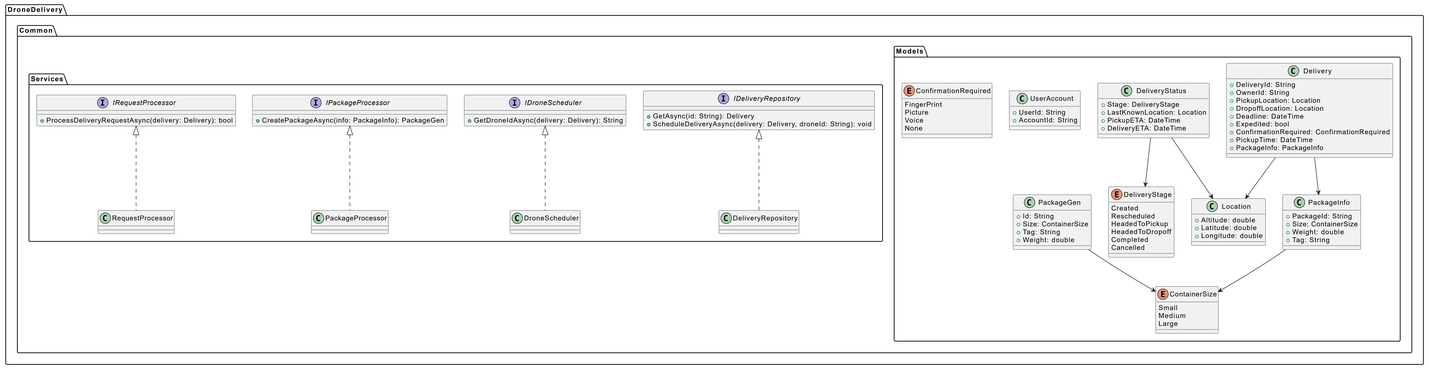
SASVA Project Scan Report

Git URL: https://github.com/vishalgoyal16444/microservices-architecture | Branch: master

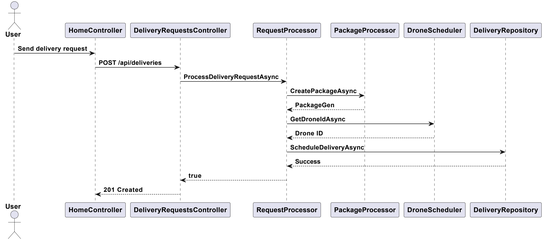
1) Class Diagram:

The class diagram represents the structure of the DroneDelivery system, focusing on the models and services. It includes classes like Delivery, DeliveryStatus, Location, PackageInfo, and PackageGen, which model the data entities. Enumerations such as ConfirmationRequired, DeliveryStage, and ContainerSize define specific sets of constants used in the system. The services package contains interfaces like IDeliveryRepository, IDroneScheduler, IPackageProcessor, and IRequestProcessor, which define the operations for managing deliveries, scheduling drones, processing packages, and handling requests. Implementations of these interfaces are also shown, indicating the system's modular design.



2) Sequence Diagram:

The sequence diagram illustrates the process of handling a delivery request in the DroneDelivery system. The user initiates a delivery request through the HomeController, which sends a POST request to the DeliveryRequestsController. The controller then calls the RequestProcessor to handle the request. The RequestProcessor coordinates with the PackageProcessor to create a package, the DroneScheduler to assign a drone, and the DeliveryRepository to schedule the delivery. Upon successful processing, a 201 Created response is returned to the HomeController.

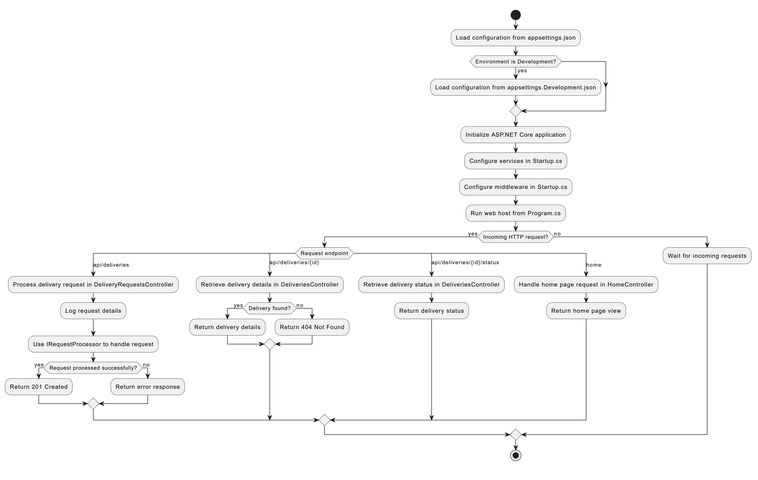


3) Logical Diagram:

The logical diagram represents the deployment and structure of a drone delivery system using Azure and ASP.NET Core. The Azure deployment includes a web application, function app, Application Insights for monitoring, and a storage account. The ASP.NET Core application consists of controllers for handling delivery requests, services for processing packages and scheduling drones, and models for representing delivery and package data. The diagram illustrates the flow from deployment to application components, highlighting the integration of various services and models within the system.

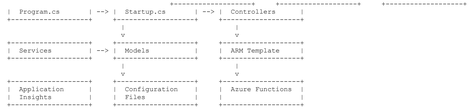
4) Activity Diagram:

The activity diagram illustrates the process flow of the DroneDelivery application. It starts by loading configuration settings from appsettings.json and appsettings.Development.json if in a development environment. The ASP.NET Core application is initialized, and services and middleware are configured in Startup.cs. The web host is run from Program.cs. When an HTTP request is received, it is routed to the appropriate controller based on the endpoint. The DeliveryRequestsController handles delivery requests, logging details and using IRequestProcessor to process them. The DeliveriesController retrieves delivery details or status, returning the appropriate response. The HomeController handles requests to the home page, returning the home page view. The application waits for incoming requests when idle.


Load configuration from appsettings.json
Environment is Development?
yes
Load configuration from appsettings.Development.json
Initialize ASP.NET Core application
Configure services in Startup.cs
Configure middleware in Startup.cs
Run web host from Program.cs
Incoming HTTP request?
yes
no
Request endpoint
api/deliveries
api/deliveries/{id}
api/deliveries/{id}/status
home
Process delivery request in DeliveryRequestsController
Retrieve delivery details in DeliveriesController
Retrieve delivery status in DeliveriesController
Handle home page request in HomeController
Wait for incoming requests
Log request details
Delivery found?
yes
no
Return delivery details
Return 404 Not Found
Return delivery status
Return home page view
Use IRequestProcessor to handle request
Request processed successfully?
yes
no
Return 201 Created
Return error response

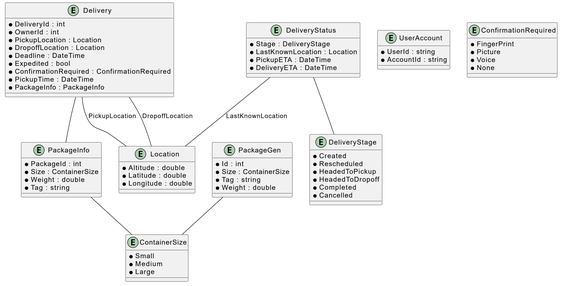
5) Component Diagram:

The component diagram represents the structure of the DroneDelivery application. The Program.cs file serves as the entry point, initializing the application and linking to Startup.cs, which configures services and middleware. Controllers handle HTTP requests, interacting with Services for business logic and Models for data structures. The ARM Template automates Azure resource deployment. Application Insights is configured for monitoring, while Configuration Files manage settings. Azure Functions handle serverless operations.



6) ER (Entity-Relationship) Diagram:

The ER diagram represents the key entities and their relationships in the drone delivery system. The 'Delivery' entity includes details about a delivery order, such as locations, deadline, and package information. 'PackageInfo' and 'PackageGen' entities describe package details, including size and weight. 'Location' represents geographical coordinates. 'DeliveryStatus' tracks the current status of a delivery, including its stage and location. Enumerations like 'ConfirmationRequired', 'ContainerSize', and 'DeliveryStage' define possible values for confirmation methods, container sizes, and delivery stages, respectively. The diagram illustrates how these entities are interconnected, providing a structured view of the system's data model.

