You are tasked with developing a C# ASP.NET REST API hosted within a Windows console application using .NET. Your objective is to create and manage essential models and implement specific behaviors as described below.

**Models:**

1. **Product Model:**
   * Properties:
     + ProductId (Guid): Unique identifier for the product.
     + Name (string): Name of the product.
2. **Lot Model:**
   * Properties:
     + LotId (Guid): Unique identifier for the lot.
     + ProductId (Guid): Identifier for the associated product.
     + Description (string): Description of the lot.
     + CreatedAt (DateTime): Timestamp of lot creation.
3. **ProductLotQuantity Model (Read-only from REST):**
   * Properties:
     + LotId (Guid): Identifier for the associated lot.
     + ProductId (Guid): Identifier for the associated product.
     + Quantity (decimal): Quantity of the product in the lot.
4. **ProductLotTransaction Model:**
   * Properties:
     + LotId (Guid): Identifier of the associated lot. This is optional in INSERT requests and can be null.
     + ProductId (Guid): Identifier for the associated product.
     + Quantity (decimal): Quantity related to the transaction.
     + Type (TransactionType enum: LIFO, FIFO): Type of transaction.

**Controllers:**

1. **Product Controller:**
   * Responsible for handling CRUD operations for the Product model.
2. **Lot Controller:**
   * Responsible for handling CRUD operations for the Lot model.
3. **ProductLotQuantity Controller:**
   * Responsible for handling READ (C**R**UD) operations for the ProductLotQuantity model.
4. **ProductLotTransaction Controller:**
   * Responsible for handling CREATE and READ (**CR**UD) operations for the ProductLotTransaction model.

**Requirements:**

1. The Product Controller should implement CRUD operations for managing products (Create, Read, Update, Delete).
2. The Lot Controller should implement CRUD operations for managing lots (Create, Read, Update, Delete).
3. The ProductLotQuantity Controller should support READ operations to retrieve product lot quantities.
4. The ProductLotTransaction Controller should support CREATE and READ operations only.
5. The ProductLotTransaction Controller should UPDATE/CREATE the corresponding ProductLotQuantity when a new transaction is created.
6. In ProductLotTransaction, LotId is optional in CREATE requests. When NULL (or not passed), the LotID should be chosen according to the Type:
   * If LIFO (Last-In-First-Out), select the Lot that belongs to the corresponding Product with the EARLIEST CreatedAt time.
   * If FIFO (First-In-First-Out), select the Lot that belongs to the corresponding Product with the LATEST CreatedAt time.
7. Implement the strategy pattern using Object-Oriented Programming (OOP) to determine FIFO and LIFO behavior in the ProductLotTransaction Controller.
8. ProductLotTransaction Controller should return an HTTP status code indicating that other HTTP methods (e.g., PUT, DELETE) are not supported for ProductLotTransaction.
9. Each attempt for a operation (i.e. HTTP request), not described in (1-4) should return an HTTP status code indicating that the operation is not supported.
10. Write test cases to ensure compliance with the requirements. Pay particular attention to the tests related to requirement (6).
11. Create API reference (documentation) for all supported endpoints, using a popular tool such as Swagger.
12. Host the project source in a public GitHub repository and ensure all the requirements are reflected in separate commits.