MATRIXGIN V2.0 — ПОЛНЫЙ ПЛАН РЕАЛИЗАЦИИ ПРОЕКТА И ЧЕКЛИСТ

> **Дата:** 21 ноября 2025
> **Версия:** 1.0
> **Статус:** READY FOR EXECUTION
> **Автор:** Экспертная команда MatrixGin

СОДЕРЖАНИЕ

1. ОБЗОР ПРОЕКТА И КЛЮЧЕВЫЕ ПАРАМЕТРЫ

1.1 Описание проекта

MatrixGin v2.0 — революционная интегрированная платформа управления предприятием с встроенным AI-ассистентом, реализующая философию Lean Production и Kaizen 2.0.

Уникальность:

- Полная адаптация для российского законодательства (152-ФЗ, ТК РФ, 54-ФЗ, НК РФ)
- Локальные LLM (Qwen, DeepSeek) вместо платных API (экономия $50,000+/год)
- Гибридная команда (Человек + AI) с Human Oversight Protocol
- Геймификация и система мотивации (MatrixCoin + GMC)
- Telegram Mini Apps как primary interface

1.2 Стратегические цели

#	Цель	Как измерить	Целевое значение
1	Запустить MVP за 3-4 месяца	Дата запуска в production	1 апреля 2026
2	Внедрить на 2-3 пилотных клиентах	Кол-во активных клиентов	3
3	Достичь 80%+ user engagement	Weekly active users / Total users	>80%
4	Обеспечить 99.5% uptime	Downtime в месяц	<3.6 часов
5	Полное соответствие 152-ФЗ	Security audit results	Pass
6	Экономия на LLM API	Затраты на облачные LLM	$0-6,000/мес

1.3 Ключевые параметры проекта

Параметр	Значение
Продолжительность MVP	12-16 недель (3-4 месяца)
Полная разработка	40-50 недель (9-12 месяцев)
Размер команды	10-12 человек (или 7-8 + аутсорс)
CAPEX	$390,000 (первый год)
OPEX	$72,000-240,000 (первый год)
Целевые клиенты	Малый/средний бизнес 50-5,000 сотрудников
Запуск пилота	Q2 2026 (май-июнь)
Break-even	Месяц 18 (май 2027) при 30 клиентах
Целевое значение ARR Год 3	$1,800,000

2. СТРУКТУРА ПРОЕКТА (WBS)

2.1 Иерархия работ

```
MatrixGin v2.0
│
├── ФАЗА 0: INFRASTRUCTURE &amp;amp; SETUP (Недели 1-2)
│   ├── Инфраструктура (Cloud, VPS)
│   ├── CI/CD pipelines (GitHub Actions)
│   ├── Database setup (PostgreSQL, Redis, Qdrant)
│   ├── Мониторинг и логирование (Prometheus, ELK)
│   └── Dev &amp;amp; staging окружения
│
├── ФАЗА 1: MVP FOUNDATION (Недели 3-8)
│   ├── Backend core (Nest.js, Authentication)
│   ├── Database schema (Users, Tasks, KPI)
│   ├── Frontend core (React 18, Tailwind CSS)
│   ├── Telegram Bot (базовые команды)
```

```
│    ├── Authorization &amp;amp; RBAC
│    └── API endpoints (первый набор)
│
├── ФАЗА 2: CORE MODULES (Недели 9-14)
│    ├── Employee Management (CRUD, структура)
│    ├── Task Management (создание, назначение, статусы)
│    ├── KPI &amp;amp; Analytics (персональные KPI, дашборды)
│    ├── MatrixCoin Economy (базовая система)
│    ├── Legal Compliance (152-ФЗ базовая)
│    └── Gamification (статусы, достижения)
│
├── ФАЗА 3: MVP LAUNCH (Недели 15-16)
│    ├── QA &amp;amp; тестирование
│    ├── Security audit
│    ├── Deployment на production
│    ├── Документация
│    └── Запуск пилота (1-2 клиента)
│
├── ФАЗА 2+ (Месяцы 5-12): EXTENDED FEATURES
│    ├── Emotional Analytics
│    ├── HR Analytics (Matrix360)
│    ├── RAG System
│    ├── ERP-модули (Закупки, Склад, Бюджет, ОС)
│    ├── Production Management
│    └── Advanced features
│
└── ОПЕРАЦИОННАЯ ПОДДЕРЖКА (Параллельно)
     ├── Go-to-market
     ├── Sales &amp;amp; Marketing
     ├── Customer support
     └── Привлечение инвестиций
```

2.2 Основные вехи проекта (Milestones)

#	Веха	Дата	Зависит от
M1	Инфраструктура ready	Неделя 2	Подбор хостинга
M2	Backend + auth working	Неделя 8	Дизайн БД
M3	Frontend + Telegram Bot	Неделя 8	React setup
M4	Core modules 70% ready	Неделя 14	M2 + M3
M5	**MVP release на staging**	Неделя 15	Все модули
M6	**Security audit passed**	Неделя 16	Рентест
M7	**MVP в production**	Неделя 17 (апрель 2026)	M6
M8	Пилот-клиент #1 active	Неделя 20	M7 + onboarding
M9	Feedback собран	Неделя 24	2 недели использования

3. ФАЗЫ РАЗРАБОТКИ (40-50 НЕДЕЛЬ)

3.1 ФАЗА 0: INFRASTRUCTURE & SETUP (Недели 1-2)

Цель: Подготовить полную инфраструктуру для разработки и production.

3.1.1 Выбор хостинга и облака

Опции:

1. **Selectel** (российский провайдер, 152-ФЗ compliant)
   - VPS: 2 CPU, 8GB RAM — $50/месяц
   - GPU A100 40GB — $2,000-3,000/месяц
   - Преимущества: локальное, compliance, низкие задержки
2. **Yandex.Cloud** (российский)
   - VM Instance (n1-standard-2) — $40-50/месяц
   - GPU (V100/L4) — $1,000-2,000/месяц
   - Плюсы: интеграция с YandexGPT, данные в РФ
3. **DigitalOcean** (если разрешено)
   - Droplet 8GB — $48/месяц
   - Kubernetes — $100-200/месяц

Выбор: Selectel (Russian-first, 152-ФЗ ready)

3.1.2 Docker & Containerization Setup

Задачи:

- [ ] Инициализировать Docker repository на GitHub
- [ ] Создать Dockerfiles для:
  - [ ] Backend (Node.js)
  - [ ] Python ML services
  - [ ] PostgreSQL + Redis
  - [ ] Nginx reverse proxy
- [ ] Написать docker-compose.yml для локальной разработки
- [ ] Протестировать запуск локально

3.1.3 CI/CD Pipeline (GitHub Actions)

Workflow:

```
events:
  - on: push to main/develop
    steps:
      1. Checkout code
      2. Run linter (ESLint, Prettier)
      3. Build Docker image
      4. Run unit tests (Jest)
      5. Run integration tests
      6. Build frontend (Vite)
      7. Push to Docker registry
      8. Deploy to staging
      9. Run E2E tests
      10. Merge to main → Deploy to production
```

Задачи:

- [ ] Создать .github/workflows/ci-cd.yml
- [ ] Настроить Docker registry (Selectel Registry или GitHub Container Registry)
- [ ] Настроить staging & production environments
- [ ] Настроить notifications (Slack/Telegram)

3.1.4 Database Setup

PostgreSQL:

- [ ] Инстанс PostgreSQL 16 на Selectel
- [ ] Инициальная схема (schema.sql)
- [ ] Backup стратегия (ежедневные в S3)
- [ ] Миграции (using Prisma Migrate)

Redis:

- [ ] Инстанс Redis 7
- [ ] Конфигурация persistence (RDB)
- [ ] Cluster mode (опционально)

Qdrant (Vector DB):

- [ ] Docker container Qdrant
- [ ] Collection config для embeddings

S3 (MinIO или Selectel S3):

- [ ] Bucket для документов, логов
- [ ] Lifecycle rules (архивирование через 30 дней)

3.1.5 Monitoring & Logging

Prometheus:

- [ ] Scrape config для Node.js, PostgreSQL, Redis
- [ ] Alarms (CPU > 80%, Disk > 90%, DB slow queries)

Grafana:

- [ ] Dashboards (System, Application, Database)
- [ ] Alert notifications (Slack)

ELK Stack (или Loki):

- [ ] Log aggregation setup
- [ ] Indexes для разных компонентов
- [ ] Retention: 30 дней

Sentry (для error tracking):

- [ ] Initialize Sentry project
- [ ] Configure DSN в backend & frontend
- [ ] Setup slack notifications

3.1.6 Development Environments

Локальный dev:

- [ ] .env.example для переменных окружения
- [ ] Инструкция по запуску (README)
- [ ] Pre-commit hooks (линтинг)

Staging:

- [ ] Отдельный PostgreSQL, Redis (на Selectel)
- [ ] Копия production конфига
- [ ] Автоматические E2E тесты

Production:

- [ ] HA setup (load balancer, replicas)
- [ ] DDoS protection
- [ ] WAF (Web Application Firewall)
- [ ] SSL/TLS c Let's Encrypt

Ответственный: DevOps Engineer
Время: 5-7 дней
Бюджет: $1,000-2,000 (setup + месячная подписка)

3.2 ФАЗА 1: MVP FOUNDATION (Недели 3-8)

Цель: Реализовать backend core, authentication, basic frontend и Telegram Bot.

