يعني ايه OOP؟ واذكر مفاهيمها الأساسية؟

- **الإجابة:**

  OOP = Object Oriented Programming. يعني بنبرمج باستخدام الكلاسات (Classes) والكائنات (Objects).

  المفاهيم الأساسية:

  - **Encapsulation (التغليف)**
  - **Inheritance (الوراثة)**
  - **Polymorphism (تعدد الأشكال)**
  - **Abstraction (التجريد)**
- **نصيحة:**

  مهم جدا تكون عارف أمثلة عملية مش بس تحفظ التعاريف.

2. يعني ايه Encapsulation؟

- **الإجابة:**

  تغليف البيانات والوظائف جوه الكلاس ومنع التلاعب المباشر بيهم، عن طريق الخصائص (Properties) والميثودز (Methods).
- **نصيحة:**

  لو سألك قولي: "بنستخدم Access Modifiers زي private/protected/public عشان نتحكم في الوصول".

3. يعني ايه Inheritance؟

- **الإجابة:**

  كلاس بياخد خصائص ودوال من كلاس تاني، يعني إعادة استخدام للكود (Code Reusability).
- **نصيحة:**

  اذكر ان الكلاس الأب (Base class) والكلاس الابن (Derived class).

4. يعني ايه Polymorphism؟

- **الإجابة:**

  قدرة الكائن إنه ياخد أكتر من شكل. مثال: Override أو Overload للميثودز.
- **نصيحة:**

  متنساش تقول Inheritance شرط أساسي للـ Polymorphism.

5. يعني ايه Abstraction؟

- **الإجابة:**

  إخفاء التفاصيل الداخلية وإظهار المهم بس. زي لما تستخدم Driver للعربية مش لازم تفهم الموتور شغال ازاي.
- **نصيحة:**
- **Interface و Abstract Class أدوات بتعمل Abstraction.**

6. يعني ايه LINQ؟

- **الإجابة:**

  Language Integrated Query. طريقة للكتابة Queries جوه كود #C على الداتا.

- **نصيحة:**

  عارف الفرق بين LINQ To Objects وLINQ To Entities؟ مهم جدا.

---

7. اكتب LINQ Query تجيب كل المنتجات اللي سعرها أكبر من 100؟

- **الإجابة:**

```
var products = context.Products.Where(p => p.Price > 100).ToList();
```

- **نصيحة:**

  اتأكد انك حافظ Structure عام للـ Queries.

---

8. الفرق بين Method Syntax و Query Syntax في LINQ؟

- **الإجابة:**
- **Method Syntax = استخدام الدوال (.Where, Select, etc)**
- **Query Syntax = كتابة شبيهة بالـ**
- **SQL (from p in products where p.Price > 100 select p);**
- **نصيحة:**

  معظم الشركات بتستخدم Method Syntax أكتر.

---

9. اشرح يعني ايه Deferred Execution في LINQ؟

- **الإجابة:**

  Query مش بتتنفذ لحظة كتابتها، بتتنفذ لما تتطلب الداتا فعليًا (زي ToList).

- **نصيحة:**

  دي نقطة بيسألو فيها كتير، فاهمها مش حافظها.

---

10. ازاي تعمل Join بين جدولين بالـ LINQ؟

- **الإجابة:**

```
var result = from o in orders
             join c in customers on o.CustomerId equals c.Id
```

```
select new { o.OrderNumber, c.CustomerName };
```

- **نصيحة:**

  متنساش انك تقدر تعمل Joins أكتر من جدول.

---

11. يعني ايه ORM؟ وايه علاقة EF بيه؟

- **الإجابة:**

  ORM = Object Relational Mapping.

  EF Core هو ORM بيعمل Map بين الكلاسات في الكود والجداول في قاعدة البيانات.
- **نصيحة:**

  بيحبوا يسألوك على العلاقة بين الكلاس والجدول.

---

12. يعني ايه DbContext؟

- **الإجابة:**

  كلاس مسؤول عن إدارة الاتصال بالداتا بيز، وتحميل وحفظ الداتا.
- **نصيحة:**

  أنت في كل مرة تتعامل مع الداتا لازم تتعامل عن طريق DbContext.

