📁 Структура проекта

```
provider-company/
├── .github/                        # GitHub workflows
├── docker/
│   ├── nginx/
│   ├── postgres/
│   └── redis/
├── docs/                           # Документация
├── libs/                           # Общие библиотеки
│   ├── shared-types/               # Общие типы TypeScript
│   ├── ui-components/              # Общие UI компоненты
│   └── utils/                      # Общие утилиты
├── apps/
│   ├── api-gateway/                # API Gateway (BFF)
│   ├── auth-service/               # Сервис аутентификации
│   ├── billing-service/            # Сервис биллинга
│   ├── crm-service/                # CRM система
│   ├── notification-service/       # Уведомления
│   ├── support-service/            # Техподдержка
│   └── monitoring-service/         # Мониторинг сети
├── frontends/
│   ├── admin-panel/                # Админ панель
│   ├── customer-cabinet/           # Личный кабинет
│   └── mobile-app/                 # Мобильное приложение
└── mobile/                         # Нативные мобильные приложения
    ├── ios/
    └── android/
```

🏗 Архитектура

1. Backend (Микросервисы на NestJS)

```
api-gateway/                          # Единая точка входа
├── src/
│   ├── modules/
│   │   ├── auth/                     # Прокси к auth-service
│   │   ├── billing/                  # Прокси к billing-service
│   │   └── profile/                  # Агрегация данных пользователя
│   ├── filters/                      # Глобальные фильтры
│   ├── interceptors/                 # Интерцепторы
│   └── guards/                       # Guards для RBAC
├── Dockerfile
└── package.json

auth-service/                         # Аутентификация и авторизация
├── src/
│   ├── modules/
│   │   ├── users/                    # Управление пользователями
│   │   ├── sessions/                 # Сессии и токены
│   │   ├── roles/                    # Роли и права
│   │   └── mfa/                      # 2FA, TOTP
│   └── entities/                     # Сущности БД
├── Dockerfile
└── package.json

billing-service/                      # Биллинг и платежи
├── src/
│   ├── modules/
│   │   ├── tariffs/                  # Тарифы и пакеты
│   │   ├── invoices/                 # Счета
│   │   ├── payments/                 # Платежи
│   │   ├── transactions/             # Транзакции
│   │   └── reports/                  # Отчеты
│   └── entities/
├── Dockerfile
└── package.json
```

2. Frontend: Личный кабинет (Vue 3 + TypeScript + Pinia)

```
customer-cabinet/
├── src/
│   ├── api/                          # API клиенты
│   │   ├── auth.ts
│   │   ├── billing.ts
│   │   └── profile.ts
│   ├── assets/
│   ├── components/
│   │   ├── ui/                       # Базовые UI компоненты
│   │   ├── layout/                   # Компоненты layout
│   │   ├── billing/                  # Компоненты биллинга
│   │   ├── profile/                  # Компоненты профиля
│   │   └── support/                  # Компоненты поддержки
│   ├── composables/                  # Composables Vue 3
│   ├── layouts/                      # Layout приложения
│   ├── router/                       # Маршрутизация
│   ├── stores/                       # Pinia stores
│   │   ├── auth.store.ts
│   │   ├── billing.store.ts
│   │   ├── profile.store.ts
│   │   └── ui.store.ts
│   ├── types/                        # Типы TypeScript
│   ├── views/                        # Страницы
│   │   ├── Dashboard.vue
│   │   ├── Billing/
│   │   ├── Profile/
│   │   ├── Support/
│   │   └── Services/
│   └── utils/                        # Утилиты
├── Dockerfile
└── vite.config.ts
```

3. Мобильное приложение (React Native + TypeScript)

```
mobile-app/
├── src/
│   ├── api/                        # API клиенты
│   ├── assets/                     # Иконки, шрифты
│   ├── components/                 # Компоненты
│   │   ├── common/
│   │   ├── billing/
│   │   ├── profile/
│   │   └── ui/
│   ├── navigation/                 # Навигация
│   │   ├── AppNavigator.tsx
│   │   ├── types.ts
│   │   └── screens/
│   ├── screens/                    # Экраны
│   │   ├── Auth/
│   │   ├── Main/
│   │   ├── Billing/
│   │   ├── Profile/
│   │   └── Support/
│   ├── store/                      # Zustand/MobX
│   │   ├── auth.store.ts
│   │   ├── billing.store.ts
│   │   └── profile.store.ts
│   ├── hooks/                      # Кастомные хуки
│   ├── utils/                      # Утилиты
│   └── types/                      # Типы TypeScript
├── android/
├── ios/
├── app.json
└── package.json
```