3.2.1 Backend Architecture & Setup

Nest.js структура:

```
src/
├── main.ts
├── app.module.ts
├── config/
│   ├── database.config.ts
│   ├── jwt.config.ts
│   └── env.validation.ts
├── auth/
│   ├── auth.service.ts
│   ├── auth.controller.ts
│   ├── jwt.strategy.ts
│   ├── rbac.guard.ts
│   └── entities/
├── users/
│   ├── users.service.ts
│   ├── users.controller.ts
│   ├── dto/
│   └── entities/
├── common/
│   ├── filters/
│   ├── pipes/
│   ├── decorators/
│   └── exceptions/
└── database/
    ├── prisma.service.ts
    └── migrations/
```

Задачи:

- [ ] Инициализировать Nest.js проект с Typescript
- [ ] Настроить Prisma ORM (schema.prisma)
- [ ] Создать базовую структуру модулей
- [ ] Setup error handling & logging
- [ ] Конфигурация переменных окружения (.env)

3.2.2 Database Schema (Phase 1)

Критичные таблицы:

```
-- Users &amp;amp; Auth
CREATE TABLE users (
    id UUID PRIMARY KEY,
    email VARCHAR(255) UNIQUE,
```

```
    password_hash VARCHAR(255),
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    role_id UUID REFERENCES roles(id),
    status VARCHAR(50), -- active, inactive
    created_at TIMESTAMPTZ,
    updated_at TIMESTAMPTZ
);

-- Tasks
CREATE TABLE tasks (
    id UUID PRIMARY KEY,
    title VARCHAR(255),
    description TEXT,
    creator_id UUID REFERENCES users(id),
    assignee_id UUID REFERENCES users(id),
    status VARCHAR(50), -- new, in_progress, completed
    priority VARCHAR(20), -- low, medium, high
    created_at TIMESTAMPTZ,
    completed_at TIMESTAMPTZ
);

-- KPI
CREATE TABLE kpi (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id),
    metric_name VARCHAR(100),
    target_value DECIMAL,
    current_value DECIMAL,
    period DATE,
    created_at TIMESTAMPTZ
);

-- Event Log
CREATE TABLE event_log (
    id UUID PRIMARY KEY,
    event_type VARCHAR(50),
    entity_type VARCHAR(50),
    entity_id UUID,
    actor_id UUID REFERENCES users(id),
    timestamp TIMESTAMPTZ,
    metadata JSONB
);

-- Departments
CREATE TABLE departments (
    id UUID PRIMARY KEY,
    name VARCHAR(100),
    parent_id UUID REFERENCES departments(id),
    created_at TIMESTAMPTZ
);

-- MatrixCoin
CREATE TABLE matrixcoin (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id),
```

```
    balance DECIMAL,
    type VARCHAR(20), -- MC or GMC
    created_at TIMESTAMPTZ,
    updated_at TIMESTAMPTZ
);
```

Задачи:

- [ ] Определить Prisma schema
- [ ] Миграция БД (prisma migrate dev)
- [ ] Seed script для dev данных (200 пользователей, 500 задач)

3.2.3 Authentication & Authorization

JWT Implementation:

```
// auth.service.ts
async login(email: string, password: string): Promise&lt;JwtPayload&gt; {
  const user = await this.usersService.findByEmail(email);
  const isValid = await bcrypt.compare(password, user.password_hash);

  if (!isValid) throw new UnauthorizedException();

  const payload = { sub: user.id, email: user.email, role: user.role };
  return { access_token: this.jwtService.sign(payload) };
}
```

RBAC Middleware:

- [ ] Guard для проверки ролей (@Roles decorator)
- [ ] Permissions mapping (Role → Actions)
- [ ] Audit logging для всех действий

Задачи:

- [ ] Реализовать JWT auth (Passport.js)
- [ ] Создать RBAC систему
- [ ] Защитить все API endpoints
- [ ] Тестирование auth flow

3.2.4 Frontend Setup (React 18)

Структура:

```
frontend/
├── src/
│   ├── main.tsx
│   ├── App.tsx
│   ├── pages/
```

```
│   │   ├── Login.tsx
│   │   ├── Dashboard.tsx
│   │   ├── Tasks.tsx
│   │   └── Settings.tsx
│   ├── components/
│   │   ├── Header.tsx
│   │   ├── Sidebar.tsx
│   │   ├── TaskCard.tsx
│   │   └── KpiDisplay.tsx
│   ├── hooks/
│   │   ├── useAuth.ts
│   │   ├── useTasks.ts
│   │   └── useApi.ts
│   ├── store/
│   │   └── store.ts (Redux Toolkit)
│   └── styles/
│       └── globals.css (Tailwind)
├── public/
└── vite.config.ts
```

Задачи:

- [ ] Инициализировать Vite + React 18 + Typescript
- [ ] Setup Tailwind CSS
- [ ] Setup Redux Toolkit + RTK Query
- [ ] Создать основные страницы (Login, Dashboard)
- [ ] Дизайн компонентов по Figma

3.2.5 Telegram Bot Integration (MVP)

Telegram Mini Apps (TMA):

```
// telegramBot.service.ts
async handleCommand(message: Update): Promise&lt;void&gt; {
  if (message.message?.text === '/start') {
    // Отправить приветствие + ссылку на web app
    const webAppButton = {
      text: 'Открыть MatrixGin',
      web_app: { url: 'https://matrixgin.app' }
    };
  }

  if (message.message?.text?.startsWith('/tasks')) {
    // Получить задачи пользователя
    const tasks = await this.tasksService.getUserTasks(userId);
    // Форматировать и отправить
  }
}
```

Базовые команды:

- `/start` — Приветствие + ссылка на web app
- `/tasks` — Мои задачи
- `/status` — Мой статус/KPI
- `/help` — Справка

Задачи:

- [ ] Setup Telegram Bot на BotFather
- [ ] Разработать Telegram Mini App (WebView)
- [ ] Интеграция с backend API
- [ ] Тестирование базовых команд

3.2.6 API Endpoints (Phase 1)

Реализовать 20-25 основных эндпоинтов:

Auth:

- [ ] POST /api/auth/register
- [ ] POST /api/auth/login
- [ ] POST /api/auth/logout
- [ ] POST /api/auth/refresh

Users:

- [ ] GET /api/users
- [ ] GET /api/users/{id}
- [ ] POST /api/users
- [ ] PUT /api/users/{id}
- [ ] PATCH /api/users/{id}
- [ ] DELETE /api/users/{id}

Tasks:

- [ ] GET /api/tasks
- [ ] GET /api/tasks/{id}
- [ ] POST /api/tasks
- [ ] PUT /api/tasks/{id}
- [ ] DELETE /api/tasks/{id}
- [ ] POST /api/tasks/{id}/assign
- [ ] POST /api/tasks/{id}/complete

KPI:

- [ ] GET /api/kpi
- [ ] GET /api/kpi/{userId}
- [ ] POST /api/kpi

Departments:

- [ ] GET /api/departments
- [ ] POST /api/departments

MatrixCoin:

- [ ] GET /api/economy/balance/{userId}
- [ ] GET /api/economy/transactions

Ответственный: Senior Backend Developer + Frontend Developer
Время: 16-20 дней
Результат: Работающий MVP с authentication, базовой функциональностью

3.3 ФАЗА 2: CORE MODULES (Недели 9-14)

Цель: Реализовать 7 критичных модулей для MVP.

