Securaa Playbook Service - Low Level Design Document

Document Information

- **Service Name**: Securaa Playbook Service
- **Version**: 1.0
- **Date**: September 11, 2025
- **Author**: Development Team
- **Related Documents**: High Level Design

Table of Contents

1. Overview

The Low Level Design document provides detailed implementation specifications for the Securaa Playbook Service, including class structures, method signatures, algorithm implementations, and detailed interaction patterns.

1.1 Scope

This document covers:

- Detailed class and method specifications
- Database schema with indexes and constraints
- Complete API specifications with validation rules
- Concurrency patterns and thread safety mechanisms
- Performance optimization techniques
- Error handling and recovery strategies

2. Detailed Component Design

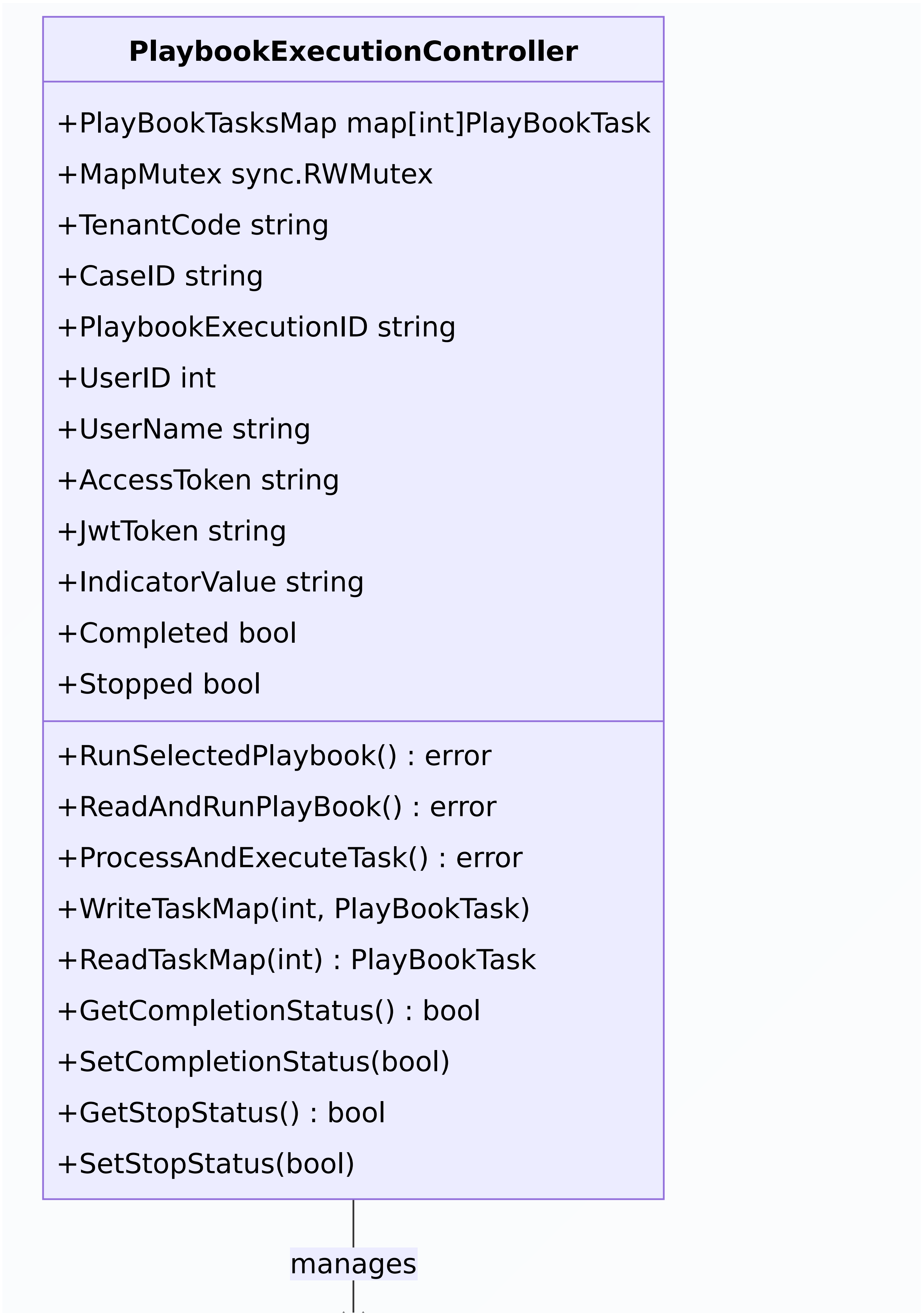
2.1 Core Package Structure

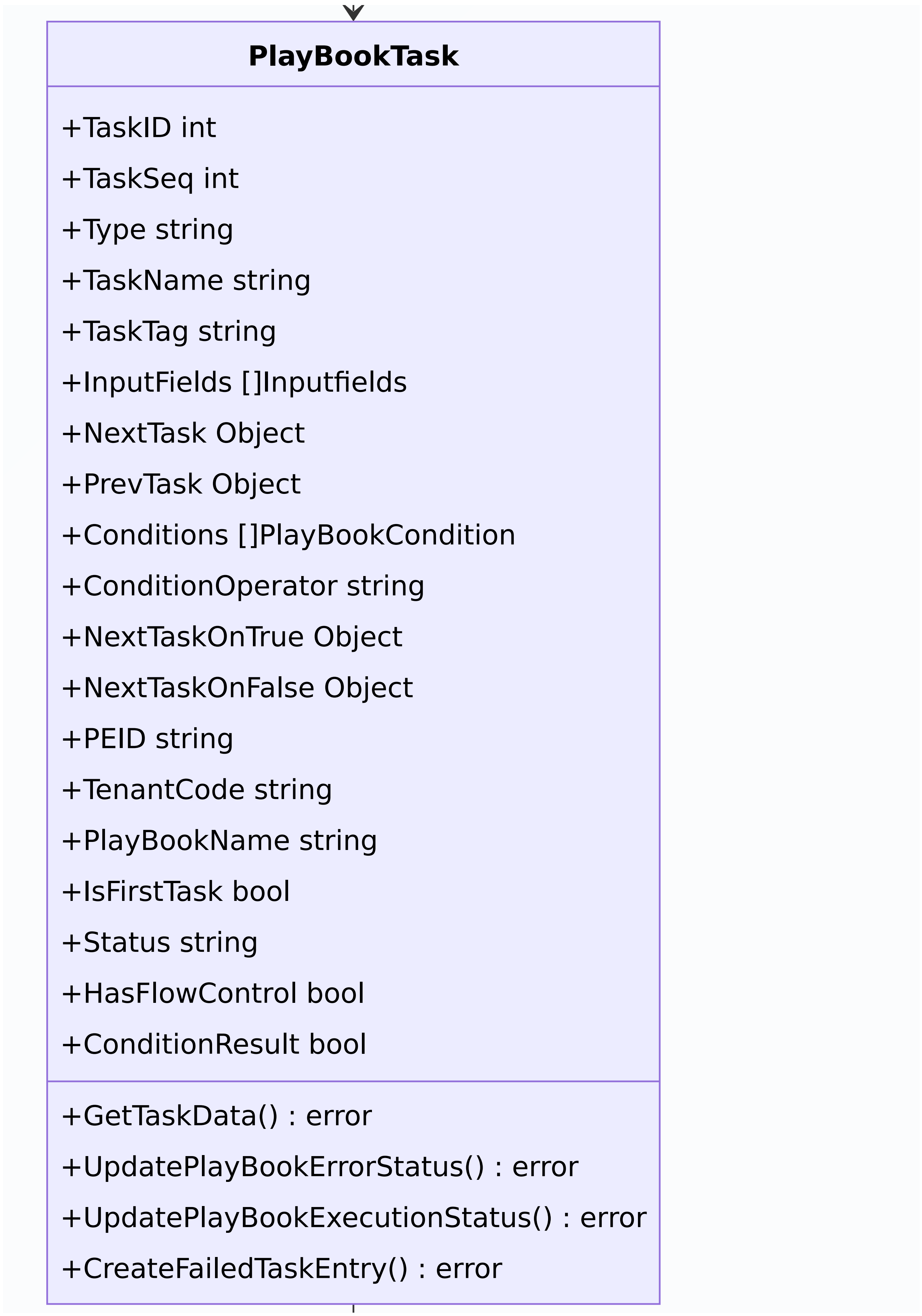
```
securaa_services/securaa_playbook/
├── main.go                     // Application entry point
├── app.go                      // Application initialization
├── controllers/                // HTTP request handlers
│   ├── playbookcontroller.go
│   ├── listController.go
│   ├── caseController.go
│   ├── supportcontroller.go
│   └── processController.go
├── executionControllers/       // Execution orchestration
│   ├── playbookExecutionController.go
│   ├── runTaskController.go
│   ├── conditionController.go
│   └── subPlaybookController.go
```

