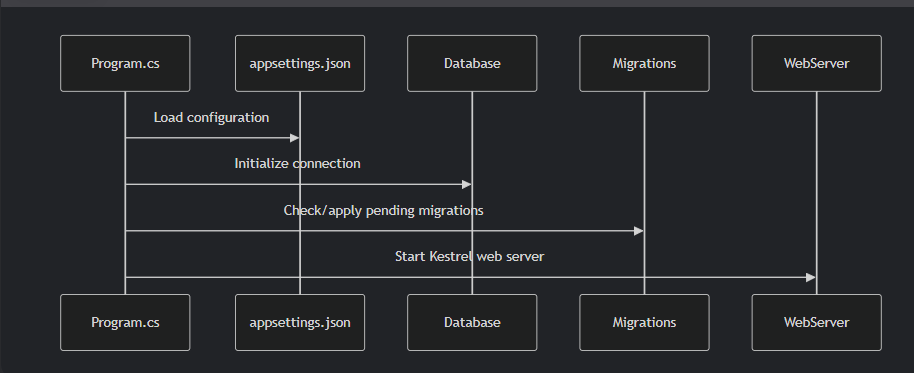
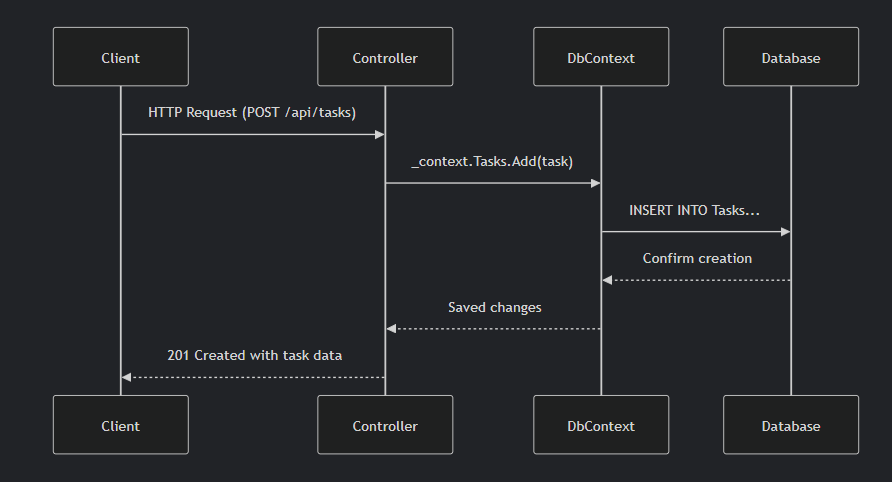
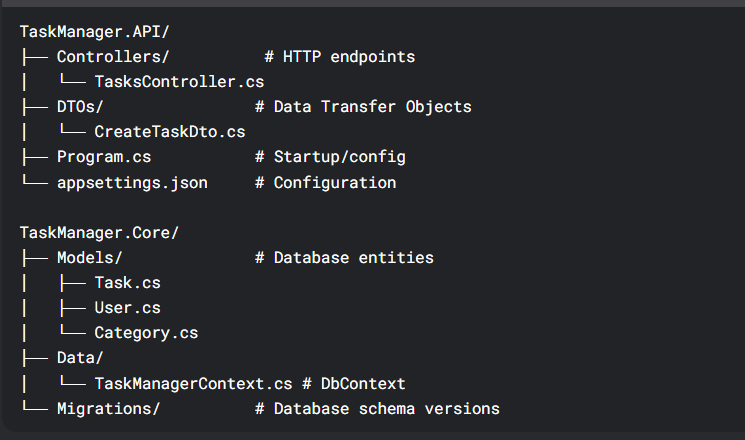
**1. Application Startup Flow**  
  


**2. HTTP Request Flow (CRUD Operations)**



**3. Component Architecture**  


**4. Detailed Operation Flow**

**A. Startup Process**

1. Program.cs reads appsettings.json
2. Configures DbContext with:
   * Connection retry policy (5 attempts)
   * 30-second command timeout
   * Detailed logging in development
3. Applies database migrations automatically
4. Starts web server

**B. Create Task Flow**

1. Client POSTs to /api/tasks with JSON body
2. Controller:

var task = new AppTask { /\* Map from DTO \*/ };

\_context.Tasks.Add(task);

await \_context.SaveChangesAsync();

1. Database:

INSERT INTO Tasks (Title, Description, ...)

VALUES ('Task 1', 'Description', ...)

