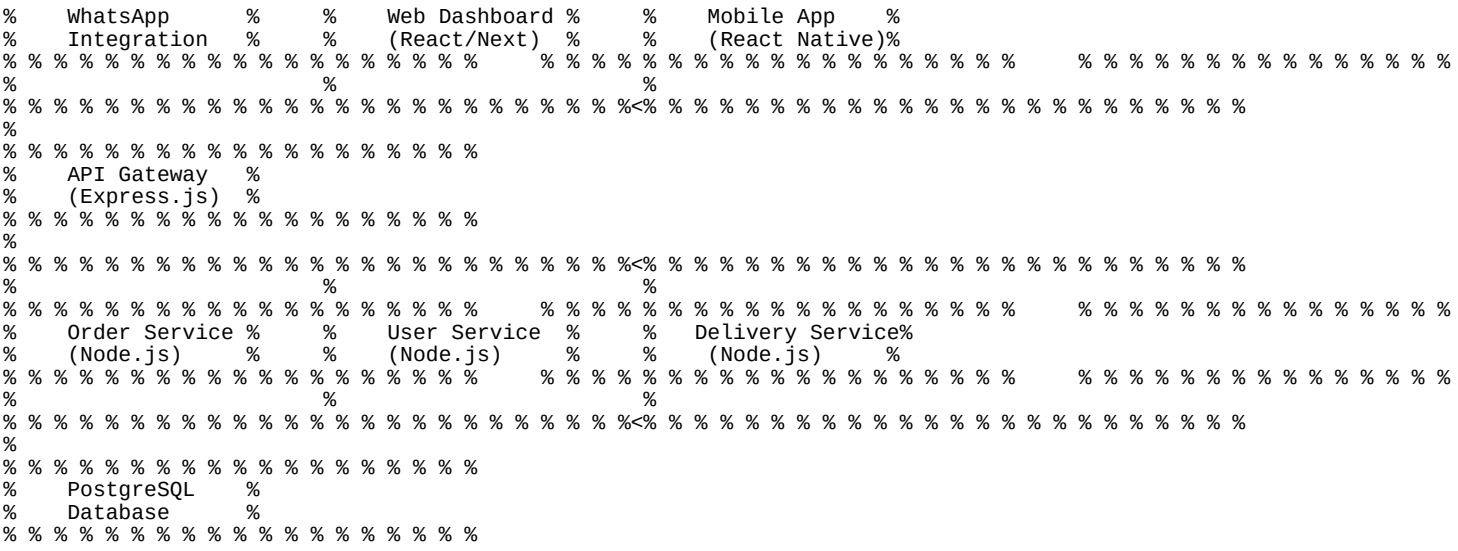
cd novi-commerce





• Install Dependencies

• Environment Configuration



Configure the following variables:

DATABASE_URL=postgresql://username:password@localhost:5432/novi_db

JWT Secret

JWT_SECRET=your-super-secret-jwt-key

WhatsApp Configuration

WHATSAPP_SESSION_PATH=./sessions

WHATSAPP_QR_TIMEOUT=60000

Server Configuration

PORT=3000

NODE_ENV=development

File Storage

UPLOAD_PATH=./uploads

MAX_FILE_SIZE=10485760

```
%    WhatsApp      %     %    Web Dashboard %     %    Mobile App    %
%    Integration   %     %    (React/Next)  %     %    (React Native)%
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    API Gateway   %
%    (Express.js)  %
% % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%    Order Service %     %    User Service  %     %   Delivery Service%
%    (Node.js)     %     %    (Node.js)     %     %    (Node.js)     %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    PostgreSQL    %
%    Database      %
% % % % % % % % % % % % % % % % % % %
```

- Database Setup

npm run migrate

Seed initial data

npm run seed

```
%    WhatsApp      %     %    Web Dashboard %     %    Mobile App    %
%    Integration   %     %    (React/Next)  %     %    (React Native)%
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    API Gateway   %
%    (Express.js)  %
% % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%    Order Service %     %    User Service  %     %   Delivery Service%
%    (Node.js)     %     %    (Node.js)     %     %    (Node.js)     %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    PostgreSQL    %
%    Database      %
% % % % % % % % % % % % % % % % % % %
```

- Start Development Server

```
%    WhatsApp      %     %    Web Dashboard %     %    Mobile App    %
%    Integration   %     %    (React/Next)  %     %    (React Native)%
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    API Gateway   %
%    (Express.js)  %
% % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%    Order Service %     %    User Service  %     %   Delivery Service%
%    (Node.js)     %     %    (Node.js)     %     %    (Node.js)     %
% % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % % % % % %     % % % % % % % % % % % % % % %
%                        %                        %
% % % % % % % % % % % % % % % % % % % % % % % % %<% % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % %
%    PostgreSQL    %
```

```
%    Database      %
% % % % % % % % % % % % % % % % % % %
```

