متوجه شدم که نیاز به یک لیست کامل و جامع از تمام جداول و فیلدهای آن‌ها دارید. با توجه به محتوای ارائه‌شده در فایل‌ها و بحث‌های قبلی، در ادامه \*\*تمام جداول\*\* و \*\*فیلدهای هر جدول\*\* به‌صورت کامل و مرتب‌شده ارائه می‌شود:

---

### \*\*1. جدول کاربران (Users)\*\*

```csharp

public class AppUser : IdentityUser<Guid>

{

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime CreatedAt { get; set; }

public DateTime? LastLoginDate { get; set; }

public bool IsVerified { get; set; }

public bool IsActive { get; set; }

public int MaxAllowedDevices { get; set; }

public int ActiveDeviceCount { get; set; }

public string PhoneNumber { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

// Relationships

public ICollection<DeviceEntity> Devices { get; set; }

public ICollection<RefreshTokenEntity> RefreshTokens { get; set; }

public ICollection<SubscriptionEntity> Subscriptions { get; set; }

}

```

**جدول انواع دستگاه‌ها (DeviceTypes)**

این جدول برای مدیریت انواع مختلف دستگاه‌ها طراحی شده است.

public class DeviceTypeEntity

{

public Guid Id { get; set; } // شناسه یکتا

public string Name { get; set; } // نام نوع دستگاه (مانند GPS Tracker, Smart Watch)

public string Description { get; set; } // توضیحات

public bool IsActive { get; set; } // وضعیت فعال بودن

public bool IsDeleted { get; set; } // حذف منطقی

public DateTime CreatedAt { get; set; } // تاریخ ایجاد

public DateTime? UpdatedAt { get; set; } // زمان آخرین به‌روزرسانی

// Relationships

public ICollection<DeviceEntity> Devices { get; set; } // ارتباط با دستگاه‌ها

}

---

### \*\*2. جدول تغییرات کاربر (UserChanges)\*\*

```csharp

public class UserChangeEntity

{

public Guid Id { get; set; }

public Guid UserId { get; set; }

public string FieldChanged { get; set; }

public string OldValue { get; set; }

public string NewValue { get; set; }

public string ChangeReason { get; set; }

public DateTime ChangedAt { get; set; }

}

```

---

### \*\*3. جدول نقش‌ها (Roles)\*\*

```csharp public class PermissionEntity

{

public Guid Id { get; set; } // شناسه یکتا

public string Name { get; set; } // نام مجوز

public string Description { get; set; } // توضیحات مجوز

public bool IsDeleted { get; set; } // حذف منطقی

public DateTime? UpdatedAt { get; set; } // زمان آخرین به‌روزرسانی

// Relationships

public ICollection<RolePermissionEntity> RolePermissions { get; set; } // ارتباط با جدول واسط

}```

**3. جدول واسط نقش‌ها و مجوزها (RolePermissions)**

این جدول برای پیاده‌سازی رابطه Many-to-Many بین **RoleEntity** و **PermissionEntity** استفاده می‌شود.

public class RolePermissionEntity

{

public Guid RoleId { get; set; } // شناسه نقش

public RoleEntity Role { get; set; } // رابطه با نقش

public Guid PermissionId { get; set; } // شناسه مجوز

public PermissionEntity Permission { get; set; } // رابطه با مجوز

public bool IsDeleted { get; set; } // حذف منطقی

public DateTime? UpdatedAt { get; set; } // زمان آخرین به‌روزرسانی

}

---

### \*\*4. جدول مجوزها (Permissions)\*\*

```csharp

public class PermissionEntity

{

public Guid Id { get; set; } // شناسه یکتا

public string Name { get; set; } // نام مجوز

public string Description { get; set; } // توضیحات مجوز

public bool IsDeleted { get; set; } // حذف منطقی

public DateTime? UpdatedAt { get; set; } // زمان آخرین به‌روزرسانی

// Relationships

public ICollection<RolePermissionEntity> RolePermissions { get; set; } // ارتباط با جدول واسط

