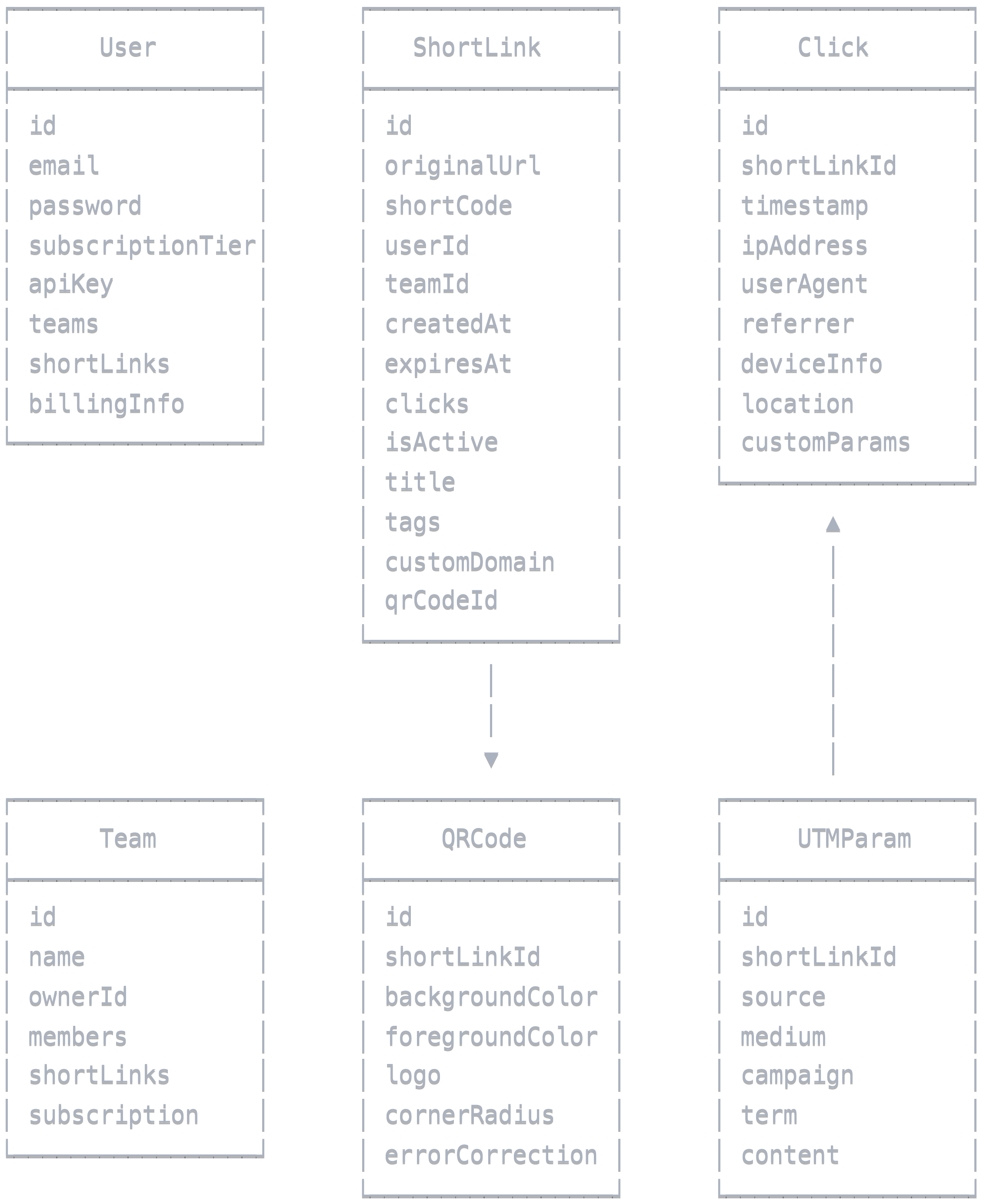
ShortLink Pro: Enterprise URL Shortener Architecture

