🏗️ Детальная структура бекенда

```
backend/
├── packages/                          # Монорепозиторий (или отдельные репозитории)
│   ├── libs/
│   │   ├── common/                    # Общие модули
│   │   │   ├── src/
│   │   │   │   ├── decorators/        # Кастомные декораторы
│   │   │   │   ├── filters/           # Глобальные фильтры
│   │   │   │   ├── guards/            # Guards (JWT, Roles)
│   │   │   │   ├── interceptors/      # Интерцепторы
│   │   │   │   ├── pipes/             # Валидаторы
│   │   │   │   ├── utils/             # Утилиты
│   │   │   │   └── types/             # Общие типы
│   │   │   └── package.json
│   │   ├── database/                  # Общая БД логика
│   │   │   ├── src/
│   │   │   │   ├── entities/          # Общие сущности
│   │   │   │   ├── migrations/        # Миграции
│   │   │   │   ├── repositories/      # Репозитории
│   │   │   │   └── seeds/             # Сиды
│   │   │   └── package.json
│   │   └── messaging/                 # RabbitMQ/Kafka клиент
│   │       ├── src/
│   │       │   ├── producers/         # Продюсеры
│   │       │   ├── consumers/         # Консьюмеры
│   │       │   └── interfaces/
│   │       └── package.json
│   │
│   └── services/                      # Микросервисы
│       ├── api-gateway/               # API Gateway (BFF)
│       ├── auth-service/              # Сервис аутентификации
│       ├── billing-service/           # Сервис биллинга
│       ├── customer-service/          # Сервис клиентов
│       ├── notification-service/      # Сервис уведомлений
│       ├── support-service/           # Техподдержка
│       ├── monitoring-service/        # Мониторинг сети
│       └── analytics-service/         # Аналитика
```

🏗️ Детальная структура каждого сервиса

1. API Gateway (BFF - Backend For Frontend)

```
api-gateway/
├── src/
│   ├── common/                        # Общее для gateway
│   │   ├── filters/
│   │   ├── interceptors/
│   │   └── middleware/
│   ├── health/                        # Health checks
│   ├── modules/
│   │   ├── auth/                      # Прокси к auth-service
│   │   │   ├── controllers/
│   │   │   │   ├── auth.controller.ts
│   │   │   │   ├── sessions.controller.ts
│   │   │   │   └── mfa.controller.ts
│   │   │   ├── dto/
│   │   │   │   ├── login.dto.ts
│   │   │   │   ├── register.dto.ts
│   │   │   │   └── mfa.dto.ts
│   │   │   ├── guards/
│   │   │   │   └── jwt-auth.guard.ts
│   │   │   ├── strategies/
```

```
│   │   │   │   └── jwt.strategy.ts
│   │   │   └── auth.module.ts
│   │   │
│   │   ├── billing/                 # Агрегация данных биллинга
│   │   │   ├── controllers/
│   │   │   │   ├── invoices.controller.ts
│   │   │   │   ├── payments.controller.ts
│   │   │   │   └── tariffs.controller.ts
│   │   │   ├── dto/
│   │   │   │   ├── create-invoice.dto.ts
│   │   │   │   └── process-payment.dto.ts
│   │   │   ├── clients/             # HTTP клиенты к сервисам
│   │   │   │   └── billing.client.ts
│   │   │   └── billing.module.ts
│   │   │
│   │   ├── customer/                # Профиль клиента
│   │   │   ├── controllers/
│   │   │   │   ├── profile.controller.ts
│   │   │   │   ├── contracts.controller.ts
│   │   │   │   └── services.controller.ts
│   │   │   ├── dto/
│   │   │   │   ├── update-profile.dto.ts
│   │   │   │   └── change-password.dto.ts
│   │   │   ├── clients/
│   │   │   │   └── customer.client.ts
│   │   │   └── customer.module.ts
│   │   │
│   │   ├── support/                 # Техподдержка
│   │   └── admin/                   # Админские эндпоинты
│   │
│   ├── interceptors/                # Глобальные интерцепторы
│   │   ├── logging.interceptor.ts
│   │   ├── transform.interceptor.ts
│   │   └── timeout.interceptor.ts
│   ├── filters/                     # Глобальные фильтры
│   │   ├── http-exception.filter.ts
│   │   └── validation.filter.ts
│   ├── decorators/                  # Кастомные декораторы
│   │   ├── roles.decorator.ts
│   │   ├── user.decorator.ts
│   │   └── public.decorator.ts
│   └── main.ts
├── test/                            # E2E тесты
├── Dockerfile
├── docker-compose.yml
├── package.json
└── tsconfig.json
```