}

### \*\*5. جدول دستگاه‌ها (Devices)\*\*

```csharp

public class DeviceEntity

{

public Guid Id { get; set; }

public Guid UserId { get; set; }

public AppUser User { get; set; }

public string SerialNumber { get; set; }

public string Name { get; set; }

public DeviceStatus Status { get; set; }

public DateTime CreatedAt { get; set; }

public DateTime? LastStatusChangeAt { get; set; }

public DeviceStatus PreviousStatus { get; set; }

public DateTime? LastSeen { get; set; }

public int? BatteryLevel { get; set; }

public string FirmwareVersion { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

// Relationships

public ICollection<LocationEntity> Locations { get; set; }

public ICollection<CommandEntity> Commands { get; set; }

public Guid SubscriptionId { get; set; }

public SubscriptionEntity Subscription { get; set; }

}

```

---

### \*\*6. جدول موقعیت‌ها (Locations)\*\*

```csharp

public class LocationEntity

{

public Guid Id { get; set; }

public Guid DeviceId { get; set; }

public DeviceEntity Device { get; set; }

public double Latitude { get; set; }

public double Longitude { get; set; }

public double? Altitude { get; set; }

public double? Speed { get; set; }

public double? Direction { get; set; }

public DateTime Timestamp { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*7. جدول دستورات (Commands)\*\*

```csharp

public class CommandEntity

{

public Guid Id { get; set; }

public Guid DeviceId { get; set; }

public DeviceEntity Device { get; set; }

public CommandType Type { get; set; }

public CommandStatus Status { get; set; }

public DateTime CreatedAt { get; set; }

public DateTime? ExecutedAt { get; set; }

public string ErrorCode { get; set; }

public string ErrorMessage { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*8. جدول اشتراک‌ها (Subscriptions)\*\*

```csharp

public class SubscriptionEntity

{

public Guid Id { get; set; }

public Guid UserId { get; set; }

public AppUser User { get; set; }

public SubscriptionType Type { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public decimal Amount { get; set; }

public PaymentStatus Status { get; set; }

public string InvoiceId { get; set; }

public string PaymentMethod { get; set; }

public bool AutoRenewal { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

// Relationships

public ICollection<DeviceEntity> Devices { get; set; }

}

```

---

### \*\*9. جدول تاریخچه اشتراک‌ها (SubscriptionHistory)\*\*

```csharp

public class SubscriptionHistoryEntity

{

public Guid Id { get; set; }

public Guid SubscriptionId { get; set; }

public SubscriptionType PreviousType { get; set; }

public DateTime PreviousStartDate { get; set; }

public DateTime PreviousEndDate { get; set; }

public decimal PreviousAmount { get; set; }

public DateTime ChangedAt { get; set; }

public bool IsDeleted { get; set; }

}

```

---

### \*\*10. جدول اعلان‌ها (Notifications)\*\*

```csharp

public class NotificationEntity

{

public Guid Id { get; set; }

public Guid UserId { get; set; }

public AppUser User { get; set; }

public string Message { get; set; }

public NotificationType Type { get; set; }

public NotificationStatus Status { get; set; }

public bool IsRead { get; set; }

public DateTime CreatedAt { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*11. جدول پیام‌های پشتیبانی (SupportMessages)\*\*

```csharp

public class SupportMessageEntity

{

public Guid Id { get; set; }

public Guid UserId { get; set; }

public AppUser User { get; set; }

public string Subject { get; set; }

public string Message { get; set; }

public DateTime CreatedAt { get; set; }

public bool IsReadByAdmin { get; set; }

public bool IsDeleted { get; set; }

}

```

---

### \*\*12. جدول توکن‌های Refresh (RefreshTokens)\*\*

```csharp

public class RefreshTokenEntity

{

public int Id { get; set; }

public Guid UserId { get; set; }

public AppUser User { get; set; }

public string Token { get; set; }

public DateTime ExpiresAt { get; set; }

public bool IsRevoked { get; set; }

public DateTime CreatedAt { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*13. جدول لاگ‌ها (RequestLogs)\*\*

```csharp

public class RequestLog

{

public int Id { get; set; }

public string RequestMethod { get; set; }

public string RequestPath { get; set; }

public int StatusCode { get; set; }

public DateTime Timestamp { get; set; }

public string UserId { get; set; }

public string IpAddress { get; set; }

public string UserAgent { get; set; }

public string ErrorMessage { get; set; }

public bool IsDeleted { get; set; }

}

```

---

### \*\*14. جدول آلارم‌ها (Alarms)\*\*

```csharp

public class AlarmEntity

{

public Guid Id { get; set; }

public Guid DeviceId { get; set; }

public DeviceEntity Device { get; set; }

public string Message { get; set; }

public DateTime TriggeredAt { get; set; }

public bool Acknowledged { get; set; }

public Guid? AcknowledgedByUserId { get; set; }

public bool IsDeleted { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*15. جدول تنظیمات سیستم (SystemSettings)\*\*