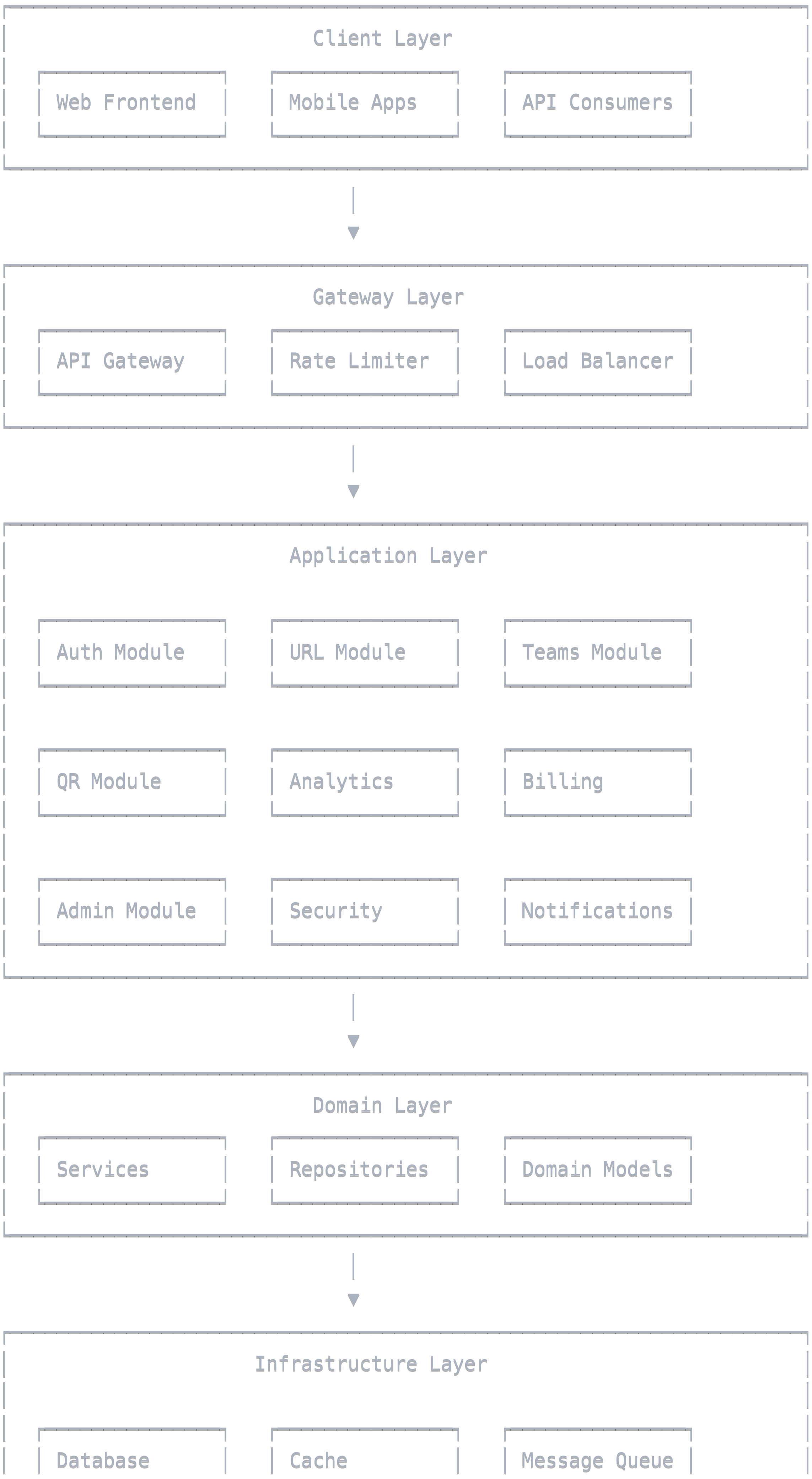
System Overview

ShortLink Pro is an enterprise-grade URL shortener with advanced features designed for businesses and power users. The system leverages NestJS to create a modular, scalable, and maintainable backend architecture.