**C. Read Operations Flow**

1. For GET /api/tasks:

return await \_context.Tasks

.Include(t => t.User)

.Include(t => t.Category)

.ToListAsync();

Generates SQL:

sql

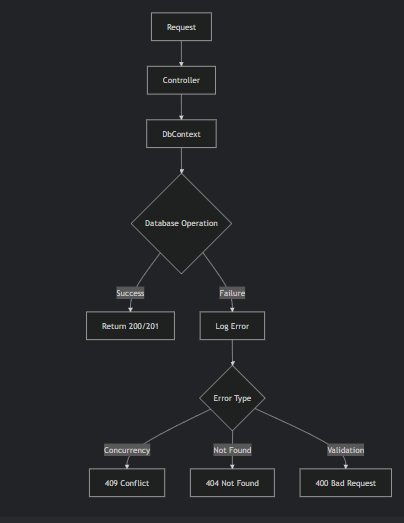
SELECT t.\*, u.\*, c.\*

FROM Tasks t

LEFT JOIN Users u ON t.UserId = u.Id

LEFT JOIN Categories c ON t.CategoryId = c.Id

**5. Error Handling Flow**



**6. Key Technical Stack**

| **Component** | **Technology** |
| --- | --- |
| Database Access | Entity Framework Core 9 |
| Database | SQL Server (LocalDB) |
| API Documentation | Swagger/OpenAPI |
| Configuration | appsettings.json |
| Logging | ILogger + Console Output |

**7. Important Dependencies**

<!-- TaskManager.API.csproj -->

<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="9.0.0"/> //for db

<PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0"/> //for swagger

<!-- TaskManager.Core.csproj -->

<PackageReference Include="Microsoft.EntityFrameworkCore" Version="9.0.0"/> //for ORM framework

**8. Database Schema Flow**

1. Models → DbContext → Migrations → Physical Database
2. Migration Process:

dotnet ef migrations add InitialCreate

dotnet ef database update

1. Generates:
   * Tasks table
   * Users table
   * Categories table
   * Proper relationships (foreign keys)

**9. Testing Flow Recommendations**

1. **Unit Tests**: Mock DbContext → Test business logic
2. **Integration Tests**:

var client = factory.CreateClient();

var response = await client.PostAsJsonAsync("/api/tasks", newTaskDto);

1. **Postman Tests**:
   * Verify status codes
   * Check response shapes
   * Test error conditions

**For Postman Testing**complete CRUD flow with sample data for your Task Manager API:

**1. CREATE (POST) - Add New Task**

Endpoint: POST /api/tasks  
Request:

{

"title": "Implement JWT Authentication",

"description": "Add secure token-based auth",

"dueDate": "2024-03-25T14:00:00",

"status": 0, // Pending

"userId": 1,

"categoryId": 2

}

Database Action:

INSERT INTO Tasks (Title, Description, DueDate, Status, UserId, CategoryId)

VALUES ('Implement JWT Authentication', 'Add secure token-based auth',

'2024-03-25 14:00:00', 0, 1, 2)

Response (201 Created):

{

"id": 1,

"title": "Implement JWT Authentication",

"description": "Add secure token-based auth",

"dueDate": "2024-03-25T14:00:00",

"status": 0,

"userId": 1,

"categoryId": 2

}

2. READ (GET) - Retrieve Tasks

A. Get All Tasks

Endpoint: GET /api/tasks  
Database Action:

SELECT t.\*, u.\*, c.\*

FROM Tasks t

LEFT JOIN Users u ON t.UserId = u.Id

LEFT JOIN Categories c ON t.CategoryId = c.Id

Response (200 OK):

[

{

"id": 1,

"title": "Implement JWT Authentication",

"description": "Add secure token-based auth",

"dueDate": "2024-03-25T14:00:00",

"status": 0,

"userId": 1,

"categoryId": 2,

"user": {

"id": 1,

"username": "dev\_user"

},

"category": {

"id": 2,

"name": "Security"

}

}

]

B. Get Single Task

Endpoint: GET /api/tasks/1  
Response (200 OK):

{

"id": 1,

"title": "Implement JWT Authentication",

"description": "Add secure token-based auth",

"dueDate": "2024-03-25T14:00:00",

"status": 0,

"userId": 1,

"categoryId": 2

}

3. UPDATE (PUT) - Modify Task

Endpoint: PUT /api/tasks/1  
Request:

{

"id": 1,

"title": "[Updated] JWT Implementation",

"description": "Include refresh tokens",

"dueDate": "2024-03-28T14:00:00",

"status": 1, // InProgress

"userId": 1,

"categoryId": 2

}

Database Action:

UPDATE Tasks SET

Title = '[Updated] JWT Implementation',

Description = 'Include refresh tokens',

DueDate = '2024-03-28 14:00:00',

Status = 1

WHERE Id = 1

Response: 204 No Content

4. DELETE - Remove Task

Endpoint: DELETE /api/tasks/1  
Database Action:

DELETE FROM Tasks WHERE Id = 1

Response: 204 No Content

Status Codes Reference

| Value | Status |
| --- | --- |
| 0 | Pending |
| 1 | InProgress |
| 2 | Completed |

Sample Test Data

Users Table

INSERT INTO Users (Id, Username, Email)

(1, 'dev\_user', 'dev@example.com'),

(2, 'qa\_user', 'qa@example.com')

Categories Table

INSERT INTO Categories (Id, Name)

(1, 'Development'),

(2, 'Security'),

(3, 'Documentation')

**Postman Test Collection**

1. Create Task Test:

pm.test("Task created successfully", () => {

pm.response.to.have.status(201);

pm.ex(pm.response.json().title).to.eql("Implement JWT Authentication");

});

1. Update Task Test:

pm.test("Status changed to InProgress", () => {

pm.ex(pm.response.json().status).to.eql(1);

});