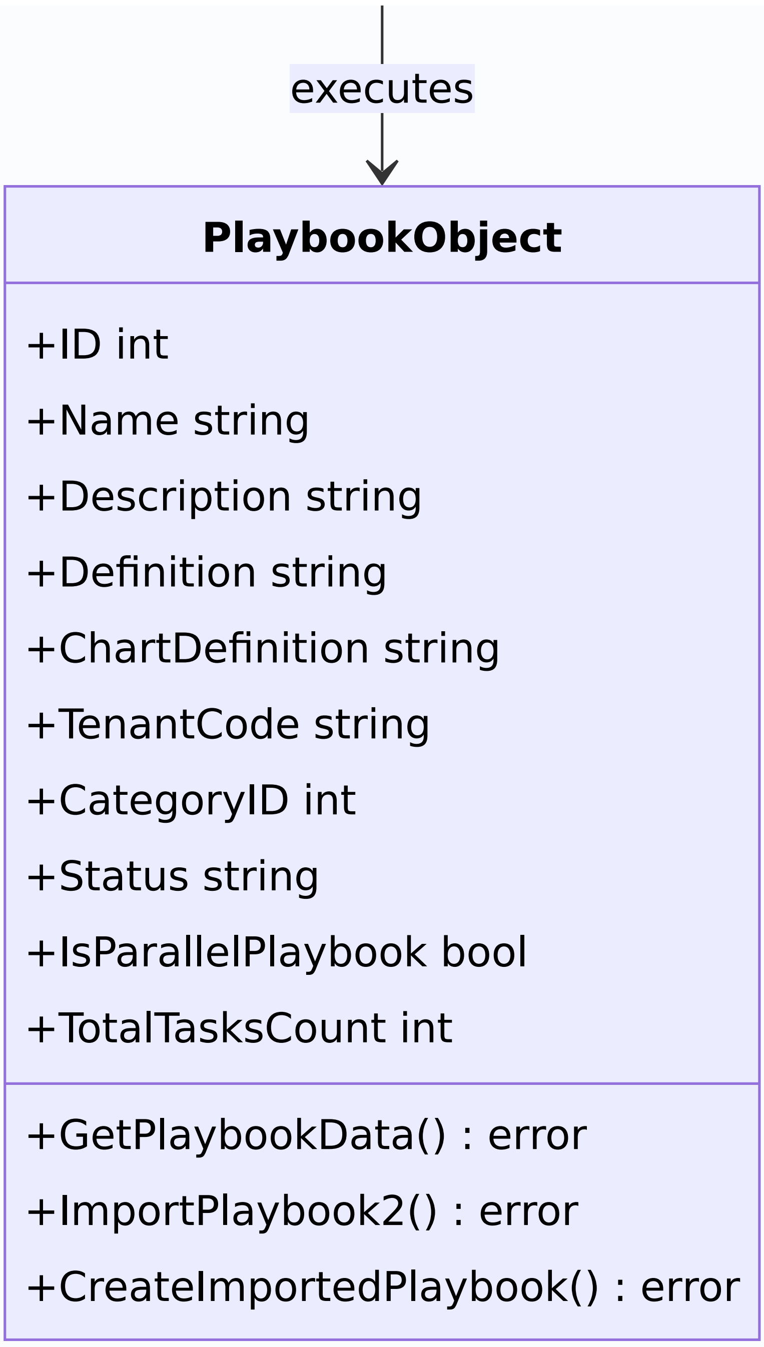
```
├── models/                     // Data models
│   ├── playbook.go
│   ├── case.go
│   ├── task.go
│   └── Response.go
├── executionModels/            // Execution-specific models
│   ├── playbook.go
│   ├── Tasks.go
│   └── incidents.go
├── services/                   // Business logic
│   ├── genericTaskService.go
│   ├── processService.go
│   └── filterNTransformService.go
├── utils/                      // Utility functions
│   ├── filterConditionUtils.go
│   ├── matchConditionUtils.go
│   └── executionUtils.go
├── handlers/                   // Error and response handlers
│   ├── errorHandler.go
│   └── taskResponse.go
├── constants/                  // Application constants
│   └── constants.go
└── cacheControllers/          // Cache management
    └── cacheController.go
```