Модуль 1: Employee Management

Функции:

- CRUD сотрудников
- Организационная структура (иерархия отделов)
- Должности и ранги
- Статусы сотрудников

Задачи:

- [ ] Расширить users table (department_id, position, rank)
- [ ] API для управления структурой
- [ ] Дашборд HR (список сотрудников)
- [ ] Экспорт в CSV/Excel
- [ ] Интеграция с 1C (отложено на Phase 2)

Время: 4-5 дней

Модуль 2: Task Management (Smart)

Функции:

- Создание задач (форма + NLP)
- Автоназначение (по MDR или волонтеры)

- Статусы и workflow
- Комментарии
- Уведомления

Задачи:

- [ ] Расширить tasks table (comments, attachments, watchers)
- [ ] Websocket для real-time updates
- [ ] NLP парсинг (отложено, сейчас простая форма)
- [ ] Notification engine (email, Telegram)
- [ ] Task filtering & search

Время: 5-6 дней

Модуль 3: KPI & Analytics (Basic)

Функции:

- Персональные KPI
- Дашборд прогресса
- Простые отчеты (PDF)
- Графики выполнения

Задачи:

- [ ] Дизайн KPI структуры (таблица kpi_snapshots)
- [ ] Автоматический расчет прогресса
- [ ] Дашборд для сотрудников
- [ ] Дашборд для менеджеров (команды)
- [ ] Charts (Recharts + Chart.js)

Время: 4-5 дней

Модуль 4: MatrixCoin Economy (Basic)

Функции:

- Кошельки (MC + GMC)
- Транзакции (переводы)
- История операций
- Простой магазин (опционально)

Задачи:

- [ ] Таблицы: wallets, transactions
- [ ] API для операций (с валидацией баланса)

- [ ] Начисление MC за выполненные задачи (+10 MC)
- [ ] Дашборд кошелька
- [ ] История транзакций

Время: 3-4 дня

Модуль 5: Legal Compliance (Base)

Функции:

- Базовая 152-ФЗ (согласия)
- Audit log
- NDA при регистрации
- Простой checklist

Задачи:

- [ ] Таблицы: compliance_log, nda_records
- [ ] API для управления согласиями
- [ ] Генерация NDA текста
- [ ] Tracking принятия документов
- [ ] Простой compliance dashboard

Время: 3-4 дня

Модуль 6: Telegram Bot Integration (Extended)

Функции:

- Расширенные команды
- Создание задач через бот
- Уведомления
- Статус пользователя

Команды:

```
/start — Приветствие
/tasks — Мои задачи
/status — Мой статус &amp;amp; баланс
/create — Создать задачу (форма)
/complete — Отметить задачу (выбор)
/help — Справка
```

Задачи:

- [ ] Разработать conversational interface
- [ ] Middleware для auth (через token)

- [ ] Обработка inline buttons
- [ ] Интеграция с основным API
- [ ] Тестирование на 100+ пользователях

Время: 4-5 дней

Модуль 7: Gamification (Base)

Функции:

- Статусы (5 уровней: Фотон, Топчик, Кремень, Углерод, UNIVERSE)
- Простые достижения
- Лидерборд
- Автоматический расчет статуса

Задачи:

- [ ] Таблицы: statuses, achievements, leaderboard (view)
- [ ] Cron job для ежедневного пересчета статуса
- [ ] API для получения статуса & привилегий
- [ ] Дашборд лидерборда
- [ ] Upgrade ceremony notifications

Время: 3-4 дня

Ответственный: 2-3 backend разработчика + 1-2 frontend разработчика
Время: 21-28 дней
Результат: 7 полностью функциональных модулей

3.4 ФАЗА 3: MVP LAUNCH & TESTING (Недели 15-16)

Цель: Протестировать, исправить баги, запустить в production.

3.4.1 Quality Assurance

Unit тестирование:

- [ ] Все services имеют unit tests (Jest)
- [ ] Coverage > 70%
- [ ] Запуск автоматически в CI/CD

Integration тестирование:

- [ ] API endpoint tests (Supertest)
- [ ] Database interactions
- [ ] Auth flow

E2E тестирование:

- [ ] Критичные user flows (login → create task → complete)
- [ ] Telegram Bot команды
- [ ] Payment flow (если есть)
- [ ] Инструмент: Playwright или Cypress

Performance тестирование:

- [ ] Load testing (k6)
- [ ] Целевой: <500ms на основные запросы
- [ ] Concurrent users: 100+

Задачи:

- [ ] Написать test plans для каждого модуля
- [ ] Выполнить manual testing (QA team 2-3 человека)
- [ ] Автоматические tests в CI/CD
- [ ] Performance profiling (Node.js profiler)
- [ ] Memory leak detection

3.4.2 Security Audit & Penetration Testing

Внутренний аудит:

- [ ] JWT token security
- [ ] SQL injection prevention (Prisma ORM защищен)
- [ ] CSRF protection
- [ ] XSS prevention
- [ ] CORS configuration
- [ ] Rate limiting

Проверки:

- [ ] OWASP Top 10
- [ ] Dependency vulnerabilities (npm audit, Snyk)
- [ ] SSL/TLS configuration
- [ ] Database encryption

Пентест (внешняя компания):

- [ ] Контрактор: Security company
- [ ] Стоимость: $3,000-5,000
- [ ] Результат: Report с рекомендациями

- [ ] Исправление критичных уязвимостей

Задачи:

- [ ] Провести внутренний аудит
- [ ] Заказать пентест (параллельно с разработкой)
- [ ] Исправить найденные проблемы
- [ ] Получить security certificate

3.4.3 Documentation

Техническая документация:

- [ ] API Documentation (OpenAPI/Swagger)
- [ ] Database Schema (EER diagram)
- [ ] Architecture Decision Records (ADR)
- [ ] Deployment guide
- [ ] Troubleshooting guide

Пользовательская документация:

- [ ] User manual (PDF, 10-15 страниц)
- [ ] Video tutorials (3-5 видео по 5-10 минут)
- [ ] FAQ

Developer documentation:

- [ ] Setup guide (README)
- [ ] Code style guide
- [ ] Contribution guidelines
- [ ] Git workflow

Задачи:

- [ ] Создать Confluence/Notion wiki
- [ ] Swagger UI для API (автогенерируется)
- [ ] Записать видео туториалы (Loom)
- [ ] Подготовить onboarding материалы

3.4.4 Production Deployment

Pre-deployment чеклист:

- [ ] Все тесты passed (100%)
- [ ] Security audit completed
- [ ] Database backups working

- [ ] Monitoring configured
- [ ] Rollback plan ready
- [ ] Communication plan для stakeholders

Deployment процесс:

```
1. Freeze на new features (24 часа до релиза)
2. Final testing на staging
3. Backup production database
4. Blue-green deployment (параллельная версия)
5. Health checks (API, Database, services)
6. Smoke tests на production
7. Gradual traffic shift (10% → 50% → 100%)
8. Monitoring escalation
```

Задачи:

- [ ] Подготовить production servers
- [ ] Миграции БД (production)
- [ ] Настроить SSL/TLS certificates
- [ ] Финальное тестирование
- [ ] Запуск deployment скрипта
- [ ] Мониторинг 24/7 в первые 48 часов

Время: 7-10 дней (параллельно с Phase 2)
Результат: MVP v1.0 в production с нулевыми инцидентами

3.5 PHASE 2+ (Месяцы 5-12): EXTENDED FEATURES

После успешного запуска MVP и сбора feedback от пилот-клиентов:

Месяцы 5-8 (Phase 2):

- Psychological Support & Emotional Analytics
- HR Analytics (Matrix360)
- Learning & Corporate University
- Advanced Gamification (GMC, Auctions)
- RAG System & Knowledge Base

Месяцы 9-12 (Phase 3):

- Критичные ERP-модули (Procurement, WMS, Budgeting, Fixed Assets)
- Production Management (MES, Quality)
- Content Factory
- Executive Dashboard

- Advanced Analytics & Predictions

4. КОМАНДА ПРОЕКТА И РОЛИ

4.1 Структура команды (10-12 человек)