Core Domain Model



System Architecture


Client Layer
Web Frontend
Mobile Apps
API Consumers
Gateway Layer
API Gateway
Rate Limiter
Load Balancer
Application Layer
Auth Module
URL Module
Teams Module
QR Module
Analytics
Billing
Admin Module
Security
Notifications
Domain Layer
Services
Repositories
Domain Models
Infrastructure Layer
Database
Cache
Message Queue

```
│  └──────────────┘  └──────────────┘  └──────────────┘         │
│                                                                │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐         │
│  │ External APIs│  │ File Storage │  │ Search Engine│         │
│  └──────────────┘  └──────────────┘  └──────────────┘         │
└────────────────────────────────────────────────────────────────┘
```

NestJS Module Structure

```
src/
├── main.ts
├── app.module.ts
├── shared/
│   ├── guards/
│   ├── interceptors/
│   ├── filters/
│   ├── pipes/
│   ├── decorators/
│   └── utils/
├── config/
│   ├── configuration.ts
│   ├── validation.schema.ts
│   └── env.validation.ts
├── common/
│   ├── dtos/
│   ├── interfaces/
│   ├── constants/
│   └── enums/
├── modules/
│   ├── auth/
│   │   ├── auth.module.ts
│   │   ├── auth.service.ts
│   │   ├── auth.controller.ts
│   │   ├── strategies/
│   │   ├── guards/
│   │   └── dtos/
│   ├── users/
│   │   ├── users.module.ts
│   │   ├── users.service.ts
│   │   ├── users.controller.ts
│   │   ├── entities/
│   │   ├── repositories/
│   │   └── dtos/
│   ├── short-links/
│   │   ├── short-links.module.ts
│   │   ├── short-links.service.ts
│   │   ├── short-links.controller.ts
│   │   ├── entities/
│   │   ├── repositories/
│   │   └── dtos/
│   ├── qr-codes/
│   │   ├── qr-codes.module.ts
│   │   ├── qr-codes.service.ts
│   │   ├── qr-codes.controller.ts
│   │   ├── entities/
│   │   └── dtos/
│   ├── analytics/
│   │   ├── analytics.module.ts
│   │   ├── analytics.service.ts
│   │   ├── analytics.controller.ts
│   │   ├── entities/
│   │   └── dtos/
│   ├── teams/
│   │   ├── teams.module.ts
│   │   ├── teams.service.ts
│   │   ├── teams.controller.ts
│   │   ├── entities/
│   │   └── dtos/
│   ├── billing/
│   │   ├── billing.module.ts
│   │   ├── billing.service.ts
│   │   ├── billing.controller.ts
│   │   ├── entities/
│   │   ├── strategies/
│   │   └── dtos/
│   ├── redirects/
│   │   ├── redirects.module.ts
│   │   ├── redirects.service.ts
│   │   └── redirects.controller.ts
│   └── admin/
│       ├── admin.module.ts
│       ├── admin.service.ts
│       └── admin.controller.ts
└── jobs/
    ├── clean-expired-links.job.ts
    ├── generate-reports.job.ts
    └── send-notifications.job.ts
```

Detailed Module Descriptions

1. Authentication Module

The authentication system will support multiple strategies:

- **JWT-based Authentication**: For session management and API access
- **OAuth2 Integration**: Support for Google, GitHub, Microsoft, etc.
- **API Key Authentication**: For programmatic access to the API
- **Two-Factor Authentication**: SMS or authenticator app integration
- **RBAC (Role-Based Access Control)**: Different permission levels

typescript

```typescript
// auth.module.ts
@Module({
  imports: [
    PassportModule,
    JwtModule.registerAsync({
      imports: [ConfigModule],
      useFactory: async (configService: ConfigService) => ({
        secret: configService.get<string>('JWT_SECRET'),
        signOptions: { expiresIn: '1d' },
      }),
      inject: [ConfigService],
    }),
    UsersModule,
    MailModule,
  ],
  controllers: [AuthController],
  providers: [
    AuthService,
    JwtStrategy,
    LocalStrategy,
    GoogleStrategy,
    GithubStrategy,
    ApiKeyStrategy,
    TwoFactorAuthService,
    RoleGuard,
  ],
  exports: [AuthService],
})
export class AuthModule {}
```

2. URL Shortening Module

The core functionality with advanced features:

- **Custom short codes**: Allow users to create branded links
- **Bulk URL shortening**: Create multiple short links at once
- **Link expiration**: Set links to expire after a certain time
- **Password protection**: Secure links with password access
- **Deep linking**: Support for mobile app deep links
- **UTM parameter management**: Built-in campaign tracking
- **Link rotation**: Change destination URLs based on rules
- **A/B testing**: Split traffic between multiple destinations

typescript

```typescript
// short-links.service.ts
@Injectable()
export class ShortLinksService {
  constructor(
    @InjectRepository(ShortLink)
    private shortLinkRepository: Repository<ShortLink>,
    private cacheManager: Cache,
    private configService: ConfigService,
    private eventEmitter: EventEmitter2,
  ) {}

  async create(createShortLinkDto: CreateShortLinkDto, user: User): Promise<ShortLink>
    const { originalUrl, customCode, expiresAt, password, ...rest } = createShortLinkD

    // Validate URL
    if (!this.isValidUrl(originalUrl)) {
      throw new BadRequestException('Invalid URL provided');
    }

    // Generate or validate custom code
    const shortCode = customCode || await this.generateUniqueCode();

    if (customCode && await this.shortLinkExists(customCode)) {
      throw new ConflictException('Custom code already in use');
    }

    // Create link with encryption for password if provided
    const shortLink = this.shortLinkRepository.create({
      originalUrl,
      shortCode,
      userId: user.id,
      expiresAt,
      password: password ? await bcrypt.hash(password, 10) : null,
      ...rest
    });

    await this.shortLinkRepository.save(shortLink);

    // Cache the short link for faster redirects
    await this.cacheManager.set(
      `shortlink:${shortCode}`,
      { originalUrl, password: !!password },
      { ttl: 3600 }
    );

    // Emit event for analytics
    this.eventEmitter.emit('shortlink.created', shortLink);

    return shortLink;
  }

  // More methods for managing links...
}
```

3. QR Code Module

Advanced QR code generation and customization:

- **Custom design**: Colors, logo integration, shape customization
- **Dynamic QR codes**: Update destination without changing the QR code
- **Analytics tracking**: Track scans and engagement
- **Error correction levels**: Control scanning reliability vs density
- **Export formats**: SVG, PNG, PDF with various resolutions
- **Batch generation**: Create multiple QR codes for campaigns

typescript

```
// qr-codes.service.ts
@Injectable()
export class QrCodesService {
  constructor(
    @InjectRepository(QrCode)
    private qrCodeRepository: Repository<QrCode>,
    private shortLinksService: ShortLinksService,
    private storageService: StorageService,
  ) {}

  async generate(createQrDto: CreateQrCodeDto, user: User): Promise<QrCode> {
    const { shortLinkId, backgroundColor, foregroundColor, logo, cornerRadius, errorCo

    // Verify the short link belongs to the user
    const shortLink = await this.shortLinksService.findOne(shortLinkId, user);

    // Generate QR code with customizations
    const qrCodeData = await this.generateQrCodeImage({
      data: `${this.configService.get('BASE_URL')}/${shortLink.shortCode}`,
      backgroundColor: backgroundColor || '#FFFFFF',
      foregroundColor: foregroundColor || '#000000',
      logoPath: logo ? await this.processLogo(logo) : null,
      cornerRadius: cornerRadius || 0,
      errorCorrectionLevel: errorCorrection || 'M',
    });

    // Store QR code image
    const imageUrl = await this.storageService.uploadFile(
      qrCodeData,
      `qr-codes/${shortLink.id}/${uuidv4()}.png`,
      'image/png',
    );

    // Create QR code record
    const qrCode = this.qrCodeRepository.create({
      shortLinkId,
      backgroundColor,
      foregroundColor,
      logo: logo ? true : false,
      cornerRadius,
      errorCorrection,
      imageUrl,
    });

    await this.qrCodeRepository.save(qrCode);
    return qrCode;
  }

  // More QR code management methods...
}
```

4. Analytics Module

Comprehensive tracking and analysis:

- **Real-time click tracking**: Monitor link performance instantly
- **Geographic data**: Track visitor locations down to city level
- **Device analytics**: Browser, OS, device type information
- **Referrer tracking**: See where traffic is coming from
- **Custom parameters**: Track additional data with custom fields
- **Conversion tracking**: Monitor goal completions
- **Heatmaps**: Visual representation of click patterns
- **Reporting**: Automated reports and dashboards

typescript

```typescript
// analytics.service.ts
@Injectable()
export class AnalyticsService {
  constructor(
    @InjectRepository(Click)
    private clickRepository: Repository<Click>,
    private readonly elasticsearchService: ElasticsearchService,
    private readonly geoIpService: GeoIpService,
    private readonly userAgentService: UserAgentService,
  ) {}

  async trackClick(shortCode: string, req: Request): Promise<void> {
    const ipAddress = this.extractIpAddress(req);
    const userAgent = req.headers['user-agent'];
    const referrer = req.headers.referer || req.headers.referrer;

    // Parse user agent information
    const deviceInfo = this.userAgentService.parse(userAgent);

    // Get geo location from IP
    const location = await this.geoIpService.lookup(ipAddress);

    // Extract custom UTM parameters if present
    const customParams = this.extractUtmParameters(req.query);

    // Create click record
    const click = this.clickRepository.create({
      shortLinkId: shortCode,
      timestamp: new Date(),
      ipAddress,
      userAgent,
      referrer,
      deviceInfo,
      location,
      customParams,
    });

    // Save click to database
    await this.clickRepository.save(click);

    // Index in Elasticsearch for faster analytics queries
    await this.elasticsearchService.index({
      index: 'clicks',
      body: click,
    });
  }
  // Advanced analytics methods...
}
```

5. Team Collaboration Module

Enterprise team features:

- **Team management**: Create and manage teams
- **Role-based permissions**: Admin, editor, viewer roles
- **Shared links**: Collaborate on link management
- **Activity logs**: Track team member actions
- **Workspaces**: Organize links by projects or campaigns
- **Comments & Notes**: Add context to links for team members
- **Approval workflows**: Set up approval process for new links

```typescript
// teams.module.ts
@Module({
  imports: [
    TypeOrmModule.forFeature([Team, TeamMember, TeamInvitation]),
    UsersModule,
    NotificationsModule,
    forwardRef(() => ShortLinksModule),
  ],
  controllers: [TeamsController],
  providers: [
    TeamsService,
    TeamMembersService,
    TeamInvitationsService,
    TeamPermissionsGuard,
  ],
  exports: [TeamsService, TeamMembersService],
})
export class TeamsModule {}
```

6. Billing and Subscription Module

Monetization features:

- **Subscription tiers**: Free, Pro, Business, Enterprise
- **Usage-based billing**: Pay per click or features used
- **Payment processing**: Integration with Stripe, PayPal

- **Invoicing**: Generate and manage invoices
- **Trial periods**: Free trials with expiration
- **White-label options**: Remove branding for premium tiers
- **Custom domains**: Allow custom domains for branded links

```typescript
// billing.module.ts
@Module({
  imports: [
    TypeOrmModule.forFeature([Subscription, Payment, Invoice, PaymentMethod]),
    HttpModule,
    ConfigModule,
    UsersModule,
    NotificationsModule,
  ],
  controllers: [BillingController, WebhooksController],
  providers: [
    BillingService,
    StripeService,
    PayPalService,
    InvoiceService,
    SubscriptionPlanService,
    {
      provide: 'PAYMENT_PROVIDERS',
      useFactory: (configService: ConfigService) => {
        return [
          new StripePaymentProvider(configService.get('STRIPE_SECRET_KEY')),
          new PayPalPaymentProvider({
            clientId: configService.get('PAYPAL_CLIENT_ID'),
            clientSecret: configService.get('PAYPAL_CLIENT_SECRET'),
          }),
        ];
      },
      inject: [ConfigService],
    },
  ],
  exports: [BillingService, SubscriptionPlanService],
})
export class BillingModule {}
```

7. Security Module

Advanced security features:

- **Rate limiting**: Prevent abuse of the API

- **DDOS protection**: Cloudflare integration
- **Link scanning**: Check destination URLs for malware/phishing
- **IP blocking**: Block suspicious traffic sources
- **Audit logging**: Track security-related events
- **Content policy**: Control link destination types
- **Compliance features**: GDPR, CCPA settings

typescript

```typescript
// security.module.ts
@Module({
  imports: [
    ThrottlerModule.forRootAsync({
      imports: [ConfigModule],
      inject: [ConfigService],
      useFactory: (config: ConfigService) => ({
        ttl: config.get('THROTTLE_TTL'),
        limit: config.get('THROTTLE_LIMIT'),
      }),
    }),
    HttpModule,
    ConfigModule,
    CacheModule.registerAsync({
      imports: [ConfigModule],
      inject: [ConfigService],
      useFactory: (configService: ConfigService) => ({
        store: redisStore,
        host: configService.get('REDIS_HOST'),
        port: configService.get('REDIS_PORT'),
        ttl: 60,
      }),
    }),
  ],
  providers: [
    LinkScanningService,
    AbuseDetectionService,
    IpFilteringService,
    ContentPolicyService,
    AuditLogService,
    {
      provide: APP_GUARD,
      useClass: ThrottlerGuard,
    },
  ],
  exports: [
    LinkScanningService,
    AbuseDetectionService,
    IpFilteringService,
    ContentPolicyService,
  ],
})
export class SecurityModule {}
```

8. API Module

Developer-friendly API:

- **RESTful API**: Complete CRUD operations
- **GraphQL API**: Flexible query capabilities
- **WebSocket API**: Real-time analytics updates
- **SDK libraries**: Client libraries for popular languages
- **Webhooks**: Event notifications for integrations
- **API documentation**: Swagger/OpenAPI specification
- **API versioning**: Support for multiple API versions

typescript

```
// main.ts (excerpt)
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  const configService = app.get(ConfigService);

  // API versioning
  app.enableVersioning({
    type: VersioningType.URI,
    defaultVersion: '1',
  });

  // Swagger documentation
  const options = new DocumentBuilder()
    .setTitle('ShortLink Pro API')
    .setDescription('The ShortLink Pro API description')
    .setVersion('1.0')
    .addTag('shortlinks')
    .addBearerAuth()
    .build();
  const document = SwaggerModule.createDocument(app, options);
  SwaggerModule.setup('api/docs', app, document);

  // GraphQL setup
  app.use(
    '/graphql',
    graphqlHTTP({
      schema: graphqlSchema,
      graphiql: configService.get('NODE_ENV') !== 'production',
    }),
  );

  // CORS configuration
  app.enableCors({
    origin: configService.get('ALLOWED_ORIGINS').split(','),
    methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
    credentials: true,
  });

  // Global validation pipe
  app.useGlobalPipes(
    new ValidationPipe({
      transform: true,
      whitelist: true,
    }),
  );
  // Global exception filter
  app.useGlobalFilters(new HttpExceptionFilter());

  await app.listen(configService.get('PORT') || 3000);
}
bootstrap();
```

