**✅ Fiche de Révision – API REST .NET Core (CRUD)**

**1. 🔍 Définition**

Une **API REST** est une interface HTTP qui permet à un client (ex: front-end ou mobile) d’interagir avec des ressources (données) via des méthodes standards :

* **GET** : lecture
* **POST** : création
* **PUT** : mise à jour
* **DELETE** : suppression

**2. 🧩 Structure d’un Contrôleur CRUD**

csharp

CopierModifier

[ApiController]

[Route("api/[controller]")]

public class ProduitsController : ControllerBase

{

private static List<Produit> Produits = new()

{

new Produit { Id = 1, Nom = "PC", Prix = 1000, Categorie = "Informatique" },

new Produit { Id = 2, Nom = "Smartphone", Prix = 800, Categorie = "Téléphonie" }

};

[HttpGet]

public ActionResult<IEnumerable<Produit>> GetAll() => Ok(Produits);

[HttpGet("{id}")]

public ActionResult<Produit> GetById(int id)

{

var produit = Produits.FirstOrDefault(p => p.Id == id);

return produit == null ? NotFound() : Ok(produit);

}

[HttpGet("search")]

public ActionResult<IEnumerable<Produit>> GetByCategorie([FromQuery] string categorie)

{

var result = Produits.Where(p => p.Categorie.Equals(categorie, StringComparison.OrdinalIgnoreCase));

return Ok(result);

}

[HttpPost]

public ActionResult<Produit> Create([FromBody] Produit produit)

{

produit.Id = Produits.Max(p => p.Id) + 1;

Produits.Add(produit);

return CreatedAtAction(nameof(GetById), new { id = produit.Id }, produit);

}

[HttpPut("{id}")]

public IActionResult Update(int id, [FromBody] Produit produitMaj)

{

var produit = Produits.FirstOrDefault(p => p.Id == id);

if (produit == null) return NotFound();

produit.Nom = produitMaj.Nom;

produit.Prix = produitMaj.Prix;

produit.Categorie = produitMaj.Categorie;

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var produit = Produits.FirstOrDefault(p => p.Id == id);

if (produit == null) return NotFound();

Produits.Remove(produit);

return NoContent();

}

}

**3. 🛠️ Annotations importantes**

| **Annotation** | **Description** |
| --- | --- |
| [ApiController] | Ajoute validation automatique du modèle, binding intelligent, gestion des erreurs |
| [Route("api/[controller]")] | Détermine l’URL racine du contrôleur |
| [HttpGet], [HttpPost], [HttpPut], [HttpDelete] | Spécifient la méthode HTTP autorisée |
| [FromQuery], [FromBody] | Contrôlent la source des paramètres (URL ou JSON) |

**4. 📌 Cas d’usage réels**

* Microservices exposant des données (Produits, Clients, Utilisateurs…)
* Backend pour application Angular/React/Mobile
* Portails d'administration ou de gestion
* Interfaces entre systèmes (ERP, SaaS)

**5. 🚨 Erreurs fréquentes**

| **Erreur** | **Cause** | **Solution** |
| --- | --- | --- |
| 404 Not Found | Mauvais route ou ID inexistant | Vérifier les routes dans [Route] ou Swagger |
| 400 Bad Request | JSON mal formé | Vérifier Content-Type: application/json |
| 415 Unsupported Media Type | Corps non parsé | Ajouter [FromBody] et Content-Type |
| 500 Internal Server Error | NullReference, index, etc. | Vérifier null et logs |
| Pas de point d'arrêt activé | Debug non attaché | Lancer en F5 / Vérifier la configuration |

**6. ✅ Bonnes pratiques**

* Utiliser CreatedAtAction() dans le POST
* Retourner NoContent() dans le PUT et DELETE
* Nommer les routes explicitement ([HttpGet("search")], etc.)
* Découpler la logique métier (via services à terme)
* Gérer les exceptions globalement avec UseExceptionHandler()
* Documenter avec Swagger (AddSwaggerGen())

**7. 🧠 À savoir pour les entretiens**

* Savoir **simuler** les appels REST avec Swagger, Postman ou .http
* Être capable d’écrire une méthode GET/POST de mémoire
* Savoir discuter des **codes de retour HTTP**
* Expliquer l’intérêt du CreatedAtAction vs un Ok()
* Connaître les alternatives REST (GraphQL, gRPC…)