**ĐỀ THI THỬ SỐ 1**

**MÔN: PHÁT TRIỂN ỨNG DỤNG WEB NÂNG CAO**

**LỚP CTK44 – THỜI GIAN: 90 PHÚT**

**CÂU 1: (2 điểm)**

1. Học phần Phát triển ứng dụng Web nâng cao đã trải qua bao nhiêu buổi thực hành?

Học phần Phát triển ứng dụng Web nâng cao đã trải qua 8 buổi thực hành.

1. Có bao nhiêu bài thực hành (Lab) chính thức? Trong số đó, bạn thấy thích nhất là bài thực hành nào? Vì sao?

Có 6 bài Lab thực hành chính thức. Trong đó em cảm thấy thích bài Lab số 1 nhất.

Vì bài Lab đã giúp em hiểu hơn về:

* Cách tổ chức cấu trúc 1 dự án.
* Hướng tiếp cận Code First và cách dùng EF Migrations để tạo cơ sở dữ liệu

1. Những nội dung chính của bài thực hành (Lab) số 3 là gì?

Xây dựng từ đầu đến cuối các chức năng cơ bản dành cho người quản trị của một trang blog đơn giản.

* Cách tạo MVC Area và các thành phần M/V/C để xây dựng các chức năng cho một phân hệ riêng của ứng dụng.
* Cách xây dựng các chức năng thêm/cập nhật/xóa dữ liệu.
* Cách kiểm tra tính hợp lệ của dữ liệu sử dụng các Attribute có sẵn và sử dụng gói thư viện FluentValidation.
* Cách sử dụng gói thư viện Mapster để đơn giản hóa việc sao chép dữ liệu giữa các đối tượng.
* Cách cấu hình và ghi lại nhật ký hệ thống sử dụng gói thư viện NLog.

**CÂU 2: (2 điểm)**

Cho ba lớp Category (danh mục), Candy (kẹo) và CandyContext như sau:

public class Category

{

// Mã số danh mục

public int Id { get; set; }

// Tên danh mục

public string Name { get; set; }

// Cho biết hiển thị lên menu hay không

public bool ShowOnMenu { get; set; }

}

public class Candy

{

// Mã số

public int Id { get; set; }

// Tên mặt hàng kẹo

public string Name { get; set; }

// Giá bán (đơn vị là USD, luôn lớn hơn 0 và

// có phần lẻ thập phân. Ví dụ: 5.99).

public decimal Price { get; set; }

// Ngày hết hạn sử dụng

public DateTime ExpirationDate { get; set; }

// Mã số danh mục (Mỗi mặt hàng kẹo phải thuộc

// đúng một danh mục)

public int CategoryId { get; set; }

public Category Category { get; set; }

}

public class CandyContext : DbContext

{

public DbSet<Category> Categories { get; set; }

public DbSet<Candy> Candies { get; set; }

}

1. Hãy xác định kiểu dữ liệu cho các thuộc tính ShowOnMenu, Price, ExpirationDate, và CategoryId. (Sinh viên sửa trực tiếp mã nguồn ở trên bằng cách thay tên kiểu dữ liệu vào chỗ [……])

* ShowOnMenu: bool
* Price: decimal
* ExpirationDate: DateTime
* CategoryId: int

1. Giả sử, ta có lớp CandyService thực thi interface ICandyService chứa phương thức GetCandiesAsync. Hãy định nghĩa phương thức này để tìm các mặt hàng kẹo chưa hết hạn sử dụng theo tên mặt hàng (nếu giá trị truyền vào khác rỗng), đơn giá (miền giá min-max có giá trị hợp lệ), danh mục (nếu có truyền giá trị) hoặc tên danh mục (tìm chính xác theo tên danh mục nếu truyền giá trị khác rỗng) và sắp xếp kết quả tăng dần theo tên mặt hàng kẹo.

public async Task<IList<Candy>> GetCandiesAsync(

string name = null,

int? categoryId = null,

string categoryName = null,

decimal? minPrice = null,

decimal? maxPrice = null,

CancellationToken cancellationToken = default)

{

// Sinh viên viết mã lệnh ở đây

}

public async Task<IList<Candy>> GetCandiesAsync(

    string name = null,

    int? categoryId = null,

    string categoryName = null,

    decimal? minPrice = null,

    decimal? maxPrice = null,

    CancellationToken cancellationToken = default)

{

    IQueryable<Candy> query = \_dbContext.Candies;

    if (!string.IsNullOrEmpty(name))

    {

        query = query.Where(c => c.Name.Contains(name));

    }

    if (categoryId.HasValue)

    {

        query = query.Where(c => c.CategoryId == categoryId);

    }

    else if (!string.IsNullOrEmpty(categoryName))

    {

        query = query.Where(c => c.Category.Name == categoryName);

    }

    if (minPrice.HasValue)

    {

        query = query.Where(c => c.Price >= minPrice);

    }

    if (maxPrice.HasValue)

    {

        query = query.Where(c => c.Price <= maxPrice);

    }

    query = query.OrderBy(c => c.Name);

    return await query.ToListAsync(cancellationToken);