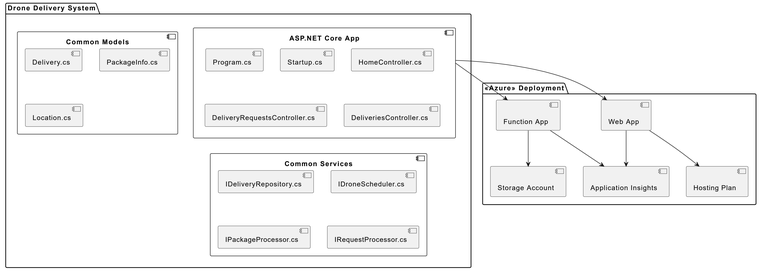


7) Use Case Diagram:

The use case diagram illustrates the interactions between actors and the system's functionalities. Developers can deploy web applications and monitor them using Application Insights. Users can handle delivery requests, retrieve delivery details, and send delivery requests. The system processes delivery requests by creating packages and scheduling drones.

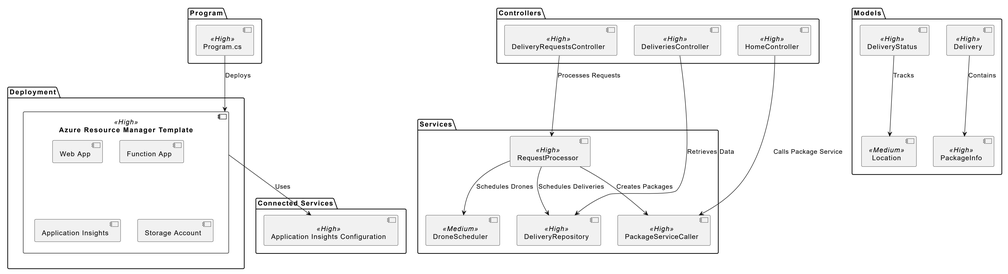
8) Deployment Diagram:

The deployment diagram illustrates the architecture of the Drone Delivery System deployed on Azure. It includes a Web App and a Function App, both connected to Application Insights for monitoring. The Web App is hosted on a Hosting Plan, while the Function App uses a Storage Account. The ASP.NET Core application, which includes various controllers and services, interacts with both the Web App and the Function App, demonstrating the integration of application components with Azure resources.



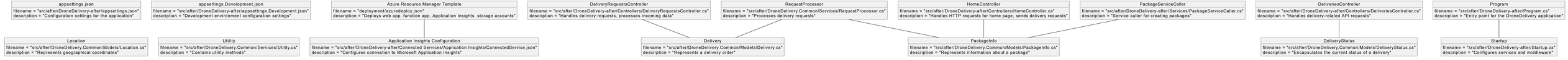
9) Composite Structure Diagram:

The composite structure diagram illustrates the key components of the DroneDelivery system and their interactions. The Azure Resource Manager Template (ARM) is responsible for deploying the web app, function app, Application Insights, and storage account. The Application Insights Configuration (AIC) is used for monitoring. The controllers (DeliveryRequestsController, DeliveriesController, HomeController) handle API requests and interact with services like the RequestProcessor and PackageServiceCaller. The Program component initializes the application. The RequestProcessor coordinates between the DeliveryRepository, DroneScheduler, and PackageServiceCaller to process delivery requests. The Delivery model contains package information, and the DeliveryStatus tracks the delivery's location.



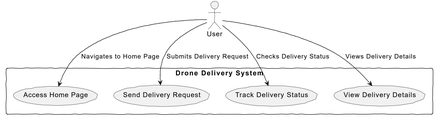
10) Object Diagram:

This object diagram represents the key components of the DroneDelivery application. It includes the Azure Resource Manager template for deploying resources, configuration files for Application Insights, and various controllers and services that handle delivery requests and manage application settings. The diagram shows the relationships between these components, such as the DeliveryRequestsController interacting with the Delivery model, and the RequestProcessor coordinating with both Delivery and PackageInfo models. The Program and Startup files are linked, indicating the setup and configuration of the application.