---

13. يعني ايه Migration؟

- **الإجابة:**

  طريقة تـ Track فيها التغييرات اللي بتحصل في الموديلات (Models) وتحولها لتغييرات على الداتا بيز.
- **نصيحة:**

  مهم جدا تعرف أوامر:

  Add-Migration MigrationName

  Update-Database

---

14. اشرح HasOne و HasMany و WithOne و WithMany؟

- **الإجابة:**
- **HasOne: علاقة كيان له واحد (One To One).**
- **HasMany: علاقة كيان له أكتر من واحد (One To Many).**
- **WithOne / WithMany: بتحدد اتجاه العلاقة للطرف التاني.**
- **نصيحة:**

  ارسم العلاقة في دماغك قبل ما تكتب الكود.

---

15. يعني ايه Lazy Loading؟

- **الإجابة:**

  البيانات المتعلقة (Related Data) مش بتتحمل غير لما تطلبها.
- **نصيحة:**

  لو عملت Disable لـ Lazy Loading استخدم Eager Loading مع Include.

---

16. يعني ايه Normalization؟

- **الإجابة:**

  تقسيم الداتا لجداول صغيرة عشان نقلل التكرار (Redundancy) ونحسن الأداء.
- **نصيحة:**

  اعرف أول 3 Forms (1NF, 2NF, 3NF).

---

17. الفرق بين Primary Key و Unique Key؟

- **الإجابة:**
- **Primary Key: بيضمن ان العمود Unique وNot Null.**
- **Unique Key: بيضمن ان العمود Unique بس ممكن يكون Null مرة واحدة.**
- **نصيحة:**

  خلي بالك Primary Key لكل جدول واحد بس.

---

18. يعني ايه Index؟ وليه بنستخدمه؟

- **الإجابة:**

  أداة بتحسن سرعة البحث والاستعلامات على الداتا بيز.
- **نصيحة:**

  ممكن يبقى السبب الأول ليه السيرفر بطيء هو إن الجدول محتاج Index.

---

19. يعني ايه Stored Procedure؟

- **الإجابة:**

  مجموعة أوامر SQL محفوظة بتتعامل معاها كأنها Function.
- **نصيحة:**

  ممكن تـ Optimize Queries تقيلة عن طريق Stored Procedures.

---

20. اشرح الفرق بين Inner Join و Left Join؟

- **الإجابة:**
- **Inner Join: بيرجع بس السجلات اللي ليها ماتش في الجدولين.**
- **Left Join: بيرجع كل سجلات الجدول الأول + الماتش من الجدول التاني (ولو مفيش ماتش بيحط Nulls).**
- **نصيحة:**

  ارسم Join في دماغك عشان متتلغبطش.

---

21. يعني ايه MVC؟

- **الإجابة:**

  Model - View - Controller:

  تقسيم المشروع 3 أقسام:
- **Model: داتا.**
- **View: UI.**
- **Controller: الوسيط اللي بيربطهم.**
- **نصيحة:**

  عارف Request Lifecycle؟ ممكن يسألك.

---

22. يعني ايه Routing في MVC؟

- **الإجابة:**

  تحديد الـ URL هيروح لأنهي Controller وأنهي Action.
- **نصيحة:**

  RouteConfig.cs مهم، وخلي بالك من ترتيب الروتات.

---

23. يعني ايه ViewBag و TempData؟

- **الإجابة:**
- **ViewBag: بيشيل داتا خلال نفس الـ Request.**
- **TempData: بيشيل داتا بين Request و Request.**
- **نصيحة:**

  ViewBag عبارة عن Dynamic Object.

---

24. اشرح الفرق بين ActionResult و JsonResult؟

- **الإجابة:**
- **ActionResult: نوع عام بيرجع أي نوع من الـ Responses.**
- **JsonResult: بيرجع Data بصيغة JSON.**
- **نصيحة:**

  ممكن ترجع JSON باستخدام return Json(data);

---

25. يعني ايه Model Validation في MVC؟

- **الإجابة:**

  التأكد إن الداتا اللي جاية من اليوزر سليمة قبل ما تدخلها الداتا بيز.
- **نصيحة:**

  استعمل DataAnnotations زي [Required], [StringLength].

26. يعني ايه Web API؟

- **الإجابة:**

  هي خدمة بتسمح إنك تبني APIs تقدر الـ Frontend (موبايل، ويب) يتعامل معاها.
- **نصيحة:**

  Web API أخف وأسرع من MVC لما الموضوع كله Data Exchange بس.