Database Schema Design

```
sql
```

```sql
-- Users table
CREATE TABLE users (
    id UUID PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    subscription_tier VARCHAR(50) DEFAULT 'free',
    api_key UUID UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_login TIMESTAMP,
    is_email_verified BOOLEAN DEFAULT FALSE,
    two_factor_enabled BOOLEAN DEFAULT FALSE,
    two_factor_secret VARCHAR(255)
);

-- Teams table
CREATE TABLE teams (
    id UUID PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    owner_id UUID REFERENCES users(id) ON DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    subscription_id UUID
);

-- Team Members table
CREATE TABLE team_members (
    id UUID PRIMARY KEY,
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(50) NOT NULL, -- admin, editor, viewer
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(team_id, user_id)
);

-- Short Links table
CREATE TABLE short_links (
    id UUID PRIMARY KEY,
    original_url TEXT NOT NULL,
    short_code VARCHAR(20) UNIQUE NOT NULL,
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    team_id UUID REFERENCES teams(id) ON DELETE SET NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    password VARCHAR(255),
    title VARCHAR(255),
    description TEXT,
    tags JSONB,
    custom_domain VARCHAR(255),
    utm_source VARCHAR(100),
    utm_medium VARCHAR(100),
    utm_campaign VARCHAR(100),
    utm_term VARCHAR(100),
    utm_content VARCHAR(100),
    click_limit INTEGER,
    click_count INTEGER DEFAULT 0,
    metadata JSONB
);

-- Clicks table
CREATE TABLE clicks (
    id UUID PRIMARY KEY,
    short_link_id UUID REFERENCES short_links(id) ON DELETE CASCADE,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    user_agent TEXT,
    referrer TEXT,
    device_type VARCHAR(50),
    browser VARCHAR(50),
    operating_system VARCHAR(50),
    country VARCHAR(100),
    city VARCHAR(100),
    region VARCHAR(100),
    utm_data JSONB,
    custom_parameters JSONB
);

-- QR Codes table
CREATE TABLE qr_codes (
    id UUID PRIMARY KEY,
    short_link_id UUID REFERENCES short_links(id) ON DELETE CASCADE,
    image_url TEXT NOT NULL,
    background_color VARCHAR(7) DEFAULT '#FFFFFF',
    foreground_color VARCHAR(7) DEFAULT '#000000',
    logo_url TEXT,
    corner_radius INTEGER DEFAULT 0,
    error_correction VARCHAR(1) DEFAULT 'M',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    metadata JSONB
);

-- Subscriptions table
CREATE TABLE subscriptions (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    plan_name VARCHAR(50) NOT NULL,
    price_id VARCHAR(100),
    amount DECIMAL(10, 2) NOT NULL,
    currency VARCHAR(3) DEFAULT 'USD',
    interval VARCHAR(20) NOT NULL, -- monthly, yearly
    status VARCHAR(20) NOT NULL, -- active, canceled, past_due
    current_period_start TIMESTAMP NOT NULL,
    current_period_end TIMESTAMP NOT NULL,
    cancel_at_period_end BOOLEAN DEFAULT FALSE,
    payment_provider VARCHAR(20) NOT NULL, -- stripe, paypal
    payment_provider_id VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    metadata JSONB,
    CHECK (user_id IS NOT NULL OR team_id IS NOT NULL)
);

-- WebHooks table
CREATE TABLE webhooks (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    team_id UUID REFERENCES teams(id) ON DELETE SET NULL,
    url TEXT NOT NULL,
    secret VARCHAR(255),
    events JSONB NOT NULL, -- Array of event types
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_triggered_at TIMESTAMP,
    failure_count INTEGER DEFAULT 0
);

-- API Usage table
CREATE TABLE api_usage (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    endpoint VARCHAR(255) NOT NULL,
    method VARCHAR(10) NOT NULL,
    status_code INTEGER NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    response_time INTEGER, -- in milliseconds
    ip_address VARCHAR(45),
    user_agent TEXT
);

-- Custom Domains table
CREATE TABLE custom_domains (
    id UUID PRIMARY KEY,
    domain VARCHAR(255) UNIQUE NOT NULL,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    team_id UUID REFERENCES teams(id) ON DELETE SET NULL,
    verification_status VARCHAR(20) DEFAULT 'pending',
    dns_target TEXT,
    verification_code VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    verified_at TIMESTAMP,
    is_active BOOLEAN DEFAULT FALSE
);
```

Key Technical Features

1. Caching Strategy

typescript

```typescript
// cache.module.ts
@Module({
  imports: [
    CacheModule.registerAsync({
      imports: [ConfigModule],
      inject: [ConfigService],
      useFactory: (configService: ConfigService) => ({
        store: redisStore,
        host: configService.get('REDIS_HOST'),
        port: configService.get('REDIS_PORT'),
        ttl: 600, // Default TTL 10 minutes
        max: 10000, // Maximum number of items in cache
      }),
    }),
  ],
  providers: [CacheService],
  exports: [CacheModule, CacheService],
})
export class CustomCacheModule {}
```

2. Scheduled Tasks with NestJS

```typescript
// clean-expired-links.job.ts
@Injectable()
export class CleanExpiredLinksJob {
  constructor(
    @InjectRepository(ShortLink)
    private shortLinkRepository: Repository<ShortLink>,
    private readonly cacheManager: Cache,
    private readonly logger: Logger,
  ) {}

  @Cron('0 0 * * *') // Run at midnight daily
  async handleCleanExpiredLinks() {
    this.logger.log('Starting cleanup of expired links');

    const now = new Date();
    const expiredLinks = await this.shortLinkRepository.find({
      where: {
        expiresAt: LessThan(now),
        isActive: true,
      },
    });

    for (const link of expiredLinks) {
      // Update DB
```