FastAPI/NextJS/Postgres + Prisma Turborepo for Multi-tenant SaaS AI Apps

This template provides a modern, full-stack SaaS application boilerplate for multi-tenant AI apps built with FastAPI and Next.js, designed for rapid development and scalability. It combines the best practices of both Python and TypeScript ecosystems to create a robust, type-safe, and performant application.

Tech Stack: FastAPI • Next.js • TypeScript • Python • PostgreSQL • Prisma • Docker • Turborepo

Key Features

- **Full-Stack Type Safety**: TypeScript for frontend and Python type hints for backend
- **Modern Authentication**: Built-in auth system with FastAPI Auth and Supabase Auth for seamless authentication
- **Database Management**: Prisma ORM for type-safe database operations
- **API Documentation**: Auto-generated OpenAPI documentation

- **Testing Infrastructure**: Comprehensive testing with Pytest and Jest
- **CI/CD Pipeline**: GitHub Actions workflow for automated testing and deployment
- **Monorepo Structure**: Turborepo for efficient workspace management
- **Caching Layer**: Redis integration for improved performance
- **Containerization**: Docker support for consistent development and deployment

Use Cases

- Building scalable SaaS applications
- Creating RESTful APIs with automatic documentation
- Developing type-safe full-stack applications
- Setting up a production-ready development environment
- Implementing modern authentication and authorization
- Managing complex database operations with type safety
- Multi-tenant application with secure data isolation

Technology Stack

Backend

- **FastAPI**: High-performance web framework for building APIs with Python
- **Python 3.11+**: Latest stable Python version with enhanced type hints
- **Pydantic**: Data validation and settings management
- **SQLModel**: SQL database library for Python, with SQLAlchemy core and Pydantic
- **FastAPI Auth**: Authentication and authorization library
- **Loguru**: Advanced logging library with better exception handling and formatting

Frontend

- **Next.js 14**: React framework with App Router and Server Components
- **TypeScript 5**: Static type checking for JavaScript
- **shadcn/ui**: Re-usable components built with Radix UI and Tailwind CSS
- **Tailwind CSS**: Utility-first CSS framework
- **React Query**: Data fetching and caching library
- **Zustand**: State management solution
- **React Hook Form**: Form validation and handling
- **Zod**: Schema validation
- **Winston**: Versatile logging library for Node.js

Database & Caching

- **PostgreSQL 15**: Advanced open-source database (powered by Supabase for managed PostgreSQL and authentication)
- **Prisma**: Type-safe ORM for database operations with built-in connection pooling
- **Redis 7**: In-memory data structure store for caching

DevOps & Infrastructure

- **Docker**: Containerization platform
- **GitHub Actions**: CI/CD automation
- **Turborepo**: High-performance build system
- **Nginx**: Web server and reverse proxy
- **Prometheus**: Monitoring and alerting
- **Grafana**: Metrics visualization
- **ELK Stack**: Log aggregation and analysis (Elasticsearch, Logstash, Kibana)

Docker Setup

The project uses Docker for containerization and development. The setup includes:

Development Environment

- **Docker Compose**: Orchestrates multiple services for local development
- **Hot Reload**: Automatic code reloading for both frontend and backend
- **Volume Mounting**: Persistent storage for development data
- **Environment Variables**: Secure configuration management

Production Environment

- **Multi-stage Builds**: Optimized container images for production
- **Security**: Non-root user execution and minimal dependencies
- **Health Checks**: Container health monitoring
- **Resource Limits**: CPU and memory constraints

Services

- **API Service**: FastAPI application with Gunicorn
- **Web Service**: Next.js application with Node.js
- **Database Service**: PostgreSQL with persistent storage
- **Cache Service**: Redis for session management
- **Nginx Service**: Reverse proxy and static file serving

Email Service

- Mailgun: Transactional email service for password reset and notifications
  - SMTP and API-based email delivery
  - Email templates for consistent branding
  - Delivery tracking and analytics
  - Spam protection and domain authentication
  - Rate limiting and throttling controls

Email Templates

- **Password Reset**: Secure token-based password reset flow
- **Welcome Email**: New user onboarding
- **Notification Emails**: System alerts and updates
- **Marketing Emails**: Optional promotional content

Security Features

- DKIM and SPF authentication
- TLS encryption for email transmission
- Rate limiting to prevent abuse
- IP allowlisting for trusted senders
- Bounce and complaint handling

Environment Management

The project uses a comprehensive environment management system to handle different deployment scenarios and configurations.