3. Class Diagrams

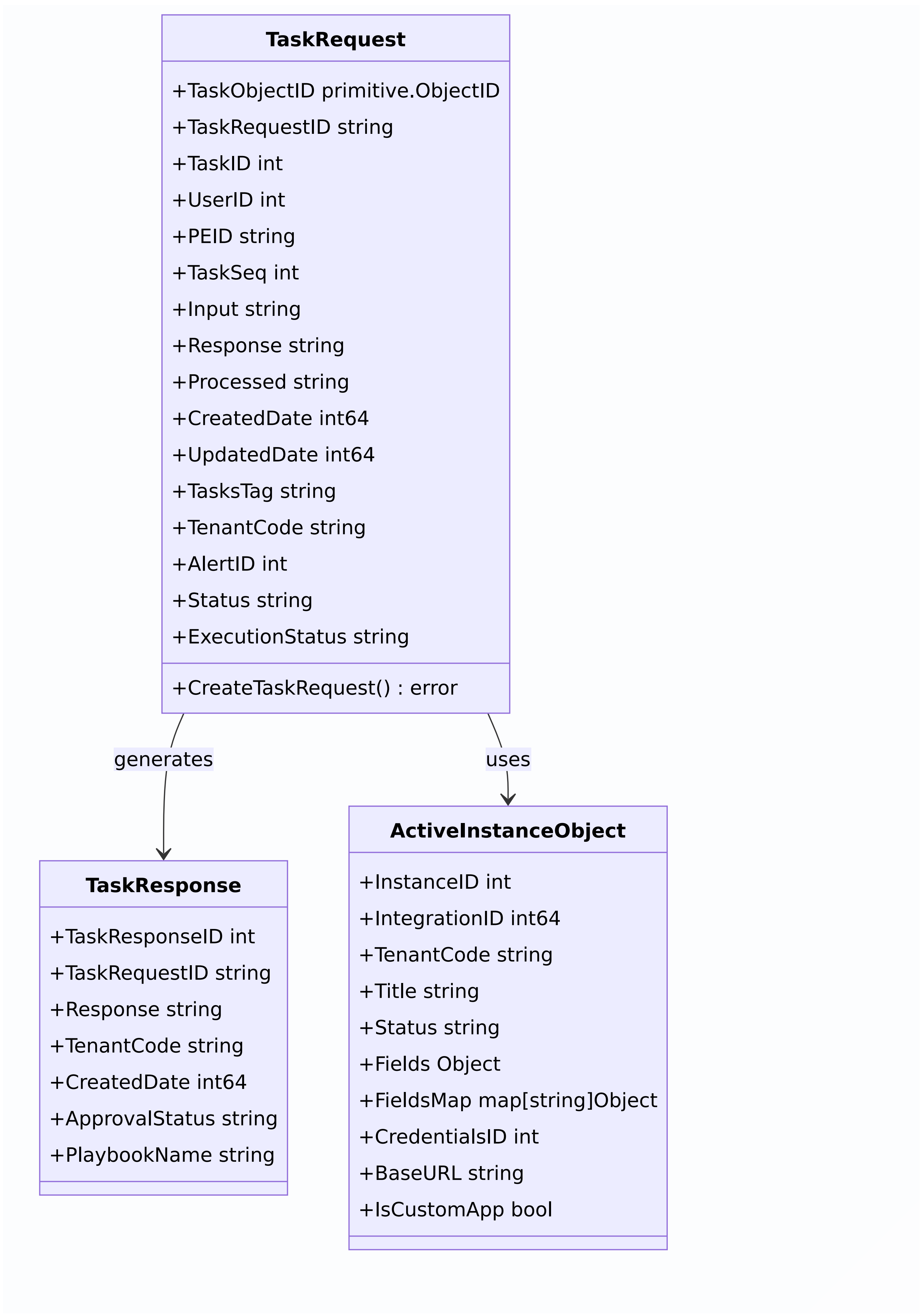
3.1 Playbook Execution Model


PlaybookExecutionController
+PlayBookTasksMap map[int]PlayBookTask
+MapMutex sync.RWMutex
+TenantCode string
+CaseID string
+PlaybookExecutionID string
+UserID int
+UserName string
+AccessToken string
+JwtToken string
+IndicatorValue string
+Completed bool
+Stopped bool
+RunSelectedPlaybook() : error
+ReadAndRunPlayBook() : error
+ProcessAndExecuteTask() : error
+WriteTaskMap(int, PlayBookTask)
+ReadTaskMap(int) : PlayBookTask
+GetCompletionStatus() : bool
+SetCompletionStatus(bool)
+GetStopStatus() : bool
+SetStopStatus(bool)
manages