```csharp

public class SystemSettingEntity

{

public Guid Id { get; set; }

public string Key { get; set; }

public string Value { get; set; }

public string Description { get; set; }

public DateTime CreatedAt { get; set; }

public DateTime? UpdatedAt { get; set; }

}

```

---

### \*\*16. جدول سلامت سیستم (SystemHealthStatus)\*\*

```csharp

public class SystemHealthStatus

{

public int Id { get; set; }

public double CpuUsage { get; set; }

public double MemoryUsage { get; set; }

public double DiskUsage { get; set; }

public string NetworkStatus { get; set; }

public DateTime LastChecked { get; set; }

public bool IsHealthy { get; set; }

}

public class DeviceEntity

{

public Guid Id { get; set; } // شناسه یکتا دستگاه

public Guid UserId { get; set; } // شناسه کاربر مالک دستگاه

public AppUser User { get; set; } // رابطه با کاربر

public Guid DeviceTypeId { get; set; } // شناسه نوع دستگاه

public DeviceTypeEntity DeviceType { get; set; } // رابطه با انواع دستگاه

public string SerialNumber { get; set; } // شماره سریال دستگاه

public string Name { get; set; } // نام دستگاه

public DeviceStatus Status { get; set; } // وضعیت دستگاه

public DateTime CreatedAt { get; set; } // تاریخ ثبت

public DateTime? LastSeen { get; set; } // آخرین زمان مشاهده دستگاه

public int? BatteryLevel { get; set; } // سطح شارژ دستگاه (اختیاری)

public string FirmwareVersion { get; set; } // نسخه فرم‌ور دستگاه

public bool IsDeleted { get; set; } // حذف منطقی

public DateTime? UpdatedAt { get; set; } // زمان آخرین به‌روزرسانی

// Relationships

public ICollection<LocationEntity> Locations { get; set; } // ارتباط با موقعیت‌ها

public ICollection<CommandEntity> Commands { get; set; } // ارتباط با دستورات

}

|  |
| --- |
| AuditEntity |

|  |
| --- |
| لاگ تغییرات سیستمی دقیق‌تر (برای دستگاه/اشتراک/نقش‌ها و...) |

### \*\*ساختار کلی پوشه‌بندی پروژه مدیریت دستگاه‌های GPS با ظرفیت پشتیبانی از 10 میلیون کاربر و بیشتر\*\*

در ادامه، ساختار کامل پوشه‌بندی پروژه مدیریت دستگاه‌های GPS به همراه توضیحات جامع درباره هر ماژول و کتابخانه‌های استفاده‌شده ارائه می‌شود. این ساختار شامل تمام ماژول‌ها، لایه‌ها، فایل‌ها، و وابستگی‌ها است.