#	Роль	Кол-во	Опыт	Зарплата/мес	Примечание
1	Product Manager	1	5-7 лет	$2,500-3,500	Vision & priorities
2	Tech Lead/Architect	1	8-10 лет	$3,500-5,000	System design
3	Senior Backend Dev	2	5-7 лет	$2,500-3,500	Core services
4	Backend Dev (Middle)	2	3-5 лет	$1,500-2,500	API development
5	Frontend Lead	1	5-7 лет	$2,500-3,500	UI/UX & design
6	Frontend Dev	1	3-5 лет	$1,500-2,500	React components
7	QA/Test Engineer	1	3-5 лет	$1,200-2,000	Тестирование
8	DevOps Engineer	1	3-5 лет	$2,000-3,000	Инфра & CI/CD
9	ML/LLM Specialist	1	3-5 лет	$2,000-3,000	Local LLM setup
10	Telegram Bot Dev (part-time)	1	2-3 года	$800-1,200	Bot integration
11	Security/Compliance Officer (part-time)	1	5-7 лет	$1,500-2,500	152-ФЗ audit

Итого: $20,600-32,720 месячно (разработка + поддержка)

За 40-50 недель (9-12 месяцев): $185,400-393,840

4.2 Распределение ответственности

Компонент	Владелец	Backup
Backend Architecture	Tech Lead	Senior Backend Dev
Frontend Architecture	Frontend Lead	Senior Backend Dev
Database Design	Tech Lead	Backend Dev (Middle)
API Endpoints	Senior Backend Dev + Middle	Backend Dev (Middle)
React Components	Frontend Lead + Frontend Dev	Frontend Lead
Telegram Bot	Telegram Bot Dev	Senior Backend Dev
Testing Strategy	QA Engineer	Tech Lead
Deployment & CI/CD	DevOps Engineer	Tech Lead
LLM Integration	ML Specialist	Senior Backend Dev
Security & Compliance	Security Officer	Tech Lead
Product Vision	Product Manager	Tech Lead

4.3 Communication & Meetings

Daily:

- 15:00 — Daily standup (15 минут, каждый день)

Weekly:

- Понедельник 10:00 — Sprint planning (1 час)
- Среда 14:00 — Technical sync (1 час)
- Пятница 16:00 — Sprint retro (1 час)

Ad-hoc:

- Code reviews (во время разработки)
- Emergency calls (при критичных багах)

5. РИСК-МЕНЕДЖМЕНТ

5.1 Основные риски и стратегии снижения

#	Риск	Вероятность	Влияние	Стратегия снижения
1	Задержка на месяц+	Средняя	Высокое	MVP из 5 модулей (вместо 7), параллельная разработка
2	Утечка данных 152-ФЗ	Низкая	Критичное	Пентест в week 8, encryption AES-256, audit logs
3	LLM API dependency	Средняя	Среднее	Локальные LLM как основа, fallback стратегия
4	Уход ключевого разработчика	Низкая	Высокое	Документирование, парное программирование
5	Performance issues на production	Средняя	Среднее	Load testing (k6), database indexing, caching strategy
6	Пилот-клиент не доволен	Средняя	Высокое	Еженедельные review, быстрое исправление багов, support 24/5
7	Несовместимость с 1С	Средняя	Среднее	Тестирование интеграции на week 6, backup API approach

5.2 Risk Register (Детальный)

Risk #1: Задержка разработки

- **Причины:** Scope creep, недооценка сложности, technical blockers
- **Индикаторы:** Более 3 tasks не завершено к концу спринта
- **Ответственный:** Tech Lead + Product Manager
- **План B:** Reduce scope (убрать 2-3 низкоприоритетных модуля)

Risk #2: Безопасность 152-ФЗ

- **Причины:** Неполное понимание требований, недостаточный аудит
- **Индикаторы:** Security review находит критичные findings
- **Ответственный:** Security Officer
- **План B:** Отложить production на 2 недели, провести полный audit

Risk #3: LLM затраты выше бюджета

- **Причины:** Неопределенный API consumption, low inference efficiency
- **Индикаторы:** Счета за API > $2,000/месяц
- **Ответственный:** ML Specialist + DevOps
- **План B:** Переключиться на полностью локальные LLM (может потребоваться замедление ответов)

6. ПОЛНЫЙ ЧЕКЛИСТ РЕАЛИЗАЦИИ

6.1 ФАЗА 0 CHECKLIST (Недели 1-2)

Инфраструктура

- [ ] **Выбран хостинг-провайдер** (Selectel)
  - [ ] Зарегистрирован аккаунт
  - [ ] Выбрана тарифный план (VPS 8GB RAM, $50/месяц)
  - [ ] SSH ключи сгенерированы
  - [ ] Сервер ready (IP, hostname)
- [ ] **Регионы и DPA (152-ФЗ)**
  - [ ] Сервер в России (Selectel Москва)
  - [ ] Data Processing Agreement подписан с провайдером
  - [ ] Документирование place of data storage
- [ ] **Docker & Containerization**
  - [ ] Docker installed на сервере
  - [ ] Docker Compose configured
  - [ ] Dockerfile для Node.js написан
  - [ ] Dockerfile для Python написан
  - [ ] Dockerfile для Nginx написан
  - [ ] docker-compose.yml для dev написан
  - [ ] Локальный docker setup тестирован
- [ ] **GitHub Repository Setup**

    - [ ] Репо создано (matrixgin-backend, matrixgin-frontend)
    - [ ] .gitignore configured
    - [ ] Protected branches (main, develop)
    - [ ] Collaborators добавлены
    - [ ] Branch protection rules:
        - [ ] Require 2 code reviews
        - [ ] Require passing CI/CD checks
- [ ] **CI/CD Pipeline (GitHub Actions)**
    - [ ] .github/workflows/ci-cd.yml создан
    - [ ] Workflow triggered on push/PR
    - [ ] Stages:
        - [ ] Lint (ESLint, Prettier)
        - [ ] Unit tests (Jest)
        - [ ] Build Docker image
        - [ ] Push to registry
        - [ ] Deploy to staging
        - [ ] E2E tests
    - [ ] Slack notifications configured
- [ ] **Database Infrastructure**
    - [ ] PostgreSQL 16 instance на Selectel
    - [ ] Connection string добавлен в .env
    - [ ] Backup стратегия (ежедневно в S3)
    - [ ] Retention: 30 дней
    - [ ] SSH tunnel для локального подключения
- [ ] **Redis Setup**
    - [ ] Redis instance deployed
    - [ ] RDB persistence enabled
    - [ ] Memory limit configured (8GB)
    - [ ] Password set
- [ ] **Qdrant Vector DB**
    - [ ] Docker контейнер запущен
    - [ ] Port mapped (6333)
    - [ ] Collection schema подготовлена (для Phase 2)
- [ ] **S3 Storage (MinIO/Selectel)**
    - [ ] S3 bucket создан (documents, logs)

    - [ ] Access keys generated
    - [ ] Lifecycle rules: archive after 30 days
- [ ] **Monitoring & Logging**
    - [ ] Prometheus deployed (на VM или cloud)
    - [ ] Scrape configs для Node.js, PostgreSQL, Redis
    - [ ] Alarms configured (CPU, Disk, DB slow queries)
    - [ ] Grafana dashboards created (System, App, DB)
    - [ ] ELK Stack deployed (или Loki)
    - [ ] Log aggregation working
    - [ ] Sentry project created
    - [ ] DSN integrated в backend & frontend
- [ ] **SSL/TLS & Security**
    - [ ] SSL certificate ordered (Let's Encrypt)
    - [ ] Nginx reverse proxy configured
    - [ ] HTTPS enforced
    - [ ] Security headers added (HSTS, CSP, X-Frame-Options)
    - [ ] WAF rules configured (DDoS protection)
- [ ] **Development Environment**
    - [ ] .env.example template created
    - [ ] Dev setup guide написан
    - [ ] Pre-commit hooks configured (husky)
    - [ ] IDE extensions recommended (VSCode)
    - [ ] README updated

Документирование

- [ ] Infra documentation создана (Confluence/Notion)
- [ ] Access management documented (who has what)
- [ ] Disaster recovery plan written

Signoff: DevOps Engineer, Tech Lead
Status: □ In Progress

6.2 ФАЗА 1 CHECKLIST (Недели 3-8)