Development Workflow

Code Structure

```
% % %  src/
%    % % %  config/          # Configuration files
%    % % %  database/        # Database migrations and seeds
%    % % %  services/        # Business logic services
%    % % %  utils/           # Utility functions
%    % % %  views/           # EJS templates
% % %  public/             # Static assets
% % %  scripts/            # Build and deployment scripts
% % %  docs/               # Documentation
% % %  tests/              # Test files
```

Git Workflow

- Create feature branch from main
- Implement changes with tests
- Run linting and tests
- Create pull request
- Code review and merge

Ø=Üñ WhatsApp Integration

WhatsApp Web.js Setup

Novi uses WhatsApp Web.js for WhatsApp Business integration. The implementation includes:

Connection Management

const { Client, LocalAuth } = require('whatsapp-web.js');

class WhatsAppService {

constructor() {

this.client = new Client({

authStrategy: new LocalAuth({

clientId: 'novi-commerce',

dataPath: process.env.WHATSAPP_SESSION_PATH

}),

puppeteer: {

headless: true,

args: ['--no-sandbox', '--disable-setuid-sandbox']

}

});

}

}

```
% % %  src/
%    % % %  config/          # Configuration files
%    % % %  database/        # Database migrations and seeds
%    % % %  services/        # Business logic services
%    % % %  utils/           # Utility functions
%    % % %  views/           # EJS templates
% % %  public/              # Static assets
% % %  scripts/             # Build and deployment scripts
% % %  docs/                # Documentation
% % %  tests/               # Test files
```

Message Processing Pipeline

- Message Reception: WhatsApp Web.js receives messages
- NLP Processing: Custom parser extracts order details
- Order Creation: Structured data saved to database
- Notification: Dashboard updated in real-time
- Response: Automated confirmation sent to customer

Order Parsing Logic

class OrderParser {

parseOrder(message) {

return {

customer: this.extractCustomer(message),

items: this.extractItems(message),

address: this.extractAddress(message),

total: this.calculateTotal(message),

notes: this.extractNotes(message)

};

}

}


```
% % %  src/
%    % % %  config/          # Configuration files
%    % % %  database/        # Database migrations and seeds
%    % % %  services/        # Business logic services
%    % % %  utils/           # Utility functions
%    % % %  views/           # EJS templates
% % %  public/              # Static assets
% % %  scripts/             # Build and deployment scripts
% % %  docs/                # Documentation
% % %  tests/               # Test files
```


Message Templates

Order Confirmation

```
Hi {customer_name},

Your order has been received and is being processed.

Order Details:
{order_items}

Total: ${total}
Estimated Delivery: {delivery_time}
```

```
Track your order: {tracking_link}

Thank you for choosing Novi! Ø=Þ€
```

Delivery Update

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Error Handling

Connection Issues

- Automatic reconnection attempts
- Session persistence across restarts
- Fallback to manual mode
- Admin notifications for critical failures

Message Processing Errors

- Graceful degradation for unparseable messages
- Manual order creation fallback
- Error logging and monitoring
- Customer notification for processing delays

Ø=ÝÄþ Database Design

Schema Overview

Core Tables

Users Table

id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password_hash VARCHAR(255) NOT NULL,