---

## \*\*ساختار پوشه‌بندی کامل پروژه\*\*

```plaintext

/src

│

├── ApiGateway

│ ├── Controllers

│ │ └── GatewayController.cs

│ ├── Middleware

│ │ ├── AuthenticationMiddleware.cs

│ │ ├── RateLimitingMiddleware.cs

│ │ └── DeviceLimitMiddleware.cs

│ ├── Extensions

│ │ └── ServiceCollectionExtensions.cs

│ ├── Configurations

│ │ └── OcelotConfiguration.json

│ ├── appsettings.json

│ └── Program.cs

│

├── LocationTrackingService

│ ├── Controllers

│ │ └── LocationsController.cs

│ ├── Services

│ │ ├── LocationService.cs

│ │ └── ILocationService.cs

│ ├── Repositories

│ │ ├── LocationRepository.cs

│ │ └── ILocationRepository.cs

│ ├── Models

│ │ ├── LocationDto.cs

│ │ └── LocationEntity.cs

│ ├── Messaging

│ │ └── KafkaLocationPublisher.cs

│ ├── BackgroundServices

│ │ └── MqttBackgroundService.cs

│ ├── Persistence

│ │ ├── LocationDbContext.cs

│ │ └── EntityConfigurations

│ │ └── LocationConfiguration.cs

│ ├── Logging

│ │ └── RequestLoggingMiddleware.cs

│ ├── Caching

│ │ └── RedisCacheService.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── DeviceManagementService

│ ├── Controllers

│ │ └── DevicesController.cs

│ ├── Services

│ │ ├── DeviceService.cs

│ │ └── IDeviceService.cs

│ ├── Repositories

│ │ ├── DeviceRepository.cs

│ │ └── IDeviceRepository.cs

│ ├── Models

│ │ ├── DeviceDto.cs

│ │ └── DeviceEntity.cs

│ ├── Enums

│ │ └── DeviceStatus.cs

│ ├── ValueObjects

│ │ └── DeviceId.cs

│ ├── Persistence

│ │ ├── DeviceDbContext.cs

│ │ └── EntityConfigurations

│ │ └── DeviceConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── CommandService

│ ├── Controllers

│ │ └── CommandsController.cs

│ ├── Services

│ │ ├── CommandService.cs

│ │ └── ICommandService.cs

│ ├── Repositories

│ │ ├── CommandRepository.cs

│ │ └── ICommandRepository.cs

│ ├── Models

│ │ ├── CommandDto.cs

│ │ └── CommandEntity.cs

│ ├── Persistence

│ │ ├── CommandDbContext.cs

│ │ └── EntityConfigurations

│ │ └── CommandConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── NotificationService

│ ├── Controllers

│ │ └── NotificationsController.cs

│ ├── Services

│ │ ├── NotificationService.cs

│ │ └── INotificationService.cs

│ ├── Repositories

│ │ ├── NotificationRepository.cs

│ │ └── INotificationRepository.cs

│ ├── Models

│ │ ├── NotificationDto.cs

│ │ └── NotificationEntity.cs

│ ├── Persistence

│ │ ├── NotificationDbContext.cs

│ │ └── EntityConfigurations

│ │ └── NotificationConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── AuthService

│ ├── Controllers

│ │ └── AuthController.cs

│ ├── Services

│ │ ├── AuthService.cs

│ │ └── IAuthService.cs

│ ├── Models

│ │ ├── UserDto.cs

│ │ └── UserEntity.cs

│ ├── Enums

│ │ └── UserRole.cs

│ ├── Persistence

│ │ ├── AuthDbContext.cs

│ │ └── EntityConfigurations

│ │ └── UserConfiguration.cs

│ ├── Token

│ │ └── JwtTokenGenerator.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── SubscriptionService

│ ├── Controllers

│ │ └── SubscriptionsController.cs

│ ├── Services

│ │ ├── SubscriptionService.cs

│ │ └── ISubscriptionService.cs

│ ├── Repositories

│ │ ├── SubscriptionRepository.cs

│ │ └── ISubscriptionRepository.cs

│ ├── Models

│ │ ├── SubscriptionDto.cs

│ │ └── SubscriptionEntity.cs

│ ├── Persistence

│ │ ├── SubscriptionDbContext.cs

│ │ └── EntityConfigurations

│ │ └── SubscriptionConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── BillingService

│ ├── Controllers

│ │ └── BillingController.cs

│ ├── Services

│ │ ├── BillingService.cs

│ │ └── IBillingService.cs

│ ├── Repositories

│ │ ├── BillingRepository.cs

│ │ └── IBillingRepository.cs

│ ├── Models

│ │ ├── BillingDto.cs

│ │ └── BillingEntity.cs

│ ├── Persistence

│ │ ├── BillingDbContext.cs

│ │ └── EntityConfigurations

│ │ └── BillingConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── EventBus

│ ├── IEventPublisher.cs

│ ├── IEventSubscriber.cs

│ ├── KafkaEventBus.cs

│ ├── Outbox

│ │ ├── OutboxMessageEntity.cs

│ │ └── OutboxPublisherService.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── SystemMonitoringService

│ ├── Controllers

│ │ └── HealthCheckController.cs

│ ├── Services

│ │ ├── MonitoringService.cs

│ │ └── IMonitoringService.cs

│ ├── Models

│ │ └── HealthCheckResult.cs

│ ├── Persistence

│ │ ├── MonitoringDbContext.cs

│ │ └── EntityConfigurations

│ │ └── MonitoringConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── JobSchedulerService

│ ├── QuartzJobService.cs

│ ├── IScheduledTask.cs

│ ├── Jobs

│ │ ├── RenewSubscriptionJob.cs

│ │ └── SendNotificationJob.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── Contracts

│ ├── Device

│ │ └── DeviceDto.cs

│ ├── Auth

│ │ └── AuthResponseDto.cs

│ ├── Command

│ │ └── CommandDto.cs

│ └── Notification

│ └── NotificationDto.cs

│

├── SharedKernel

│ ├── Exceptions

│ │ └── CustomException.cs

│ ├── Interfaces

│ │ └── IRepository.cs

│ ├── Extensions

│ │ └── ServiceCollectionExtensions.cs

│ ├── Constants

│ │ └── AppConstants.cs

│ ├── Audit

│ │ └── AuditLogService.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── Infrastructure

│ ├── Persistence

│ │ └── BaseDbContext.cs

│ ├── Messaging

│ │ └── MqttConfiguration.cs

│ ├── Logging

│ │ └── SerilogLogger.cs

│ ├── Caching

│ │ └── RedisConfiguration.cs

│ ├── Security

│ │ └── EncryptionHelper.cs

│ ├── HealthChecks

│ │ └── HealthCheckExtensions.cs

│ ├── appsettings.json

│ └── Program.cs

│

└── Tests

├── UnitTests

│ ├── LocationServiceTests.cs

│ ├── DeviceServiceTests.cs

│ ├── CommandServiceTests.cs

│ ├── NotificationServiceTests.cs

│ ├── AuthServiceTests.cs

│ └── BillingServiceTests.cs

└── IntegrationTests

├── LocationTrackingIntegrationTests.cs

├── DeviceManagementIntegrationTests.cs

├── CommandIntegrationTests.cs

└── NotificationIntegrationTests.cs