Backend Foundation

- [ ] **Nest.js Setup**
  - [ ] Project initialized (npm install)
  - [ ] TypeScript configured (tsconfig.json)
  - [ ] @nestjs/cli installed
  - [ ] Main modules created (AppModule, etc.)
  - [ ] Main.ts configured
- [ ] **Database Connection**
  - [ ] Prisma ORM initialized
  - [ ] Prisma schema drafted (initial tables)
  - [ ] Database migrations working
  - [ ] Seed script for dev data
  - [ ] Connection pooling configured
- [ ] **Authentication (JWT)**
  - [ ] @nestjs/jwt installed
  - [ ] Passport.js with JWT strategy
  - [ ] auth.controller.ts created
  - [ ] auth.service.ts with login/register
  - [ ] Password hashing (bcrypt) working
  - [ ] JWT tokens generated (15 min expiry)
  - [ ] Refresh tokens implemented (7 days)
  - [ ] JWT stored securely (httpOnly cookies)
  - [ ] Tests written for auth flow
- [ ] **Authorization (RBAC)**
  - [ ] roles table in database
  - [ ] Roles defined (Admin, Manager, Employee, Bot)
  - [ ] @Roles() decorator created
  - [ ] RolesGuard middleware
  - [ ] Routes protected with @Roles
  - [ ] Unit tests for RBAC
- [ ] **Error Handling**
  - [ ] ExceptionFilter for HTTP errors
  - [ ] Custom exception classes

    - [ ] Error responses formatted standardly
    - [ ] Logging errors with stack trace
    - [ ] 404, 400, 401, 403, 500 handled
- [ ] **Logging**
    - [ ] Winston logger configured
    - [ ] Logs to console + file
    - [ ] Log levels: debug, info, warn, error
    - [ ] Request logging middleware
    - [ ] Performance logging
- [ ] **API Documentation**
    - [ ] @nestjs/swagger installed
    - [ ] Swagger decorators on endpoints
    - [ ] Swagger UI at /api/docs
    - [ ] All endpoints documented

Database Schema (Phase 1)

- [ ] **Core Tables**
    - [ ] users table (id, email, password_hash, first_name, last_name, role_id, status, created_at)
    - [ ] roles table (id, name, permissions JSONB)
    - [ ] tasks table (id, title, description, creator_id, assignee_id, status, priority, created_at, completed_at)
    - [ ] kpi table (id, user_id, metric_name, target, current, period)
    - [ ] departments table (id, name, parent_id)
    - [ ] matrixcoin table (id, user_id, balance, type (MC/GMC))
    - [ ] event_log table (id, event_type, entity_type, entity_id, actor_id, timestamp, metadata JSONB)
    - [ ] compliance_log table (id, user_id, action, timestamp)
- [ ] **Migrations & Seeds**
    - [ ] Migration files created
    - [ ] Initial schema migrated to dev DB
    - [ ] Seed script creates:
        - [ ] 5 roles (Admin, HR, Manager, Employee, Bot)
        - [ ] 100+ users (test@example.com, etc.)
        - [ ] 10 departments
        - [ ] 500 tasks (various statuses)

        - [ ] Sample KPI data
    - [ ] Seed script tested on clean DB

Frontend (React 18)

- [ ] **Project Setup**
    - [ ] Vite + React 18 + TypeScript
    - [ ] package.json dependencies
    - [ ] tsconfig.json proper
    - [ ] vite.config.ts configured
    - [ ] ESLint + Prettier setup
- [ ] **Styling**
    - [ ] Tailwind CSS installed
    - [ ] tailwind.config.js
    - [ ] CSS reset applied
    - [ ] Color scheme defined (use design system from Architecture doc)
    - [ ] Responsive utilities (sm, md, lg breakpoints)
- [ ] **State Management**
    - [ ] Redux Toolkit configured
    - [ ] RTK Query setup (@reduxjs/toolkit/query)
    - [ ] Base query with auth header
    - [ ] Store persisted (localStorage for tokens)
- [ ] **Pages & Components**
    - [ ] Layout component (Header, Sidebar, Main)
    - [ ] Login page (email/password form)
    - [ ] Dashboard page (skeleton)
    - [ ] Tasks page (list + create form)
    - [ ] Settings page (profile, logout)
    - [ ] 404 page
- [ ] **Hooks & Utilities**
    - [ ] useAuth hook (login, logout, currentUser)
    - [ ] useApi hook (API calls with error handling)
    - [ ] useTasks hook (getTasks, createTask, updateTask)
    - [ ] useLocalStorage hook (for token persistence)
- [ ] **Form Handling**
    - [ ] React Hook Form installed

    - [ ] Zod for schema validation
    - [ ] Login form with validation
    - [ ] Task creation form
    - [ ] Error messages displayed
- [ ] **API Integration**
    - [ ] API client configured (Axios or fetch)
    - [ ] Base URL from environment
    - [ ] Auth interceptor (add JWT to headers)
    - [ ] Error handler middleware
    - [ ] Request/response logging

Telegram Bot (MVP)

- [ ] **Bot Setup**
    - [ ] Bot token from BotFather
    - [ ] Webhook or polling method chosen
    - [ ] Bot library (node-telegram-bot-api or telegraf)
- [ ] **Commands Implemented**
    - [ ] /start command (welcome message + web app link)
    - [ ] /help (list of commands)
    - [ ] /tasks (show user tasks)
    - [ ] /status (show user status & MC balance)
    - [ ] /create (create task via inline form)
    - [ ] /logout (invalidate bot session)
- [ ] **Integration with Backend**
    - [ ] Bot communicates with backend API
    - [ ] Authentication via user ID + bot token
    - [ ] Error handling & retry logic
    - [ ] Response formatting (readable for Telegram)
- [ ] **Telegram Mini App (TMA)**
    - [ ] TMA button in /start message
    - [ ] Web app URL configured
    - [ ] TMA SDK integrated (window.Telegram.WebApp)
    - [ ] User data passed from Telegram to web app
    - [ ] Web app communicates back to bot (optional)

API Endpoints (Phase 1)

Auth:

- [ ] POST /api/auth/register (email, password, first_name, last_name)
- [ ] POST /api/auth/login (email, password)
- [ ] POST /api/auth/logout
- [ ] POST /api/auth/refresh (using refresh token)
- [ ] GET /api/auth/me (current user info)

Users:

- [ ] GET /api/users (with pagination: ?page=1&limit=20)
- [ ] GET /api/users/{id}
- [ ] POST /api/users (Admin only)
- [ ] PUT /api/users/{id} (self or admin)
- [ ] PATCH /api/users/{id} (partial update)
- [ ] DELETE /api/users/{id} (soft delete)
- [ ] GET /api/users/{id}/analytics (KPI summary)

Tasks:

- [ ] GET /api/tasks (filters: status, assignee, department, priority)
- [ ] GET /api/tasks/{id}
- [ ] POST /api/tasks (title, description, assignee_id, priority)
- [ ] PUT /api/tasks/{id}
- [ ] DELETE /api/tasks/{id}
- [ ] POST /api/tasks/{id}/assign (reassign task)
- [ ] POST /api/tasks/{id}/complete (mark done)
- [ ] POST /api/tasks/{id}/comment (add comment)

KPI:

- [ ] GET /api/kpi (user KPIs)
- [ ] GET /api/kpi/{userId}
- [ ] POST /api/kpi (create KPI target)
- [ ] PUT /api/kpi/{id} (update current value)

Departments:

- [ ] GET /api/departments (list with hierarchy)
- [ ] GET /api/departments/{id}
- [ ] GET /api/departments/{id}/employees

MatrixCoin:

- [ ] GET /api/economy/balance/{userId} (MC + GMC)
- [ ] GET /api/economy/transactions (with pagination)
- [ ] POST /api/economy/transactions (transfer MC between users)

Roles & Permissions:

- [ ] GET /api/roles
- [ ] GET /api/permissions

Testing

- [ ] **Unit Tests (Backend)**
  - [ ] auth.service.spec.ts (login, register)
  - [ ] users.service.spec.ts (CRUD)
  - [ ] tasks.service.spec.ts (create, update, complete)
  - [ ] Coverage goal: >60%
- [ ] **Integration Tests**
  - [ ] Auth flow (register → login → access protected endpoint)
  - [ ] Task creation → assignment → completion
  - [ ] Database transactions
- [ ] **Frontend Tests (Optional for MVP)**
  - [ ] Login component renders correctly
  - [ ] Form validation works
  - [ ] API errors displayed

Documentation

- [ ] Backend README created
- [ ] Database schema documented (Excalidraw diagram)
- [ ] API endpoints listed (Swagger)
- [ ] Setup guide for developers
- [ ] Environment variables documented (.env.example)