2. Auth Service

```
auth-service/
├── src/
│   ├── modules/
│   │   ├── users/                    # Управление пользователями
│   │   │   ├── entities/
│   │   │   │   └── user.entity.ts
│   │   │   ├── repositories/
│   │   │   │   └── user.repository.ts
│   │   │   ├── dto/
│   │   │   │   ├── create-user.dto.ts
│   │   │   │   └── update-user.dto.ts
│   │   │   ├── services/
│   │   │   │   └── user.service.ts
│   │   │   ├── controllers/
│   │   │   │   └── user.controller.ts
│   │   │   └── users.module.ts
│   │   │
│   │   ├── auth/                     # Аутентификация
│   │   │   ├── services/
│   │   │   │   ├── auth.service.ts
│   │   │   │   ├── password.service.ts
│   │   │   │   └── token.service.ts
│   │   │   ├── controllers/
│   │   │   │   ├── auth.controller.ts
│   │   │   │   └── logout.controller.ts
│   │   │   ├── guards/
│   │   │   │   └── local-auth.guard.ts
│   │   │   ├── strategies/
│   │   │   │   ├── local.strategy.ts
│   │   │   │   └── jwt.strategy.ts
│   │   │   └── auth.module.ts
│   │   │
│   │   ├── sessions/                 # Управление сессиями
│   │   │   ├── entities/
│   │   │   │   └── session.entity.ts
│   │   │   ├── services/
│   │   │   │   └── session.service.ts
│   │   │   └── sessions.module.ts
│   │   │
│   │   ├── mfa/                      # Двухфакторная аутентификация
│   │   │   ├── services/
│   │   │   │   └── mfa.service.ts
│   │   │   ├── controllers/
│   │   │   │   └── mfa.controller.ts
│   │   │   └── mfa.module.ts
│   │   │
│   │   └── roles/                    # Роли и разрешения
│   │       ├── entities/
│   │       │   ├── role.entity.ts
│   │       │   └── permission.entity.ts
│   │       ├── services/
│   │       │   └── role.service.ts
│   │       └── roles.module.ts
│   │
│   ├── config/                       # Конфигурация
│   │   ├── database.config.ts
│   │   ├── jwt.config.ts
│   │   └── redis.config.ts
│   └── main.ts
├── migrations/                       # Миграции БД
├── seeds/                            # Сиды
└── package.json
```

3. Billing Service

```
billing-service/
├── src/
│   ├── modules/
│   │   ├── tariffs/                 # Тарифные планы
│   │   │   ├── entities/
│   │   │   │   └── tariff.entity.ts
│   │   │   ├── dto/
│   │   │   │   ├── create-tariff.dto.ts
│   │   │   │   └── update-tariff.dto.ts
│   │   │   ├── services/
│   │   │   │   └── tariff.service.ts
│   │   │   ├── controllers/
│   │   │   │   └── tariff.controller.ts
│   │   │   └── tariffs.module.ts
│   │   │
│   │   ├── invoices/                # Счета
│   │   │   ├── entities/
│   │   │   │   └── invoice.entity.ts
│   │   │   ├── services/
│   │   │   │   ├── invoice.service.ts
│   │   │   │   └── invoice-generator.service.ts
│   │   │   ├── controllers/
│   │   │   │   └── invoice.controller.ts
│   │   │   └── invoices.module.ts
│   │   │
│   │   ├── payments/                # Платежи
│   │   │   ├── entities/
│   │   │   │   └── payment.entity.ts
│   │   │   ├── services/
│   │   │   │   ├── payment.service.ts
│   │   │   │   ├── payment-processor.service.ts
│   │   │   │   └── payment-gateway.service.ts
│   │   │   ├── controllers/
│   │   │   │   └── payment.controller.ts
│   │   │   └── payments.module.ts
│   │   │
│   │   ├── transactions/            # Транзакции
│   │   │   ├── entities/
│   │   │   │   └── transaction.entity.ts
│   │   │   ├── services/
│   │   │   │   └── transaction.service.ts
│   │   │   └── transactions.module.ts
│   │   │
│   │   ├── subscriptions/           # Подписки клиентов
│   │   │   ├── entities/
│   │   │   │   └── subscription.entity.ts
│   │   │   ├── services/
│   │   │   │   └── subscription.service.ts
│   │   │   └── subscriptions.module.ts
│   │   │
│   │   └── reports/                 # Отчеты
│   │       ├── services/
│   │       │   └── report.service.ts
│   │       └── reports.module.ts
│   │
│   ├── cron/                        # Периодические задачи
│   │   ├── invoice-cron.service.ts  # Генерация счетов
│   │   ├── payment-cron.service.ts  # Списание платежей
│   │   └── subscription-cron.service.ts
│   └── main.ts
```