role VARCHAR(20) DEFAULT 'user',

is_active BOOLEAN DEFAULT true,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

```
Hi {customer_name},
```

```
Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Businesses Table

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

owner_id INTEGER REFERENCES users(id),

short_code VARCHAR(10) UNIQUE NOT NULL,

settings JSONB DEFAULT '{}',

is_active BOOLEAN DEFAULT true,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Orders Table

id SERIAL PRIMARY KEY,

business_id INTEGER REFERENCES businesses(id),

customer_name VARCHAR(100) NOT NULL,

customer_phone VARCHAR(20) NOT NULL,

customer_address TEXT,

items JSONB NOT NULL,

total_amount DECIMAL(10,2) NOT NULL,

status VARCHAR(20) DEFAULT 'pending',

notes TEXT,

response_times JSONB DEFAULT '{}',

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_by INTEGER REFERENCES users(id)

);

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Groups Table

id SERIAL PRIMARY KEY,

business_id INTEGER REFERENCES businesses(id),

name VARCHAR(100) NOT NULL,

whatsapp_group_id VARCHAR(100) UNIQUE,

group_type VARCHAR(20) DEFAULT 'orders',

settings JSONB DEFAULT '{}',

is_active BOOLEAN DEFAULT true,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Indexes and Performance

Primary Indexes

CREATE INDEX idx_orders_business_id ON orders(business_id);

CREATE INDEX idx_orders_status ON orders(status);

CREATE INDEX idx_orders_created_at ON orders(created_at);

CREATE INDEX idx_orders_customer_phone ON orders(customer_phone);

-- Groups table indexes

CREATE INDEX idx_groups_business_id ON groups(business_id);

CREATE INDEX idx_groups_whatsapp_id ON groups(whatsapp_group_id);

-- Users table indexes

CREATE INDEX idx_users_email ON users(email);

CREATE INDEX idx_users_role ON users(role);

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Query Optimization

- Composite indexes for common query patterns
- Partial indexes for active records
- Regular VACUUM and ANALYZE maintenance
- Connection pooling for high concurrency

Data Migration Strategy

Version Control

- Knex.js migrations for schema changes
- Rollback support for failed migrations
- Data validation before deployment
- Backup creation before major changes

Migration Process

- Create migration file
- Test in development environment
- Validate data integrity
- Deploy to staging
- Monitor performance impact
- Deploy to production

Ø=Ý Security Implementation

Authentication & Authorization

JWT Token Management

const jwt = require('jsonwebtoken');

class AuthService {

generateToken(user) {

return jwt.sign(

{

id: user.id,

role: user.role,

business_id: user.business_id

},

process.env.JWT_SECRET,

{ expiresIn: '24h' }

);

}

verifyToken(token) {

try {

return jwt.verify(token, process.env.JWT_SECRET);

} catch (error) {

throw new Error('Invalid token');

}

}

}

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Role-Based Access Control

const authorize = (roles) => {

return (req, res, next) => {

if (!req.user) {

return res.status(401).json({ error: 'Unauthorized' });

}

if (!roles.includes(req.user.role)) {

return res.status(403).json({ error: 'Insufficient permissions' });

}

next();

};

};

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Data Protection

Input Validation

- Sanitize all user inputs
- Validate data types and formats
- Prevent SQL injection attacks
- Rate limiting on API endpoints

File Upload Security

- File type validation
- Size limits enforcement
- Virus scanning (optional)
- Secure file storage paths

WhatsApp Security

Session Management

- Encrypted session storage
- Secure QR code handling
- Session timeout and cleanup
- Multi-device session handling

Message Privacy

- End-to-end encryption (WhatsApp native)
- Secure message storage
- Access control for sensitive data
- Audit logging for compliance

Ø=ÜÊ API Documentation

Authentication Endpoints

POST /api/auth/login

{

"email": "user@example.com",

"password": "securepassword"

}

// Response

{

"success": true,

"token": "jwt_token_here",

"user": {

"id": 1,

"email": "user@example.com",

"role": "admin",

"business_id": 1

}

}

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

POST /api/auth/register

{

"username": "newuser",

"email": "newuser@example.com",

"password": "securepassword",

"business_name": "My Business"

}

// Response

{

"success": true,

"message": "User registered successfully",

"user_id": 2

}


```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}
```

```
Questions? Reply to this message.
```

Order Management Endpoints

GET /api/orders

{

"business_id": 1,

"status": "pending",

"page": 1,

"limit": 20,

"date_from": "2024-01-01",

"date_to": "2024-01-31"

}

// Response

{

"success": true,

"orders": [...],

"pagination": {

"page": 1,

"limit": 20,

"total": 150,

"pages": 8

}

}

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

POST /api/orders

{

"business_id": 1,

"customer_name": "John Doe",

"customer_phone": "+1234567890",

"customer_address": "123 Main St",

"items": [

{

"name": "Pizza Margherita",

"quantity": 2,

"price": 15.99

}

],

"total_amount": 31.98,

"notes": "Extra cheese please"

}

// Response

{

"success": true,

"order": {

"id": 123,

"status": "pending",

"created_at": "2024-01-15T10:30:00Z"

}

}

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Business Management Endpoints

GET /api/businesses/:id

{

"success": true,

"business": {

"id": 1,

"name": "Pizza Palace",

"description": "Best pizza in town",

"short_code": "PIZZA001",

"settings": {

"auto_confirm": true,

"delivery_fee": 5.00

},

"stats": {

"total_orders": 1250,

"total_revenue": 18750.00,

"active_customers": 89

}

}

}

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Ø=Þ€ Deployment

Production Environment

Railway Deployment

- Connect Repository
- Link GitHub repository to Railway
- Configure build settings
- Set environment variables
- Database Setup
- Provision PostgreSQL database
- Run migrations automatically
- Configure connection pooling
- Environment Configuration

PORT=3000

DATABASE_URL=postgresql://...

JWT_SECRET=production_secret

WHATSAPP_SESSION_PATH=/app/sessions

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Process Management

module.exports = {

apps: [{

name: 'novi-commerce',

script: 'src/server.js',

instances: 'max',

exec_mode: 'cluster',

env: {

NODE_ENV: 'production'

},

error_file: './logs/err.log',

out_file: './logs/out.log',

log_file: './logs/combined.log',

time: true

}]

};