Signoff: Tech Lead, Frontend Lead, Backend Lead
Status: In Progress

6.3 ФАЗА 2 CHECKLIST (Недели 9-14)

Module 1: Employee Management

- [ ] Database schema expanded
  - [ ] users: добавить position, rank, hire_date, status
  - [ ] departments: parent_id для иерархии
  - [ ] positions table (if separate)
- [ ] API endpoints
  - [ ] GET /api/employees (with filters: department, status)
  - [ ] GET /api/employees/{id}
  - [ ] POST /api/employees (HR only)
  - [ ] PUT /api/employees/{id}
  - [ ] GET /api/employees/{id}/analytics
  - [ ] POST /api/employees/{id}/status/upgrade (Admin)
- [ ] Frontend
  - [ ] Employee list page (with department tree)
  - [ ] Employee detail page
  - [ ] Create/edit employee form
  - [ ] Department management (tree view)
- [ ] Integration
  - [ ] Async job for 1C sync (if applicable)
  - [ ] CSV import/export
- [ ] Tests
  - [ ] CRUD operations tested
  - [ ] Permission checks

Module 2: Task Management (Extended)

- [ ] Database schema
  - [ ] tasks: add comments (as separate table), watchers
  - [ ] task_comments table
  - [ ] task_watchers table
- [ ] API endpoints
  - [ ] POST /api/tasks/{id}/comment (add comment)
  - [ ] GET /api/tasks/{id}/comments
  - [ ] POST /api/tasks/{id}/watch (add watcher)

    - [ ] Task notifications (when assigned, commented)
- [ ] Frontend
    - [ ] Task detail page with comments
    - [ ] Real-time updates (WebSocket if budget allows)
    - [ ] Task filtering & search
    - [ ] Bulk operations (mark multiple as done)
- [ ] Notifications
    - [ ] Email notifications (task assigned, commented)
    - [ ] In-app notifications (bell icon)
    - [ ] Telegram notifications
- [ ] Tests
    - [ ] Comment creation, deletion
    - [ ] Notification sending

Module 3: KPI & Analytics (Basic)

- [ ] Database schema
    - [ ] kpi_targets table (quarterly goals)
    - [ ] kpi_snapshots table (daily/weekly snapshots)
- [ ] API endpoints
    - [ ] GET /api/kpi/my (my KPI status)
    - [ ] GET /api/kpi/department/{id} (team KPI)
    - [ ] POST /api/kpi (create target)
    - [ ] PUT /api/kpi/{id} (update current)
- [ ] Frontend
    - [ ] KPI dashboard (progress bars, trends)
    - [ ] KPI history (chart)
    - [ ] Department KPI view
- [ ] Cron job
    - [ ] Daily snapshot calculation
    - [ ] Alerts if below threshold
- [ ] Tests
    - [ ] KPI calculation logic

Module 4: MatrixCoin Economy (Extended)

- [ ] Database schema
  - [ ] wallets table (MC, GMC balances)
  - [ ] transactions table (with from_user, to_user, amount, type)
  - [ ] matrixcoin_store table (items, prices, inventory)
- [ ] API endpoints
  - [ ] GET /api/economy/balance/{userId}
  - [ ] GET /api/economy/transactions (with pagination)
  - [ ] POST /api/economy/transactions (transfer)
  - [ ] GET /api/economy/store (items)
  - [ ] POST /api/economy/store/{itemId}/buy (purchase)
  - [ ] Automatic MC reward for task completion (+10 MC)
  - [ ] Automatic MC reward for achievement (+50 MC)
- [ ] Frontend
  - [ ] Wallet display (MC, GMC)
  - [ ] Transaction history
  - [ ] Store page (buy items)
  - [ ] Personal cabinet section
- [ ] Cron job
  - [ ] Daily MC expiration check (90 days)
  - [ ] MC lifecycle management
- [ ] Tests
  - [ ] Transaction validation (sufficient balance)
  - [ ] MC expiration logic

Module 5: Legal Compliance (Extended)

- [ ] Database schema
  - [ ] compliance_consents table (what user consented to)
  - [ ] nda_signatures table (who signed NDA)
  - [ ] audit_log table (detailed logging)
- [ ] API endpoints
  - [ ] GET /api/compliance/consents (current user)
  - [ ] POST /api/compliance/consents/{consentType}/accept
  - [ ] GET /api/compliance/nda/content
  - [ ] POST /api/compliance/nda/accept (at registration)

    - [ ] GET /api/compliance/audit-log (Admin only)
    - [ ] GET /api/compliance/checklist (compliance status)
- [ ] Frontend
    - [ ] Consent acceptance flow (at login/registration)
    - [ ] NDA display & acceptance checkbox
    - [ ] Admin compliance dashboard
- [ ] Compliance features
    - [ ] 152-ФЗ consents (personal data processing)
    - [ ] Cookie consent
    - [ ] Terms of service
    - [ ] NDA signature tracking
    - [ ] Audit log (all user actions logged)
- [ ] Tests
    - [ ] Consent tracking
    - [ ] Audit log entries

Module 6: Telegram Bot (Extended)

- [ ] Commands expanded
    - [ ] /tasks — list user's tasks (with inline buttons to complete)
    - [ ] /status — current status, MC balance, next goal
    - [ ] /create — create task (inline form)
    - [ ] /help — command help
    - [ ] /settings — Telegram preferences
- [ ] Integration
    - [ ] Task completion via Telegram (button)
    - [ ] Real-time task notifications
    - [ ] Status change announcements
- [ ] Frontend
    - [ ] Telegram Mini App (web version opens in Telegram)
- [ ] Tests
    - [ ] All commands work
    - [ ] Error handling

Module 7: Gamification (Basic)

- [ ] Database schema
  - [ ] statuses table (id, name, level, requirements JSONB)
  - [ ] user_statuses table (user_id, status_id, achieved_at)
  - [ ] achievements table (id, name, description, icon_url)
  - [ ] user_achievements table (user_id, achievement_id, earned_at)
- [ ] API endpoints
  - [ ] GET /api/gamification/my-status (current status)
  - [ ] GET /api/gamification/status/leaderboard (top 100 by status)
  - [ ] POST /api/gamification/status/calc (Cron job endpoint)
  - [ ] GET /api/gamification/achievements/{userId}
  - [ ] POST /api/gamification/achievements/{userId}/award (Admin)
- [ ] Frontend
  - [ ] Status badge display (in dashboard, profile)
  - [ ] Leaderboard page
  - [ ] Achievements display
  - [ ] Progress to next status
- [ ] Cron job
  - [ ] Daily status recalculation
  - [ ] Achievement checking
- [ ] Status levels (5 levels)
  - [ ] 1. Фотон (New user, 0 tasks, 0 MC)
  - [ ] 2. Топчик (>10 tasks, 100 MC)
  - [ ] 3. Кремень (>50 tasks, 1 year tenure, 500 MC)
  - [ ] 4. Углерод (>100 tasks, KPI 100%+, 2,000 MC)
  - [ ] 5. UNIVERSE (Founder/Director, 10,000 MC)
- [ ] Tests
  - [ ] Status calculation logic
  - [ ] Leaderboard ordering

Quality Assurance (Phase 2)

- [ ] **Integration Tests**
  - [ ] Module 1-7 integration scenarios
  - [ ] Workflow: Register → Create task → Assign → Complete → Earn MC → Status upgrade
- [ ] **Performance Tests**

    - [ ] Load test 100 concurrent users (k6)
    - [ ] API response time <500ms
    - [ ] Database query optimization
- [ ] **Security Tests**
    - [ ] SQL injection tests
    - [ ] XSS tests
    - [ ] CSRF protection
    - [ ] Authorization checks (access other users' data)
- [ ] **Browser Compatibility**
    - [ ] Chrome, Firefox, Safari, Edge
    - [ ] Mobile browsers (iOS Safari, Chrome Android)

Signoff: Module leads, QA Engineer
Status: In Progress

6.4 ФАЗА 3 CHECKLIST (Недели 15-16)

