Emite Backend Developer Coding Test: Call Center Management API

Task: Building a Call Center Management API

Your task is to create a RESTful API for managing call center operations using .NET Core or later. This project should demonstrate your skills in API design, database interactions, authentication, and testing.

Requirements:

1. Create a Web API using ASP.NET Core (.NET 6 or later)
2. Implement CRUD operations for all the data models
3. Use Entity Framework Core for database operations
4. Implement JWT authentication
5. Write unit tests for your services
6. Implement basic error handling and logging
7. Use dependency injection
8. Use https://www.usebruno.com/ as rest api client tool.

Data Models:

Implement the following models:

1. **Agent** model:
   - Id (int)
   - Name (string)
   - Email (string)
   - PhoneExtension (string)

- Status (enum: Available, Busy, Offline)

2. **Call** model:

- Id (int)
- CustomerId (string)
- AgentId (int, nullable)
- StartTime (DateTime)
- EndTime (DateTime, nullable)
- Status (enum: Queued, InProgress, Completed, Dropped)
- Notes (string)

3. **Customer** model:

- Id (string)
- Name (string)
- Email (string)
- PhoneNumber (string)
- LastContactDate (DateTime, nullable)

4. **Ticket** model

- Id (int)
- CustomerId (string)
- AgentId (int, nullable)
- Status (enum: Open, InProgress, Resolved, Closed)
- Priority (enum: Low, Medium, High, Urgent)
- CreatedAt (DateTime)
- UpdatedAt (DateTime)
- Description (string)
- Resolution (string, nullable)

API Endpoints:

Implement the following endpoints:

1. Agents:

- Retrieve all agents
- Retrieve a specific agent
- Add a new agent
- Update an existing agent
- Delete an agent
- Update agent status

2. Calls:

- Retrieve all calls
- Retrieve a specific call
- Create a new call
- Update an existing call
- Delete a call
- Assign a call to an agent

3. Customers:

- Retrieve all customers
- Retrieve a specific customer
- Add a new customer
- Update an existing customer
- Delete a customer

4. Tickets:

- Retrieve all tickets
- Retrieve a specific ticket
- Create a new ticket
- Update an existing ticket
- Delete a ticket
- Assign a ticket to an agent

Authentication:

- Implement JWT authentication

- Secure all endpoints except getting all calls and getting specific call

Testing:

- Write unit tests for your service layer

- Implement integration tests for your API endpoints

Bonus (Optional):

1. Implement pagination for calls and tickets.

2. Add a search functionality to filter calls by status, date range, or agent

3. Implement a simple in-memory cache for GET requests to improve performance

4. Create an endpoint to get basic statistics (e.g., average call duration, calls per agent)

5. Implement a simple rate limiting mechanism

6. Add real-time notifications for new calls using SignalR

7. Implement a basic call routing algorithm to assign calls to available agents

8. If you know how to use Elasticsearch that would be a bonus.

Submission:

- Provide a GitHub repository link with your solution
- Include a README.md file with:
    - Instructions on how to set up and run your project
    - Any assumptions or design decisions you made
    - A brief explanation of how you approached the problem
- Submit the bruno files used for developing the Api.

Evaluation Criteria:

- Code quality and organization
- Proper use of .NET Core and C# features
- Correct implementation of RESTful API principles

- Effective use of Entity Framework Core
- Proper implementation of authentication
- Quality and coverage of unit and integration tests
- Error handling and logging
- Bonus points for completing optional tasks or adding extra features

Good luck! We look forward to reviewing your solution.