Environment Variables

- **Development**: `.env.development` for local development
- **Testing**: `.env.test` for CI/CD pipeline
- **Production**: `.env.production` for live deployment
- **Shared**: `.env.shared` for common variables

Variable Categories

- **Database**: Connection strings and credentials
- **Authentication**: JWT secrets and OAuth keys
- **External Services**: API keys and endpoints
- **Application**: Feature flags and configuration

- **Security**: Encryption keys and certificates

Security Best Practices

- Never commit `.env` files to version control
- Use environment-specific variable files
- Implement variable validation using Zod
- Encrypt sensitive values in production
- Rotate credentials regularly

Vercel Deployment

The frontend application is deployed on Vercel for optimal performance and scalability.

Deployment Process

1. **Automatic Deployments**
    - Triggered by pushes to main branch
    - Preview deployments for pull requests
    - Branch-specific deployments
2. **Environment Configuration**
    - Vercel dashboard for environment variables
    - Production and preview environments
    - Secret management for sensitive data
3. **Build Settings**
    - Framework preset: Next.js
    - Build command: `npm run build`
    - Output directory: `.next`
    - Node.js version: 18.x
4. **Performance Optimization**
    - Edge Network distribution
    - Automatic HTTPS
    - Image optimization
    - Serverless functions
5. **Monitoring and Analytics**
    - Real-time performance metrics

- Error tracking
- Analytics integration
- Deployment logs

Deployment Checklist

- [ ] Configure environment variables
- [ ] Set up custom domains
- [ ] Enable HTTPS
- [ ] Configure build settings
- [ ] Set up monitoring
- [ ] Test production build
- [ ] Verify API endpoints
- [ ] Check environment variables
- [ ] Validate authentication
- [ ] Test email functionality

GitHub Actions and CI

1. **Continuous Integration Pipeline**
    - Automated testing on pull requests
    - Branch protection rules
    - Code quality checks
    - Security scanning
2. **Backend CI Workflow**
    - Python 3.11 environment setup
    - Poetry dependency management
    - PostgreSQL 15 service container
    - Redis 7 service container
    - Pytest test execution
    - Code coverage reporting
3. **Frontend CI Workflow**
    - Node.js environment setup
    - NPM dependency installation
    - Jest test execution
    - TypeScript type checking
    - Build verification

- Bundle analysis

4. **CI Configuration**

```
# Key CI Features
- Automated testing
- Service containers
- Dependency caching
- Parallel job execution
- Artifact management
```

5. **Quality Gates**

   - Test coverage thresholds
   - Type checking requirements
   - Linting standards
   - Security scanning
   - Performance benchmarks

6. **CI Best Practices**

   - Fast feedback loops
   - Cached dependencies
   - Parallel test execution
   - Automated deployments
   - Environment parity
   - Security scanning
   - Code quality checks

CI Checklist

- [ ] Configure GitHub Actions workflows
- [ ] Set up service containers
- [ ] Configure test environments
- [ ] Set up caching
- [ ] Configure branch protection
- [ ] Set up automated testing
- [ ] Configure code coverage
- [ ] Set up security scanning
- [ ] Configure deployment automation
- [ ] Set up monitoring and alerts

Author

This template was created by Mac Anderson, a full-stack developer specializing in AI/ML systems. With experience in building and scaling successful startups, Mac is dedicated to creating robust applications and sharing technical knowledge with the developer community.

Connect & Learn More

- **Personal Blog & Portfolio**: macanderson.com - Technical articles, project showcases, and development insights
- **Code Writing Founders**: 02Beta.com - A blog dedicated to helping founders write better code and build better products

Feel free to reach out for:

- Technical consulting
- Code reviews
- Project collaboration
- Speaking engagements
- Writing opportunities

Support the Project

If you find this template helpful, consider:

- Starring the repository
- Contributing to the codebase
- Sharing with your network
- Following for updates