PlayBookTask
+TaskID int
+TaskSeq int
+Type string
+TaskName string
+TaskTag string
+InputFields []Inputfields
+NextTask Object
+PrevTask Object
+Conditions []PlayBookCondition
+ConditionOperator string
+NextTaskOnTrue Object
+NextTaskOnFalse Object
+PEID string
+TenantCode string
+PlayBookName string
+IsFirstTask bool
+Status string
+HasFlowControl bool
+ConditionResult bool
+GetTaskData() : error
+UpdatePlayBookErrorStatus() : error
+UpdatePlayBookExecutionStatus() : error
+CreateFailedTaskEntry() : error

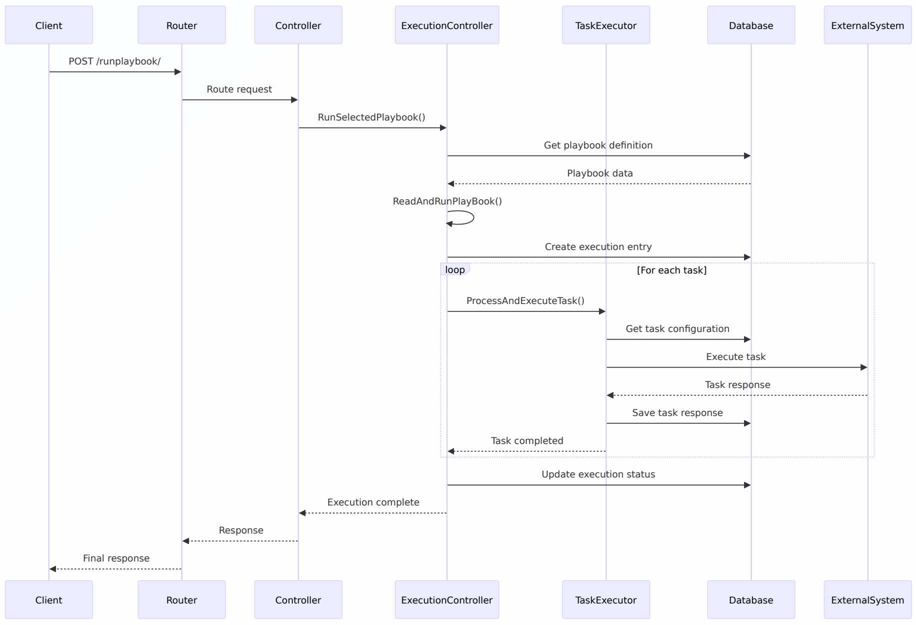

executes
PlaybookObject
+ID int
+Name string
+Description string
+Definition string
+ChartDefinition string
+TenantCode string
+CategoryID int
+Status string
+IsParallelPlaybook bool
+TotalTasksCount int
+GetPlaybookData() : error
+ImportPlaybook2() : error
+CreateImportedPlaybook() : error

3.2 Task Execution Model


TaskRequest
+TaskObjectID primitive.ObjectID
+TaskRequestID string
+TaskID int
+UserID int
+PEID string
+TaskSeq int
+Input string
+Response string
+Processed string
+CreatedDate int64
+UpdatedDate int64
+TasksTag string
+TenantCode string
+AlertID int
+Status string
+ExecutionStatus string
+CreateTaskRequest() : error
generates
uses
TaskResponse
+TaskResponseID int
+TaskRequestID string
+Response string
+TenantCode string
+CreatedDate int64
+ApprovalStatus string
+PlaybookName string
ActiveInstanceObject
+InstanceID int
+IntegrationID int64
+TenantCode string
+Title string
+Status string
+Fields Object
+FieldsMap map[string]Object
+CredentialsID int
+BaseURL string
+IsCustomApp bool