Технологический стек

Backend:

- **NestJS** - фреймворк для Node.js
- **TypeORM/Prisma** - ORM
- **PostgreSQL** - основная БД
- **Redis** - кэш и сессии
- **RabbitMQ/Kafka** - message queue
- **JWT** - аутентификация
- **Swagger** - документация API
- **Prometheus + Grafana** - мониторинг

Frontend (Личный кабинет):

- **Vue 3** + Composition API
- **TypeScript**
- **Pinia** - state management
- **Vue Router** - маршрутизация
- **Vuetify/Quasar/Element Plus** - UI компоненты
- **Vite** - сборка
- **Axios** - HTTP клиент
- **Vue i18n** - интернационализация

Мобильное приложение:

- **React Native** (или Flutter)
- **TypeScript**
- **Zustand/Redux Toolkit** - state management
- **React Navigation** - навигация
- **React Query** - управление запросами
- **Push Notifications** (Firebase/OneSignal)

База данных

```
-- Пример основных таблиц
users (
  id UUID PRIMARY KEY,
  email VARCHAR(255) UNIQUE,
  phone VARCHAR(20),
  password_hash VARCHAR(255),
  is_active BOOLEAN DEFAULT true,
  role VARCHAR(50) DEFAULT 'customer',
  created_at TIMESTAMP
);

customers (
  id UUID PRIMARY KEY REFERENCES users(id),
  full_name VARCHAR(255),
  address JSONB,
  contract_number VARCHAR(50),
  balance DECIMAL(10,2) DEFAULT 0
);

tariffs (
  id UUID PRIMARY KEY,
  name VARCHAR(100),
  speed_down INTEGER, -- Мбит/с
  speed_up INTEGER,
  price DECIMAL(10,2),
  is_active BOOLEAN DEFAULT true
);

invoices (
  id UUID PRIMARY KEY,
  customer_id UUID REFERENCES customers(id),
  amount DECIMAL(10,2),
  status VARCHAR(20), -- pending, paid, cancelled
  due_date DATE,
  paid_at TIMESTAMP
);

tickets (
  id UUID PRIMARY KEY,
  customer_id UUID REFERENCES customers(id),
  subject VARCHAR(255),
  description TEXT,
  status VARCHAR(20), -- open, in_progress, resolved
  priority VARCHAR(20), -- low, medium, high, critical
  created_at TIMESTAMP
);
```

Docker Compose для разработки

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: provider_db
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: secure_password
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine

  api-gateway:
    build: ./apps/api-gateway
    ports:
      - "3000:3000"
    depends_on:
      - postgres
      - redis
    environment:
      DATABASE_URL: postgresql://admin:secure_password@postgres:5432/provider_db
      REDIS_URL: redis://redis:6379

  customer-cabinet:
    build: ./frontends/customer-cabinet
    ports:
      - "8080:8080"
    depends_on:
      - api-gateway

  admin-panel:
    build: ./frontends/admin-panel
    ports:
      - "8081:8081"
    depends_on:
      - api-gateway

volumes:
  postgres_data:
```

Безопасность

1. **JWT токены** с коротким временем жизни
2. **Refresh tokens** с rotation
3. **Rate limiting** на API Gateway
4. **CORS** политики
5. **Хеширование паролей** (bcrypt/argon2)
6. **2FA** для администраторов
7. **Audit log** всех действий

Основные модули личного кабинета:

1. **Авторизация** - вход, регистрация, восстановление пароля
2. **Профиль** - личные данные, смена пароля, 2FA
3. **Биллинг** - баланс, история платежей, счета, тарифы
4. **Услуги** - подключенные услуги, заявки на подключение
5. **Техподдержка** - тикеты, чат, FAQ
6. **Уведомления** - рассылки, push-уведомления
7. **Документы** - договоры, акты, отчеты
8. **Настройки** - уведомления, приватность

Нужна более детальная структура какого-то конкретного модуля?