# إعداد وتوثيق API لقائمة البيتزا

## 1. تحديد الاحتياجات والمخرجات (Contract)

قرِّر ما الذي يريده العميل (Client) من هذا الـ API: ما هي البيانات المدخلة وما هي البيانات المرجعة؟

مثلاً:

* GET /api/menu → ترجع قائمة بيتزا قصيرة (Id, Name, Price)
* GET /api/menu/{id} → ترجع تفاصيل كاملة (Description, AvailableSizes, Toppings, ImageUrl)

## 2. نموذج الدومين (Domain Model / Entity)

تأكد أنّ الـ Pizza (أو اسم الكيان) يحتوي على كلّ الخصائص المطلوبة في الـ DB.

إذا احتجت حقل جديد (مثلاً Description أو ImageUrl) ضعه هنا أولاً.

## 3. تحديث الـ DbContext وMigrations

أضف DbSet<Pizza> أو تأكد من وجوده.

شغّل migration جديد (Add-Migration) ثم Update-Database لتحديث قاعدة البيانات.

## 4. إنشاء DTOs

لقدرتين:

* PizzaSummaryDto للـ GET /api/menu (حفيفة)
* PizzaDetailsDto للـ GET /api/menu/{id} (مفصّلة)

حدّد بالضبط الحقول التي يحتاجها كل واحد.

## 5. إعداد Mapping

إذا تستخدم AutoMapper أو يدويًّا: أنشئ Profile أو دوال تحويل بين Pizza وPizzaSummaryDto/PizzaDetailsDto.

## 6. واجهة الخدمة (Service Interface)

في IPizzaService (أو Repo):

Task<IEnumerable<PizzaSummaryDto>> GetAllAsync();  
Task<PizzaDetailsDto> GetByIdAsync(int id);

## 7. تنفيذ الخدمة (Service / Repository Implementation)

استخدم EF Core لاسترجاع البيانات:

* للكل: context.Pizzas.Select(p => new PizzaSummaryDto { … })
* للواحد: context.Pizzas.FindAsync(id) ثم تحويله.

## 8. إنشاء أو تحديث الـ Controller

في PizzaController تحت [Route("api/menu")]:

[HttpGet]   
public async Task<IActionResult> GetAll() => Ok(await \_pizzaService.GetAllAsync());  
  
[HttpGet("{id}")]   
public async Task<IActionResult> Get(int id) { … }

## 9. تسجيل الـ DI

تأكد من إضافة services.AddScoped<IPizzaService, PizzaService>(); في Program.cs.

## 10. توثيق واختبار

جرّب endpoints على Swagger UI أو Postman.

تأكد من الاستجابة الصحيحة (200 مع البيانات، 404 إذا غير موجود).