QA & Testing Finalization

- [ ] **Unit Test Coverage**
    - [ ] Backend modules: >70% coverage
    - [ ] npm run test:cov (report)
    - [ ] Critical paths: 100% coverage
- [ ] **Integration Test Suite**
    - [ ] End-to-end user flows tested
    - [ ] All API endpoints tested
    - [ ] Database transactions tested
    - [ ] Rollback on errors working
- [ ] **E2E Tests (Playwright)**
    - [ ] User registration flow
    - [ ] Login flow
    - [ ] Create task → Complete → Earn MC
    - [ ] Status upgrade flow
    - [ ] Telegram Bot commands
    - [ ] At least 5 critical flows automated
    - [ ] Tests passing in CI/CD
- [ ] **Performance Testing (k6)**

    - [ ] Load test: 100 VUs for 5 minutes
    - [ ] Spike test: 500 VUs for 30 sec
    - [ ] Results documented
    - [ ] P95 response time <1s
- [ ] **Security Audit (Internal)**
    - [ ] OWASP Top 10 checklist
    - [ ] SQL injection: NOT vulnerable
    - [ ] XSS: NOT vulnerable
    - [ ] CSRF: Protected
    - [ ] Authentication: Secure
    - [ ] Authorization: Properly enforced
    - [ ] Data encryption: AES-256
    - [ ] JWT: Secure (httpOnly cookies, short expiry)
    - [ ] Dependency vulnerabilities: npm audit clean
    - [ ] Secrets: Not in code (use environment variables)
    - [ ] HTTPS: Enforced
    - [ ] Security headers: Set (HSTS, CSP, X-Frame-Options)
    - [ ] Rate limiting: Implemented
    - [ ] Input validation: All fields validated
- [ ] **Security Audit (External - Pентест)**
    - [ ] Pентест company engaged
    - [ ] Findings addressed (Critical & High)
    - [ ] Report signed off by Security Officer
- [ ] **Database Optimization**
    - [ ] Indexes created for foreign keys
    - [ ] Query plans analyzed
    - [ ] Slow query log reviewed
    - [ ] N+1 queries fixed
- [ ] **Frontend Testing**
    - [ ] All pages render without errors
    - [ ] Forms submit successfully
    - [ ] Error messages display correctly
    - [ ] Responsive on mobile (375px, 768px, 1920px)
    - [ ] Accessibility: axe-core scan
    - [ ] No console errors

Documentation Completion

- [ ] **API Documentation**
  - [ ] Swagger UI accessible at /api/docs
  - [ ] All endpoints documented
  - [ ] Request/response examples
  - [ ] Error codes explained
  - [ ] Authentication described
- [ ] **Database Documentation**
  - [ ] EER diagram (Excalidraw)
  - [ ] Table descriptions
  - [ ] Foreign key relationships
  - [ ] Indexes listed
- [ ] **Architecture Documentation**
  - [ ] System design overview
  - [ ] Component diagram
  - [ ] Data flow diagram
  - [ ] Technology choices documented
- [ ] **Developer Guide**
  - [ ] Setup instructions (macOS, Linux, Windows)
  - [ ] Running tests locally
  - [ ] Code style guide
  - [ ] Git workflow (branch naming, commit messages)
  - [ ] Common tasks (add new endpoint, add new page)
- [ ] **User Manual (PDF)**
  - [ ] Getting started guide
  - [ ] Key features explained
  - [ ] Screenshots with annotations
  - [ ] FAQ section
  - [ ] Troubleshooting
- [ ] **Video Tutorials (3-5 videos)**
  - [ ] Part 1: Registration & login (3 min)
  - [ ] Part 2: Creating tasks (3 min)
  - [ ] Part 3: Dashboard overview (3 min)
  - [ ] Part 4: Telegram Bot (2 min)

    - [ ] Part 5: Compliance (2 min)
    - [ ] Hosted on Loom or YouTube
- [ ] **Change Log**
    - [ ] All commits summarized
    - [ ] Version history (v0.1.0 → v1.0.0)

Pre-Launch Checklist

- [ ] **Production Environment**
    - [ ] Server capacity planned (RAM, CPU, Disk)
    - [ ] Load balancer configured (if multi-server)
    - [ ] Firewall rules set
    - [ ] SSL/TLS certificate valid
    - [ ] DNS records updated
    - [ ] Domain pointing to production
- [ ] **Database Production**
    - [ ] PostgreSQL optimized (shared_buffers, effective_cache_size)
    - [ ] Backups automated (daily, retained 30 days)
    - [ ] Backup tested (restore to test DB)
    - [ ] Connection pooling configured (pgBouncer)
    - [ ] Replication setup (optional, for HA)
- [ ] **Deployment Automation**
    - [ ] GitHub Actions workflow for production
    - [ ] Blue-green deployment configured
    - [ ] Rollback script tested
    - [ ] Health check endpoints
    - [ ] Smoke tests after deployment
- [ ] **Monitoring Live**
    - [ ] Prometheus scraping production
    - [ ] Grafana dashboards updated
    - [ ] Alarms configured and tested
    - [ ] Slack/PagerDuty notifications
    - [ ] Log aggregation from production
    - [ ] Error tracking (Sentry) active
- [ ] **Backup & Disaster Recovery**
    - [ ] Daily backups to S3

    - [ ] Backup restoration tested
    - [ ] RTO (Recovery Time Objective): <1 hour
    - [ ] RPO (Recovery Point Objective): <1 day
    - [ ] Incident response plan documented
- [ ] **Performance Baseline**
    - [ ] Production load test results
    - [ ] P95 latency <500ms
    - [ ] Database query times <100ms (p95)
    - [ ] CPU utilization <70% under normal load
- [ ] **Security Compliance**
    - [ ] 152-ФЗ compliance verified
    - [ ] Penetration testing completed
    - [ ] Vulnerabilities fixed (if any)
    - [ ] Security certificate from auditor
    - [ ] DPA (Data Processing Agreement) signed
    - [ ] Роскомнадзор registration (if required)
- [ ] **Communication Plan**
    - [ ] Stakeholder list identified
    - [ ] Launch announcement prepared
    - [ ] Пилот clients notified
    - [ ] Support team trained
    - [ ] Escalation process documented

Production Deployment

- [ ] **Deployment Day (Friday 3 PM)**
    - [ ] Feature freeze 24 hours before
    - [ ] Final staging tests passed
    - [ ] Database backup taken (production)
    - [ ] Production database migrated (new schema)
    - [ ] Deployment window announced (30-60 min)
    - [ ] Nginx drain connections gracefully
    - [ ] Blue server remains active (current production)
    - [ ] Green server deployed (new version)
    - [ ] Health checks on green server
    - [ ] Switch traffic 10% → green

    - [ ] Monitor errors/performance (5 min)
    - [ ] Shift 50% traffic
    - [ ] Shift 100% traffic
    - [ ] Keep blue server ready for rollback (1 hour)
    - [ ] Post-deployment smoke tests
    - [ ] Announce launch to stakeholders
- [ ] **Monitoring (48 Hours)**
    - [ ] On-call team engaged 24/7
    - [ ] Dashboard watched continuously
    - [ ] Alerts checked every 15 minutes
    - [ ] Error logs reviewed
    - [ ] Performance metrics checked
    - [ ] User feedback collected
    - [ ] Incident response if needed

Post-Launch

- [ ] **Launch Report**
    - [ ] Deployment date & time
    - [ ] Issues encountered (none ideally)
    - [ ] Performance metrics
    - [ ] Lessons learned
- [ ] **Pilot Client Onboarding**
    - [ ] Client account created
    - [ ] Test data loaded
    - [ ] Training session conducted (1-2 hours)
    - [ ] Support contact provided
    - [ ] Go-live date scheduled
- [ ] **Feedback Collection**
    - [ ] Weekly demo sessions with pilot client
    - [ ] Bug report system active
    - [ ] Feature request form
    - [ ] NPS survey (after 1 month)

Signoff: Tech Lead, Product Manager, CTO
Status: □ In Progress

7. ФИНАНСОВЫЕ ПАРАМЕТРЫ

7.1 CAPEX (Capital Expenditure - Первый год)

Категория	Статья	Сумма
Разработка ПО	Salaries (10-12 人 × $25,000 avg)	$250,000
	Contractor/Outsourcing (QA, Security)	$50,000
Инфраструктура	Selectel VPS (12 мес × $50)	$600
	GPU аренда (3 мес × $2,500 для LLM)	$7,500
	Domain + SSL (1 год)	$200
	Monitoring tools (Grafana, Prometheus)	$500
Лицензии	GitHub Enterprise (10 seats × $200)	$2,000
	Atlassian (Confluence, Jira)	$3,000
	Security tools (Snyk, SonarQube)	$2,000
Legal & Compliance	Lawyer consultation (152-ФЗ)	$5,000
	Security audit / Pentesting	$5,000
	Регистрация Роскомнадзор	$500
Tools & Services	Sentry error tracking	$500
	Slack Pro (50+ people)	$1,500
	Design tools (Figma Pro)	$1,500
Other	Training & conferences	$2,000
	Buffer for contingencies	$55,600
ИТОГО		**$390,000**