```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```


Monitoring & Logging

Application Monitoring

- PM2 process monitoring
- Custom health check endpoints
- Performance metrics collection
- Error tracking and alerting

Log Management

```
const winston = require('winston');

const logger = winston.createLogger({

level: 'info',

format: winston.format.combine(

winston.format.timestamp(),

winston.format.json()

),

transports: [

new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),

new winston.transports.File({ filename: 'logs/combined.log' })

]

});
```

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Backup Strategy

Database Backups

- Automated daily backups
- Point-in-time recovery
- Cross-region backup storage
- Backup verification and testing

File Backups

- Session data backup
- Upload file backup
- Configuration backup
- Disaster recovery procedures

Ø>Ýê Testing Strategy

Unit Testing

Service Layer Tests

```
describe('OrderService', () => {

test('should create order successfully', async () => {
```

```
const orderData = {

customer_name: 'Test Customer',

customer_phone: '+1234567890',

items: [{ name: 'Test Item', quantity: 1, price: 10.00 }],

total_amount: 10.00

};

const order = await OrderService.createOrder(orderData);

expect(order.id).toBeDefined();

expect(order.status).toBe('pending');

});

});
```

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

API Endpoint Tests

```
describe('Orders API', () => {

test('GET /api/orders should return orders list', async () => {

const response = await request(app)

.get('/api/orders')

.set('Authorization', `Bearer ${token}`);

expect(response.status).toBe(200);

expect(response.body.success).toBe(true);

expect(Array.isArray(response.body.orders)).toBe(true);

});

});
```

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Integration Testing

WhatsApp Integration Tests

- Message processing pipeline
- Order creation flow
- Error handling scenarios
- Performance under load

Database Integration Tests

- Migration testing
- Data integrity validation
- Query performance testing
- Connection pool testing

End-to-End Testing

User Journey Tests

- Complete order flow
- Customer registration
- Business setup process
- Admin operations

Performance Testing

- Load testing with realistic data
- Stress testing for peak usage
- Memory usage monitoring
- Response time optimization

Ø=Ý CI/CD Pipeline

GitHub Actions Workflow

name: Deploy to Production

on:

push:

branches: [main]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- uses: actions/setup-node@v2

with:

node-version: '18'

- run: npm ci

- run: npm test

- run: npm run lint

deploy:

needs: test

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Deploy to Railway

uses: railway/deploy@v1

with:

service: novi-commerce

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Deployment Stages

- Development
- Local development environment
- Feature testing and validation
- Code review process
- Staging
- Production-like environment
- Integration testing
- User acceptance testing
- Production
- Live environment deployment
- Monitoring and alerting
- Rollback procedures

Ø=ÜÈ Performance Optimization

Database Optimization

Query Optimization

- Index strategy implementation
- Query plan analysis
- Connection pooling
- Read replicas for scaling

Caching Strategy

```
const Redis = require('ioredis');

class CacheService {

constructor() {

this.redis = new Redis(process.env.REDIS_URL);

}

async get(key) {

return await this.redis.get(key);

}

async set(key, value, ttl = 3600) {

return await this.redis.setex(key, ttl, JSON.stringify(value));

}

}
```

```
Hi {customer_name},

Your order is out for delivery and should arrive in {estimated_time}.

Driver: {driver_name}
Vehicle: {vehicle_info}

Track delivery: {tracking_link}

Questions? Reply to this message.
```

Frontend Optimization

Code Splitting

- Route-based code splitting
- Component lazy loading
- Bundle size optimization
- Tree shaking implementation

Performance Monitoring

- Core Web Vitals tracking
- User experience metrics
- Error tracking and reporting
- Performance budget enforcement

Ø=Ý' Maintenance & Updates

Regular Maintenance

Weekly Tasks

- Database performance review
- Log file rotation and cleanup
- Security patch application
- Backup verification

Monthly Tasks

- Performance optimization review
- Security audit and updates
- Feature usage analysis
- Infrastructure cost review

Update Procedures

Application Updates

- Create feature branch
- Implement changes with tests
- Code review and approval
- Staging environment testing
- Production deployment
- Post-deployment monitoring

Database Updates

- Create migration scripts
- Test in development environment
- Backup production database
- Deploy migration
- Verify data integrity
- Monitor performance impact