}

**CÂU 3: (3 điểm)**

Bạn An sử dụng kết quả ở câu 2 để định nghĩa một API Endpoint như sau:

app.MapGet("/candies", (ICandyService service, CandyFilterModel model) =>

{

var candies = await service.GetCandiesAsync(

model.Name, model.CategoryId, model.CategoryName,

model.MinPrice, model.MaxPrice);

return candies;

})

.WithName("GetCandies")

.Produces<Candy>()

.WithOpenApi();

1. Hãy giúp bạn An định nghĩa lớp CandyFilterModel.

public class CandyFilterModel

{

    public string Name { get; set; }

    public int? CategoryId { get; set; }

    public string CategoryName { get; set; }

    public decimal? MinPrice { get; set; }

    public decimal? MaxPrice { get; set; }

}

1. Hãy giúp bạn An tìm những lỗi sai trong đoạn mã trên và sửa lại cho đúng.

* Thiếu từ khóa "async"
* Hàm trả về 1 danh sách các Candy

App

.MapGet("/candies", async (ICandyService service, CandyFilterModel model) =>

{

  var candies = await service.GetCandiesAsync(

    model.Name, model.CategoryId, model.CategoryName,

    model.MinPrice, model.MaxPrice);

  return candies;

})

.WithName("GetCandies")

.Produces<IList<Candy>>()

.WithOpenApi();

1. Hãy giải thích ý nghĩa của toàn bộ đoạn mã trên (sau khi đã sửa).

Đoạn mã trên định nghĩa một API Endpoint cho việc lấy danh sách các kẹo dựa trên các điều kiện lọc từ đối tượng CandyFilterModel. API Endpoint này sử dụng một đối tượng dịch vụ ICandyService để lấy danh sách các kẹo tương ứng với các điều kiện lọc và trả về dữ liệu dạng JSON.

1. Hãy nêu một đề xuất để cải thiện hiệu suất cho API Endpoint trên.

* Sử dụng caching để lưu trữ kết quả truy vấn của API Endpoint, từ đó giảm số lần truy vấn tới đối tượng dịch vụ ICandyService.
* Sử dụng kỹ thuật indexing cho cơ sở dữ liệu lưu trữ các kẹo để tăng tốc độ truy vấn và lọc dữ liệu.
* Sử dụng kỹ thuật paging để giới hạn số lượng kết quả trả về, từ đó giảm tải cho API Endpoint và tăng tốc độ phản hồi.

**CÂU 4: (3 điểm)**

Sau khi xây dựng API Endpoint (ở câu 3) hoàn chỉnh, bạn An tạo một React component – tên là CandyList – để hiển thị danh sách mặt hàng kẹo lên ứng dụng Web. Bạn An định nghĩa mã nguồn cho component này như sau:

import React, { useState } from 'react'

const CandyList = () => {

    const [candies, setCandies] = useState([]);

    fetch(`https://localhost:7076/candies`)

        .then(response => response.json())

        .then(data => setCandies(data));

    return (

        <div class="product-list">

            {candies.map((item) => (

               <div className="product-item">

                    <h1>item.name</h1>

                    <p>$ { item.price }</p>

                </div>

            ))}

        </div>

    )

}

export default CandyList;

1. Hãy cho biết đoạn mã trên chứa những lỗi sai hoặc cảnh báo nào? Giải thích nguyên nhân gây ra lỗi.

* Lỗi về cú pháp: Trong đoạn mã, cần sử dụng từ khóa className thay vì class để đặt lớp cho phần tử div.
* Lỗi về việc gọi API: Đoạn mã sử dụng fetch để gọi API endpoint, tuy nhiên, nó không được gọi trong một phương thức hoặc sự kiện nào cả. Fetch cần được gọi trong một phương thức như useEffect để đảm bảo nó được gọi khi component được render.
* Khi sử dụng hàm map cần thêm thuộc tính key cho thẻ

1. Hãy giúp bạn An cách sửa lỗi và viết lại mã nguồn đúng.
2. Hãy bổ sung mã lệnh để hiển thị dòng thông báo “***Đang tải dữ liệu …***” trong khi chờ kết quả trả về từ API hoặc thông báo lỗi “***Không thể tải dữ liệu***” nếu gặp lỗi khi gọi API.

***Trả lời:***

import React, { useState, useEffect } from 'react';

const CandyList = () => {

  const [candies, setCandies] = useState([]);

  const [isLoading, setIsLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    fetch('https://localhost:7076/candies')

      .then(response => {

        if (!response.ok) {

          throw new Error('Không thể tải dữ liệu');

        }

        return response.json();

      })

      .then(data => {

        setCandies(data);

        setIsLoading(false);

      })

      .catch(error => {

        setError(error);

        setIsLoading(false);

      });

  }, []);

  if (error) {

    return <div>{error.message}</div>;

  }

  return (

    <>

      {isLoading && <div>Đang tải dữ liệu...</div>}

      <div className="product-list">

      {candies.map((item) => (

        <div className="product-item" key={item.id}>

          <h1>{item.name}</h1>

          <p>$ {item.price}</p>

        </div>

      ))}

    </div>

    </>

  );

};

export default CandyList;

**--- HẾT ---**