4. Customer Service

```
customer-service/
├── src/
│   ├── modules/
│   │   ├── profiles/                # Профили клиентов
│   │   │   ├── entities/
│   │   │   │   └── customer.entity.ts
│   │   │   ├── services/
│   │   │   │   └── customer.service.ts
│   │   │   └── profiles.module.ts
│   │   │
│   │   ├── contracts/               # Договоры
│   │   │   ├── entities/
│   │   │   │   └── contract.entity.ts
│   │   │   └── contracts.module.ts
│   │   │
│   │   ├── addresses/               # Адреса подключения
│   │   │   ├── entities/
│   │   │   │   └── address.entity.ts
│   │   │   └── addresses.module.ts
│   │   │
│   │   ├── devices/                 # Оборудование клиентов
│   │   │   ├── entities/
│   │   │   │   ├── device.entity.ts
│   │   │   │   └── device-model.entity.ts
│   │   │   └── devices.module.ts
│   │   │
│   │   └── applications/            # Заявки
│   │       ├── entities/
│   │       │   └── application.entity.ts
│   │       └── applications.module.ts
│   └── main.ts
```

5. Notification Service

```
notification-service/
├── src/
│   ├── modules/
│   │   ├── notifications/            # Уведомления
│   │   │   ├── entities/
│   │   │   │   └── notification.entity.ts
│   │   │   ├── services/
│   │   │   │   ├── notification.service.ts
│   │   │   │   └── notification-dispatcher.service.ts
│   │   │   └── notifications.module.ts
│   │   │
│   │   ├── channels/                 # Каналы отправки
│   │   │   ├── email/
│   │   │   │   └── email.service.ts
│   │   │   ├── sms/
│   │   │   │   └── sms.service.ts
│   │   │   ├── push/
│   │   │   │   └── push.service.ts
│   │   │   └── channels.module.ts
│   │   │
│   │   ├── templates/                # Шаблоны
│   │   │   ├── entities/
│   │   │   │   └── template.entity.ts
│   │   │   └── templates.module.ts
│   │   │
│   │   └── events/                   # Обработка событий
│   │       ├── handlers/
│   │       │   ├── billing-events.handler.ts
│   │       │   └── support-events.handler.ts
│   │       └── events.module.ts
│   └── main.ts
```

🔧 Примеры основных сущностей

User Entity (auth-service):

```
// packages/libs/database/src/entities/user.entity.ts
@Entity('users')
export class User {
  @PrimaryGeneratedColumn('uuid')
  id: string;

  @Column({ unique: true })
  email: string;

  @Column({ nullable: true, unique: true })
  phone: string;

  @Column()
  passwordHash: string;

  @Column({ default: true })
  isActive: boolean;

  @Column({ default: 'customer' })
  role: UserRole; // 'customer', 'operator', 'admin', 'tech'

  @Column({ type: 'jsonb', nullable: true })
  mfaSettings: {
    enabled: boolean;
    secret?: string;
    backupCodes?: string[];
  };

  @CreateDateColumn()
  createdAt: Date;

  @UpdateDateColumn()
  updatedAt: Date;

  // Relations
  @OneToOne(() => Customer, customer => customer.user)
  customer: Customer;

  @OneToMany(() => Session, session => session.user)
  sessions: Session[];
}
```