7.2 OPEX (Operational Expenditure - Первый год)

Категория	Месячно	Годово
Инфраструктура		
- Selectel VPS (2 instances)	$100	$1,200
- PostgreSQL managed (если не self-hosted)	$0	$0
- Redis managed	$0	$0
- S3/Object Storage	$50	$600
- Bandwidth & traffic	$100	$1,200
Итого инфра	$250	$3,000

Категория	Месячно	Годово
LLM Services		
- GigaChat API (если использовать)	$1,500	$18,000
- YandexGPT API	$1,000	$12,000
- Local LLM GPU compute (если оптимизировать)	$500	$6,000
Итого LLM	$3,000	$36,000
Tools & Services		
- Monitoring (Prometheus, Grafana, ELK)	$300	$3,600
- Error tracking (Sentry)	$200	$2,400
- Security scanning	$200	$2,400
- CI/CD (GitHub Actions - included)	$0	$0
Итого tools	$700	$8,400
Support & Operations		
- 24/7 Support team (2 people)	$3,000	$36,000
- DevOps/SRE part-time	$1,000	$12,000
Итого support	$4,000	$48,000
Marketing & Sales (later)		
- Placeholder for future marketing	$2,000	$24,000
ИТОГО OPEX / МЕСЯЦ	**$10,950**	
ИТОГО OPEX / ГОД		**$131,400**

Note: Это для самого первого года разработки + пилота. В последующих годах OPEX возрастет с добавлением support team и LLM затрат.

7.3 Финансовая модель (3 года)

Метрика	Год 1	Год 2	Год 3
CAPEX	$390,000	$50,000	$30,000
OPEX	$131,400	$150,000	$180,000
Total Cost	**$521,400**	**$200,000**	**$210,000**
Cumulative	$521,400	$721,400	$931,400
Revenue (SaaS)	$120,000	$600,000	$1,800,000
Revenue (Consulting)	$0	$100,000	$300,000

Метрика	Год 1	Год 2	Год 3
Total Revenue	$120,000	$700,000	$2,100,000
Profit/Loss	($401,400)	+$500,000	+$1,890,000
Cumulative P&L	($401,400)	+$98,600	+$1,988,600
Break-even	—	**Месяц 18**	—

7.4 Модель доходов (SaaS + Services)

SaaS подписки:

- Базовый: $10/пользователь/месяц (5+ пользователей)
- Профессиональный: $20/пользователь/месяц (20+ функций)
- Корпоративный: $35/пользователь/месяц (все функции + support)

Консалтинг:

- Внедрение: $50,000-150,000 за клиента
- Кастомизация: $150-200/час
- Обучение: $5,000-20,000 за компанию

Дополнительные модули:

- Production Management: +$5/пользователь
- Advanced Analytics: +$10/пользователь

7.5 Привлечение инвестиций

Раунд	Сумма	Разведение	Использование	Timeline
Pre-seed	$100,000	10%	Прототип, MVP	Q4 2025
Seed (Series A)	$500,000	15-20%	Разработка, пилот, go-to-market	Q1 2026
Series B	$2,000,000	20-25%	Масштабирование, маркетинг, team	Q4 2026

Целевое значение компании:

- Год 1: $2,000,000 (на основе Multiples)
- Год 3: $25,000,000 (при условии 10x revenue growth)

8. KPI И МЕТРИКИ УСПЕХА

8.1 Ключевые метрики проекта

Technical KPIs

KPI	Target	Измерение	Ответственный
Uptime	99.5%	Prometheus + Grafana	DevOps
API Response Time (P95)	<500ms	Application monitoring	Backend Lead
Database Query Time (P95)	<100ms	Database monitoring	Tech Lead
Test Coverage	>70%	Code coverage reports	QA
Security Vulnerabilities	0 Critical	Security scanning tools	Security Officer
Bug Rate	<5 bugs per 100 lines	Bug tracking	QA

Product KPIs

KPI	Target	Measurement	Owner
User Adoption (MVP)	80%+ weekly active	Analytics	Product Manager
Feature Usage	70%+ using core features	Analytics	Product Manager
NPS Score	>30	Survey after 1 month	Product Manager
Customer Retention	>80% after 3 months	Churn rate	Product Manager
Average Tasks/User/Month	>20	Database query	Product Manager
Average MC Balance	Growing over time	Database query	Product Manager

Business KPIs

KPI	Target (Year 1)	Measurement	Owner
Active Customers	3-5 pilots	Deal tracking	Sales
MRR (Monthly Recurring Revenue)	$10,000 by month 12	Revenue report	CFO
Customer Acquisition Cost	<$10,000 per customer	Sales tracking	Sales
LTV (Lifetime Value)	>$50,000 per customer	Revenue analysis	CFO
Churn Rate	<5% monthly	Customer tracking	Product
NPS Score	>30	Customer survey	Product

8.2 Метрики команды

Метрика	Target	Measurement
Sprint Velocity	Stabilize after 3 sprints	Story points completed
Cycle Time	<7 days avg	GitHub milestones
Code Review Time	<24 hours	PR metrics
Production Incidents	<2 per month	Incident tracking
Employee Satisfaction	>4/5 avg	Monthly survey

8.3 Go/No-Go Gate Decision Points

Gate 1: End of Phase 1 (Week 8)

Go-live criteria:

- [ ] Backend auth + API working
- [ ] Frontend basic pages rendering
- [ ] Telegram Bot basic commands working
- [ ] Database schema in place
- [ ] No critical security issues
- [ ] Documentation 50% complete

Decision: Proceed to Phase 2 → YES / Pivot → NO

Gate 2: End of Phase 2 (Week 14)

Go-live criteria:

- [ ] All 7 MVP modules feature-complete
- [ ] Unit tests passing (>70% coverage)
- [ ] Integration tests for critical flows
- [ ] Security audit passed
- [ ] Performance tests: <500ms p95
- [ ] Documentation 90% complete
- [ ] Pilot client ready

Decision: Proceed to Phase 3 & launch MVP → YES / Delay → NO

Gate 3: Production Launch (Week 17)

Go-live criteria:

- [ ] All tests passing
- [ ] Security audit cleared
- [ ] External pentesting completed
- [ ] Backup/disaster recovery tested
- [ ] Monitoring live
- [ ] Support team trained
- [ ] Pilot client onboarded
- [ ] Launch communication ready

Decision: Launch to production → GO / Delay → NO-GO

CONCLUSION

MatrixGin v2.0 — это амбициозный проект, требующий:

✓ **Команды:** 10-12 высокопрофессиональных разработчиков
✓ **Времени:** 40-50 недель для полной разработки (9-12 месяцев)
✓ **Финансирования:** $390,000 CAPEX + $131,400 OPEX первый год
✓ **Внимания к деталям:** особенно к 152-ФЗ compliance
✓ **MVP-подхода:** минимум 7 модулей, быстрый запуск, итеративное развитие

Критичные факторы успеха:

1. Сильная техническая команда (особенно Tech Lead)
2. Продуманная архитектура (избежать переделок)
3. Фокус на MVP (не пытаться реализовать все сразу)
4. Пилот-клиент для валидации (1-2 месяца использования)
5. Соответствие 152-ФЗ с первого дня (не добавлять позже)
6. Локальные LLM как основа (экономия + безопасность)

Следующие шаги:

1. ✓ Утвердить этот план у Advisory Board
2. ✓ Подобрать ключевых сотрудников (Tech Lead, Frontend/Backend Leads)
3. ✓ Начать Phase 0 (Infrastructure setup) на неделе 1
4. ✓ Создать Jira backlog для Phase 1-3
5. ✓ Запланировать еженедельные синхронизации

Status: ✓ **READY FOR EXECUTION**

Документ подготовлен: 21 ноября 2025

Версия: 1.0 (Full Implementation Plan + Checklist)

Согласовано: Tech Lead, Product Manager, CTO
[1] [2]

⁂

1. MatrixGin-Architecture-v2.md
2. MatrixGin-Architecture-v2.md