```

---

## \*\*توضیحات جامع برای هر ماژول\*\*

### \*\*1. ApiGateway\*\*

- \*\*هدف\*\*: نقطه ورودی واحد برای تمام API‌ها.

- \*\*کاربرد\*\*: توزیع درخواست‌ها به میکروسرویس‌های مختلف (Routing).

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Ocelot\*\* یا \*\*YARP\*\* برای مدیریت API Gateway.

- \*\*Polly\*\* برای مدیریت Circuit Breaker و Retry Policies.

- \*\*فایل‌ها\*\*:

- \*\*Controllers\*\*: مدیریت درخواست‌ها.

- \*\*Middleware\*\*: مدیریت خطاها، CORS، و محدودیت‌های نرخ.

- \*\*Extensions\*\*: تنظیمات Dependency Injection.

- \*\*Program.cs\*\*: راه‌اندازی برنامه.

- \*\*appsettings.json\*\*: تنظیمات پیکربندی مانند آدرس سرویس‌ها و کلیدهای امنیتی.

---

### \*\*2. LocationTrackingService\*\*

- \*\*هدف\*\*: مدیریت داده‌های موقعیت GPS.

- \*\*کاربرد\*\*: دریافت، پردازش، و ذخیره داده‌های موقعیت.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*InfluxDB\*\* یا \*\*TimescaleDB\*\* برای ذخیره داده‌های زمان‌محور.

- \*\*Redis\*\* برای کش کردن داده‌ها.

- \*\*Kafka\*\* برای ارسال داده‌ها به صف‌های رویداد.

- \*\*فایل‌ها\*\*:

- \*\*Controllers\*\*: دریافت درخواست‌ها از کلاینت.

- \*\*Services\*\*: منطق کسب‌وکار (مانند محاسبه فاصله).

- \*\*Repositories\*\*: دسترسی به دیتابیس.

- \*\*Persistence\*\*: مدیریت دیتابیس.

---

### \*\*3. DeviceManagementService\*\*

- \*\*هدف\*\*: مدیریت ثبت، ویرایش، حذف، و اختصاص دستگاه‌ها.

- \*\*کاربرد\*\*: اعتبارسنجی شماره سریال دستگاه‌ها و اتصال آن‌ها به حساب کاربری.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*AutoMapper\*\* برای نقشه‌برداری بین مدل‌ها.

- \*\*FluentValidation\*\* برای اعتبارسنجی داده‌ها.

- \*\*فایل‌ها\*\*:

- \*\*Models\*\*: تعریف مدل‌های داده‌ای.

- \*\*Enums\*\*: وضعیت دستگاه‌ها (مانند روشن/خاموش).

- \*\*ValueObjects\*\*: شناسه دستگاه به‌صورت Value Object.

---

### \*\*4. CommandService\*\*

- \*\*هدف\*\*: ارسال دستورات به دستگاه‌ها.

- \*\*کاربرد\*\*: مدیریت صف‌ها و رویدادها با استفاده از Kafka.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Kafka\*\* برای مدیریت صف‌ها.