Invoice Entity (billing-service):

```
// billing-service/src/modules/invoices/entities/invoice.entity.ts
@Entity('invoices')
export class Invoice {
  @PrimaryGeneratedColumn('uuid')
  id: string;

  @Column()
  customerId: string;

  @Column('decimal', { precision: 10, scale: 2 })
  amount: number;

  @Column({ default: 'pending' })
  status: InvoiceStatus; // 'pending', 'paid', 'cancelled', 'overdue'

  @Column()
  dueDate: Date;

  @Column({ nullable: true })
  paidAt: Date;

  @Column({ type: 'jsonb' })
  items: InvoiceItem[];

  @CreateDateColumn()
  createdAt: Date;
}
```

Взаимодействие сервисов

Event-Driven Architecture:

```
// packages/libs/messaging/src/producers/billing.producer.ts
@Injectable()
export class BillingEventProducer {
  constructor(private readonly rabbitmqService: RabbitMQService) {}

  async invoiceCreated(invoice: Invoice) {
    await this.rabbitmqService.publish('billing.invoice.created', {
      eventId: uuidv4(),
      timestamp: new Date(),
      data: {
        invoiceId: invoice.id,
        customerId: invoice.customerId,
        amount: invoice.amount,
        dueDate: invoice.dueDate,
      },
    });
  }
}

// notification-service/src/modules/events/handlers/billing-events.handler.ts
@RabbitMQSubscribe('billing.invoice.created')
export class BillingEventsHandler {
  constructor(private readonly notificationService: NotificationService) {}

  async handleInvoiceCreated(event: InvoiceCreatedEvent) {
    await this.notificationService.sendInvoiceNotification(
      event.data.customerId,
      event.data.invoiceId,
    );
  }
}
```

Docker Compose для всех сервисов

```
version: '3.8'

services:
  # Базы данных
  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: provider
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    command: redis-server --appendonly yes

  rabbitmq:
    image: rabbitmq:3-management
    environment:
      RABBITMQ_DEFAULT_USER: ${RABBITMQ_USER}
      RABBITMQ_DEFAULT_PASS: ${RABBITMQ_PASSWORD}

  # Микросервисы
  api-gateway:
    build: ./services/api-gateway
    ports:
      - "3000:3000"
    environment:
      NODE_ENV: development
```

```
      NODE_ENV: development
      DATABASE_URL: postgresql://admin:${DB_PASSWORD}@postgres:5432/provider
      REDIS_URL: redis://redis:6379
      AUTH_SERVICE_URL: http://auth-service:3001
      BILLING_SERVICE_URL: http://billing-service:3002
    depends_on:
      - postgres
      - redis
      - auth-service
      - billing-service

  auth-service:
    build: ./services/auth-service
    environment:
      DATABASE_URL: postgresql://admin:${DB_PASSWORD}@postgres:5432/auth_db
      JWT_SECRET: ${JWT_SECRET}
      JWT_EXPIRES_IN: 3600
    depends_on:
      - postgres
      - redis

  billing-service:
    build: ./services/billing-service
    environment:
      DATABASE_URL: postgresql://admin:${DB_PASSWORD}@postgres:5432/billing_db
      RABBITMQ_URL: amqp://${RABBITMQ_USER}:${RABBITMQ_PASSWORD}@rabbitmq
    depends_on:
      - postgres
      - rabbitmq

  # Мониторинг
  prometheus:
    image: prom/prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./monitoring/prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana
    ports:
      - "3001:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD: admin

volumes:
  postgres_data:
```

📊 Мониторинг и логирование

```
backend/
├── monitoring/
│   ├── prometheus.yml               # Конфиг Prometheus
│   ├── grafana/
│   │   ├── dashboards/              # Дашборды Grafana
│   │   └── datasources/             # Источники данных
│   └── alerts/                      # Alert правила
│
├── logs/                            # Централизованные логи
│   ├── api-gateway/
│   ├── auth-service/
│   └── billing-service/
│
└── tracing/                         # Distributed tracing
    ├── jaeger/
    └── opentelemetry/
```

Нужны примеры кода для конкретных модулей или более детальная структура какого-то сервиса?