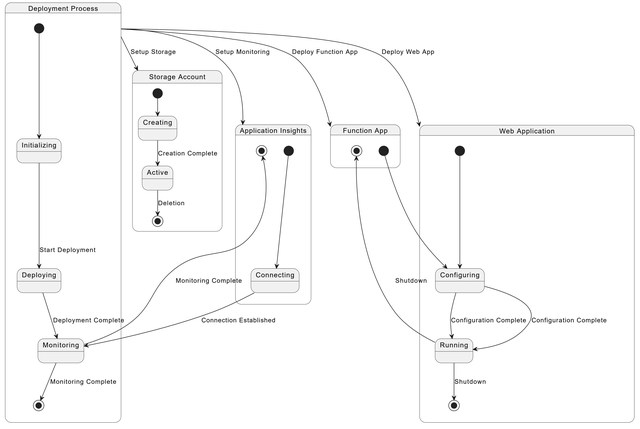
11) User Journey Map:

The user journey map illustrates the interactions between a user and the Drone Delivery System. The user can access the home page, send delivery requests, track the status of deliveries, and view detailed delivery information. Each interaction is represented as a use case within the system.



12) State Diagram:

The state diagram represents the deployment process of a web application and function app using an Azure Resource Manager (ARM) template. The process begins with initialization, followed by deployment, and ends with monitoring. The web application and function app both go through configuration and running states. Application Insights is connected for monitoring, and a storage account is created and activated. The diagram shows the transitions between these states and the dependencies between different components during the deployment process.

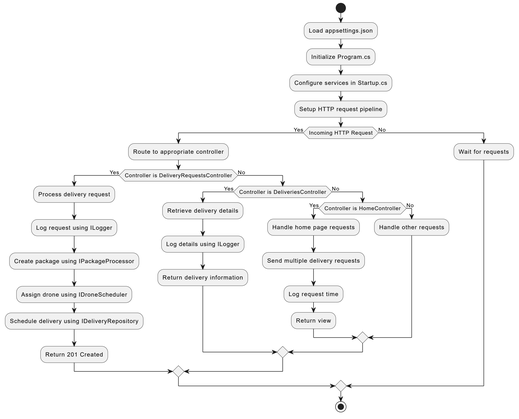


13) Architecture Diagram:

The architecture diagram represents a drone delivery system deployed on Azure. It includes a Web Application and a Function App, both connected to a Storage Account for data persistence. The Web Application interacts with Application Insights for monitoring and logging. The Function App, which includes the Package Service, handles HTTP requests for package updates. The Delivery System Components manage delivery operations, including scheduling and processing, facilitated by various services and interfaces defined in the project.

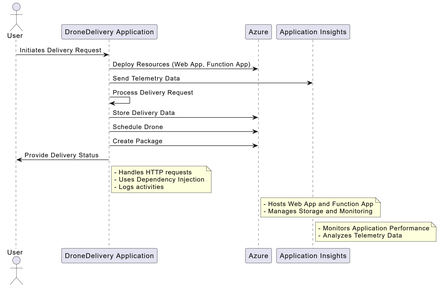
14) Workflow Diagram:

The workflow diagram illustrates the process flow of the DroneDelivery application. It starts with loading configuration settings from appsettings.json, initializing the application via Program.cs, and configuring services and middleware in Startup.cs. When an HTTP request is received, it is routed to the appropriate controller. The DeliveryRequestsController processes delivery requests, logs them, creates packages, assigns drones, and schedules deliveries. The DeliveriesController retrieves and logs delivery details, while the HomeController handles home page requests and sends multiple delivery requests. The diagram captures the decision-making process and interactions between components in handling requests.



15) Communication Diagram:

The communication diagram illustrates the interactions between the user, the DroneDelivery application, Azure services, and Application Insights. The user initiates a delivery request, which the application processes by deploying necessary resources on Azure, such as a web app and function app. The application sends telemetry data to Application Insights for monitoring. It processes the delivery request, stores delivery data, schedules a drone, and creates a package on Azure. Finally, the application provides the delivery status back to the user. The diagram highlights the application's use of dependency injection, logging, and its role in handling HTTP requests.


User
DroneDelivery Application
Azure
Application Insights
Initiates Delivery Request
Deploy Resources (Web App, Function App)
Send Telemetry Data
Process Delivery Request
Store Delivery Data
Schedule Drone
Create Package
Provide Delivery Status
- Handles HTTP requests
- Uses Dependency Injection
- Logs activities
- Hosts Web App and Function App
- Manages Storage and Monitoring
- Monitors Application Performance
- Analyzes Telemetry Data
User
DroneDelivery Application
Azure
Application Insights