- \*\*Twilio\*\* یا \*\*Nexmo\*\* برای ارسال دستورات از طریق SMS.

- \*\*فایل‌ها\*\*:

- \*\*Controllers\*\*: دریافت درخواست‌های کاربر.

- \*\*Services\*\*: ارسال دستورات به دستگاه‌ها.

- \*\*Models\*\*: تعریف دستورات و وضعیت آن‌ها.

---

### \*\*5. NotificationService\*\*

- \*\*هدف\*\*: مدیریت ارسال اعلان‌ها و هشدارها.

- \*\*کاربرد\*\*: ارسال Push Notification، ایمیل، یا پیامک.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Firebase Cloud Messaging (FCM)\*\* برای ارسال Push Notification.

- \*\*SendGrid\*\* یا \*\*MailKit\*\* برای ارسال ایمیل.

- \*\*فایل‌ها\*\*:

- \*\*Controllers\*\*: مدیریت درخواست‌های مربوط به اعلان‌ها.

- \*\*Services\*\*: منطق ارسال اعلان‌ها.

- \*\*Repositories\*\*: ذخیره اعلان‌ها در دیتابیس.

---

### \*\*6. AuthService\*\*

- \*\*هدف\*\*: مدیریت احراز هویت و دسترسی‌ها.

- \*\*کاربرد\*\*: استفاده از JWT برای احراز هویت.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*IdentityServer4\*\* برای مدیریت احراز هویت.

- \*\*BCrypt.Net\*\* برای رمزنگاری رمز عبور.

- \*\*فایل‌ها\*\*:

- \*\*Models\*\*: تعریف کاربران و نقش‌ها.

- \*\*Token\*\*: تولید و اعتبارسنجی توکن‌ها.

---

### \*\*7. SubscriptionService\*\*

- \*\*هدف\*\*: مدیریت اشتراک‌ها و پرداخت‌ها.

- \*\*کاربرد\*\*: مدیریت اشتراک‌ها و درگاه‌های پرداخت.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Stripe\*\* یا \*\*PayPal\*\* برای مدیریت پرداخت‌ها.

- \*\*Dapper\*\* برای عملیات CRUD سریع.

- \*\*فایل‌ها\*\*:

- \*\*Models\*\*: تعریف اشتراک‌ها و پرداخت‌ها.

- \*\*Repositories\*\*: ذخیره اطلاعات اشتراک‌ها.

---

### \*\*8. EventBus\*\*

- \*\*هدف\*\*: مدیریت ارتباط بین ماژول‌ها با استفاده از رویدادها.

- \*\*کاربرد\*\*: استفاده از Kafka برای انتشار و دریافت رویدادها.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Confluent.Kafka\*\* برای پیاده‌سازی EventBus.

- \*\*فایل‌ها\*\*:

- \*\*IEventPublisher\*\*: انتشار رویدادها.

- \*\*IEventSubscriber\*\*: دریافت رویدادها.

- \*\*KafkaEventBus\*\*: پیاده‌سازی EventBus.

---

### \*\*9. JobSchedulerService\*\*

- \*\*هدف\*\*: مدیریت وظایف زمان‌بندی‌شده (Scheduled Tasks).

- \*\*کاربرد\*\*: اجرای وظایف مانند تمدید خودکار اشتراک‌ها و ارسال نوتیفیکیشن.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Quartz.NET\*\* برای زمان‌بندی وظایف.

- \*\*فایل‌ها\*\*:

- \*\*QuartzJobService.cs\*\*: پیاده‌سازی Scheduler برای مدیریت وظایف.

- \*\*IScheduledTask.cs\*\*: Interface برای تعریف وظایف زمان‌بندی‌شده.

---

### \*\*10. SharedKernel\*\*

- \*\*هدف\*\*: شامل کدهای مشترک بین ماژول‌ها.

- \*\*کاربرد\*\*: ارائه Interface‌ها، DTO‌ها، و Extension Methods.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*AutoMapper\*\* برای نقشه‌برداری بین مدل‌ها.

- \*\*FluentValidation\*\* برای اعتبارسنجی.

---

### \*\*11. Infrastructure\*\*

- \*\*هدف\*\*: شامل پیاده‌سازی مخازن، اتصال به دیتابیس، وب‌سوکت‌ها، و ابزارهای جانبی.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Entity Framework Core\*\* برای مدیریت دیتابیس.