4. Sequence Diagrams

4.1 Playbook Execution Flow



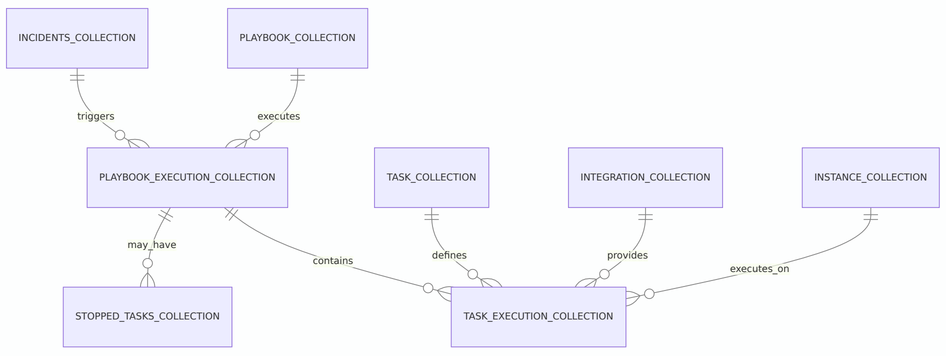
5. Database Schema

5.1 MongoDB Collections Schema

5.1.1 Playbook Collection

```
{
  "_id": ObjectId,
  "id": 1001,
  "name": "Malware Response Playbook",
  "description": "Automated malware response workflow",
  "version": "1.0.0",
  "definition": "...",
  "chart_definition": "...",
  "tenant_code": "tenant123",
  "category_id": 5,
  "status": "active",
  "created_date": 1694443200000,
  "updated_date": 1694443200000,
  "user_id": 1001,
  "group_id": 100,
  "type": "case",
  "filename": "1001_tenant123.json",
  "commit_id": "abc123",
  "list_names": ["suspicious_ips", "malware_domains"],
  "all_nodes_connected": "yes",
  "custom_utils_added": false,
  "custom_utils_names": [],
  "vertical_pb": false,
  "is_parallel_playbook": true,
  "total_tasks_count": 15,
  "total_utils_count": 3,
  "shard_bucket": 1
}
```

5.2 Data Relationships



6. API Specifications

6.1 Playbook Management APIs

6.1.1 Create Playbook

```
POST /createplaybook/
Content-Type: application/json
Authorization: Bearer {jwt_token}

Request Body:
{
  "name": "Malware Response Playbook",
  "description": "Automated response to malware incidents",
  "definition": "...",
  "chart_definition": "...",
  "category_id": 5,
  "type": "case",
  "tenant_code": "tenant123",
  "user_id": 1001,
  "version": "1.0.0",
  "is_parallel_playbook": true,
  "total_tasks_count": 15,
  "total_utils_count": 3
}
```

```
Response:
{
  "success": true,
  "data": {
    "playbook_id": 1001,
    "filename": "1001_tenant123.json",
    "commit_id": "abc123"
  },
  "error": "",
  "displayMessage": "Playbook created successfully",
  "time": 1694443200000
}
```

6.1.2 Run Playbook

```
POST /runplaybook/
Content-Type: application/json
Authorization: Bearer {jwt_token}

Request Body:
{
  "tenantcode": "tenant123",
  "playbook_name": "Malware Response Playbook",
  "case_id": "50001",
  "is_bot": "false",
  "uid": "1001",
  "username": "security_analyst",
  "type": "case",
  "indicator": "192.168.1.100",
  "playbook_execution_id": "",
  "resume_playbook": "false"
}

Response:
{
  "success": true,
  "data": {
    "playbook_execution_id": "pb_exec_123456",
    "status": "inprogress",
    "total_tasks": 15,
    "estimated_duration": 300000
  },
  "error": "",
  "displayMessage": "Playbook execution started",
  "time": 1694443200000
}
```

7. Algorithm Specifications

This section includes detailed algorithm implementations for core functionalities like parallel task execution, condition evaluation, and cache management.

8. Configuration Management

Configuration structure and environment-based configuration management for the service.

9. Error Handling Implementation

Comprehensive error handling strategies including error types, hierarchy, and retry mechanisms.

10. Concurrency & Thread Safety

Thread-safe implementations for concurrent operations and channel-based communication patterns.

11. Performance Optimizations

Performance optimization techniques including connection pooling, batch operations, and memory management.

12. Testing Strategy

Comprehensive testing approach including unit testing, integration testing, and performance testing strategies.