- \*\*Serilog\*\* برای لاگ‌گیری.

- \*\*Redis\*\* برای کش کردن داده‌ها.

---

### \*\*12. Tests\*\*

- \*\*هدف\*\*: شامل تست‌های واحد و ادغام.

- \*\*کاربرد\*\*: استفاده از xUnit و Moq برای تست‌ها.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*xUnit\*\* برای نوشتن تست‌های واحد.

- \*\*Moq\*\* برای شبیه‌سازی وابستگی‌ها.

---

### \*\*نتیجه‌گیری\*\*

این ساختار پوشه‌بندی و پیکربندی، تمام نیازمندی‌های پروژه را پوشش می‌دهد و قابلیت مقیاس‌پذیری، انعطاف‌پذیری، و توسعه‌پذیری بالایی دارد.

### \*\*ساختار کامل پروژه مدیریت دستگاه‌های GPS با توجه به نیازمندی‌های اضافی\*\*

در ادامه، ساختار کامل پوشه‌بندی پروژه مدیریت دستگاه‌های GPS با توجه به تمام نیازمندی‌های ارائه‌شده (مانند \*\*Event Sourcing\*\*، \*\*MQTT\*\*، \*\*Kafka\*\*، \*\*Stream Processing\*\*، و سایر موارد) طراحی شده است. این ساختار شامل تمام ماژول‌ها، فایل‌ها، وابستگی‌ها، و توضیحات جامع برای هر بخش است.

---

## \*\*ساختار کلی پوشه‌بندی پروژه\*\*

```plaintext

/src

│

├── ApiGateway

│ ├── Controllers

│ │ └── GatewayController.cs

│ ├── Middleware

│ │ ├── AuthenticationMiddleware.cs

│ │ ├── RateLimitingMiddleware.cs

│ │ └── DeviceLimitMiddleware.cs

│ ├── Extensions

│ │ └── ServiceCollectionExtensions.cs

│ ├── Configurations

│ │ ├── OcelotConfiguration.json

│ │ └── RateLimiterConfiguration.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── LocationTrackingService

│ ├── Controllers

│ │ └── LocationsController.cs

│ ├── Services

│ │ ├── LocationService.cs

│ │ └── ILocationService.cs

│ ├── Repositories

│ │ ├── LocationRepository.cs

│ │ └── ILocationRepository.cs

│ ├── Models

│ │ ├── LocationDto.cs

│ │ └── LocationEntity.cs

│ ├── Messaging

│ │ └── KafkaLocationPublisher.cs

│ ├── BackgroundServices

│ │ └── MqttBackgroundService.cs

│ ├── Persistence

│ │ ├── LocationDbContext.cs

│ │ └── EntityConfigurations

│ │ └── LocationConfiguration.cs

│ ├── Logging

│ │ └── RequestLoggingMiddleware.cs

│ ├── Caching

│ │ └── RedisCacheService.cs

│ ├── RealTimeCommunication

│ │ └── SignalRHub.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── EventStoreService

│ ├── Controllers

│ │ └── EventsController.cs

│ ├── Services

│ │ ├── EventStoreService.cs

│ │ └── IEventStoreService.cs

│ ├── Repositories

│ │ ├── EventRepository.cs

│ │ └── IEventRepository.cs

│ ├── Models

│ │ ├── EventDto.cs

│ │ └── EventEntity.cs

│ ├── Persistence

│ │ ├── EventDbContext.cs

│ │ └── EntityConfigurations

│ │ └── EventConfiguration.cs

│ ├── Streaming

│ │ └── StreamProcessor.cs

│ ├── appsettings.json

│ └── Program.cs

│

├── Infrastructure

│ ├── Messaging

│ │ ├── MqttConfiguration.cs

│ │ └── SignalRConfiguration.cs

│ ├── Kafka

│ │ ├── KafkaProducer.cs

│ │ └── KafkaConsumer.cs

│ ├── Cache

│ │ └── RedisConfiguration.cs

│ ├── Monitoring

│ │ └── PrometheusMetrics.cs

│ ├── appsettings.json

│ └── Program.cs

│

└── Tests

├── UnitTests

│ ├── LocationServiceTests.cs

│ ├── EventStoreServiceTests.cs

└── IntegrationTests

├── MqttIntegrationTests.cs

├── KafkaIntegrationTests.cs

└── SignalRIntegrationTests.cs

```

---

## \*\*توضیحات جامع برای هر ماژول\*\*

### \*\*1. EventStoreService\*\*

- \*\*هدف\*\*: مدیریت رویدادها (Event Sourcing).

- \*\*کاربرد\*\*:

- ذخیره‌سازی تمام رویدادها به صورت غیرقابل تغییر.

- امکان Replay و Versioning برای بازسازی حالت سیستم.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*PostgreSQL\*\*: برای ذخیره‌سازی رویدادها.

- \*\*Apache Flink\*\* یا \*\*Apache Spark Streaming\*\*: برای پردازش داده‌ها.

- \*\*فایل‌ها\*\*:

- \*\*Controllers\*\*: مدیریت درخواست‌های مربوط به رویدادها.

- \*\*Services\*\*: منطق ذخیره و بازیابی رویدادها.

- \*\*Repositories\*\*: دسترسی به دیتابیس رویدادها.

- \*\*Streaming\*\*: پردازش داده‌ها با استفاده از Flink یا Spark.

---

### \*\*2. MQTT Broker (EMQX یا HiveMQ)\*\*

- \*\*هدف\*\*: نقطه ورودی داده‌ها از دستگاه‌ها.

- \*\*کاربرد\*\*:

- ارتباط مستقیم با دستگاه‌ها.

- استفاده از QoS 1 یا 2 برای تضمین دریافت پیام.

- Persistent Session برای مدیریت قطع و وصل دستگاه‌ها.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*MQTTnet\*\*: برای پیاده‌سازی MQTT در سمت سرور.

- \*\*EMQX\*\* یا \*\*HiveMQ\*\*: به عنوان MQTT Broker.

---

### \*\*3. Kafka – Central Message Bus\*\*

- \*\*هدف\*\*: مدیریت صف‌های پیام مقاوم و مقیاس‌پذیر.

- \*\*کاربرد\*\*:

- فوروارد داده‌ها از MQTT به Kafka.

- تضمین عدم از دست رفتن داده‌ها.

- Consumer Groups برای پردازش موازی.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Confluent.Kafka\*\*: برای پیاده‌سازی Kafka.

- \*\*Kafka Connect\*\*: برای اتصال به EMQX.

---

### \*\*4. Stream Processor (Apache Flink / Apache Spark Streaming)\*\*

- \*\*هدف\*\*: پردازش بلادرنگ داده‌ها.

- \*\*کاربرد\*\*:

- شناسایی دستگاه‌های Offline یا ورود به مناطق خاص (GeoFence).

- تولید هشدارها و ارسال به SignalR یا ذخیره‌سازی.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Apache Flink\*\* یا \*\*Apache Spark Streaming\*\*.

---

### \*\*5. Cache (Redis)\*\*

- \*\*هدف\*\*: کاهش بار روی دیتابیس و افزایش سرعت دسترسی.

- \*\*کاربرد\*\*:

- ذخیره موقت داده‌های پرتکرار (مانند آخرین موقعیت دستگاه‌ها).

- استفاده فقط برای Read-heavy APIs.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*StackExchange.Redis\*\*: برای پیاده‌سازی Redis.

---

### \*\*6. SignalR / WebSocket Layer\*\*

- \*\*هدف\*\*: ارسال داده‌های زنده به پنل مدیریت یا اپ موبایل.

- \*\*کاربرد\*\*:

- متصل به Kafka Consumer یا Stream Processor.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*SignalR\*\*: برای ارتباط بلادرنگ.

---

### \*\*7. Prometheus + Grafana\*\*

- \*\*هدف\*\*: مانیتورینگ عملکرد سیستم.

- \*\*کاربرد\*\*:

- جمع‌آوری متریک‌ها.

- نمایش داشبوردهای عملکرد.

- \*\*کتابخانه‌ها و ابزارها\*\*:

- \*\*Prometheus-net\*\*: برای جمع‌آوری متریک‌ها.

- \*\*Grafana\*\*: برای نمایش داشبوردها.

---

### \*\*8. نتیجه‌گیری\*\*

این ساختار پوشه‌بندی و پیکربندی، تمام نیازمندی‌های پروژه را پوشش می‌دهد و قابلیت مقیاس‌پذیری، انعطاف‌پذیری، و توسعه‌پذیری بالایی دارد. اگر نیاز به توضیحات بیشتری در مورد هر بخش دارید، خوشحال می‌شوم کمک کنم! 😊