---

27. الفرق بين API Controller و MVC Controller؟

- **الإجابة:**
- **API Controller بيرجع Data زي JSON.**
- **MVC Controller بيرجع HTML Pages (Views).**
- **نصيحة:**

  API Controller بيورك عليه [ApiController] Attribute.

---

28. يعني ايه RESTful API؟

- **الإجابة:**

  طريقة لبناء APIs بتركز على استخدام HTTP Methods (GET, POST, PUT, DELETE) بطريقة قياسية.
- **نصيحة:**

  مهم تعرف RESTful Conventions كويس.

---

29. الفرق بين GET و POST و PUT و DELETE؟

- **الإجابة:**

- **GET: يجيب داتا.**
- **POST: يعمل إضافة داتا جديدة.**
- **PUT: يعدل داتا موجودة.**
- **DELETE: يمسح داتا.**
- **نصيحة:**

  ركز كويس في الفرق بين PUT و PATCH.

---

30. يعني ايه Status Codes؟ ادِيني أمثلة؟

- **الإجابة:**

  كود بيرجع مع كل Response يوضح حالة الطلب:

  - **OK 200**
  - **Created 201**
  - **Bad Request 400**
  - **Unauthorized 401**
  - **Not Found 404**
  - **Internal Server Error 500**
- **نصيحة:**

  مش تحفظهم، افهمهم مع كل سيناريو.

---

31. يعني ايه Model Binding في Web API؟

- **الإجابة:**

  تحويل الداتا اللي جايه من الريكوست (JSON مثلًا) إلى Object في الكود.
- **نصيحة:**

  خلي بالك من [FromBody] و [FromQuery] و [FromRoute].

---

32. يعني ايه Middleware في ASP.NET Core؟

- **الإجابة:**

  كود بيتنفذ في نص البايبلاين بتاع الـ Request/Response Cycle.
- **نصيحة:**

  الـ Middlewares بترتب في Program.cs أو Startup.cs.

---

33. اذكر أمثلة على Middleware جاهزة؟

- **الإجابة:**

- Authentication Middleware
- Authorization Middleware
- Exception Handling Middleware
- Static Files Middleware
- نصيحة:

ممكن كمان تكتب Middleware بنفسك.

---

34. ازاي تعمل Custom Middleware؟

- الإجابة:

```
public class MyMiddleware
{
    private readonly RequestDelegate _next;
    public MyMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public async Task InvokeAsync(HttpContext context)
    {
        // Logic هنا
        await _next(context);
    }
}
```

وتسجله بـ app.UseMiddleware<MyMiddleware>();

- نصيحة:

خلي Middleware خفيف وسريع عشان ميبطءش السيستم.

---

35. يعني ايه Dependency Injection؟

- الإجابة:

طريقة تخلي الكلاسات تاخد الـ Dependencies اللي محتاجاها من برة بدل ما تنشئها بنفسها.

- نصيحة:

فهم الفرق بين Scoped و Transient و Singleton.

---

36. اذكر أنواع الـ Dependency Injection lifetimes؟

- **الإجابة:**
- **Singleton: كائن واحد بيتعمل مرة واحدة طول عمر الأبلكيشن.**
- **Scoped: كائن بيتعمل لكل Request.**
- **Transient: كل مرة تستدعيه بيتعمل Object جديد.**
- **نصيحة:**

لو فيه State، خليه Scoped أو Transient مش Singleton.

---

37. يعني ايه Fluent Validation؟

- **الإجابة:**

مكتبة تانية بنستخدمها نعمل بيها Validation بشكل مرتب وقوي أكتر من الـ DataAnnotations العادية.

- **نصيحة:**

بتكتب RuleSet لكل Property.

---

38. اكتب مثال صغير على Fluent Validation Rule؟

- **الإجابة:**

```
public class UserValidator : AbstractValidator<User>
{
    public UserValidator()
    {
        RuleFor(x => x.Name).NotEmpty().WithMessage("Name is required");
        RuleFor(x => x.Email).EmailAddress();
    }
}
```

نصيحة:

ممكن تجمع أكتر من Rule على نفس الـ Property.

---

39. يعني ايه Authorization و Authentication؟

- **الإجابة:**
- **Authentication: تتأكد مين اللي بيطلب (تسجيل الدخول).**
- **Authorization: تتأكد عنده صلاحيات يعمل ايه.**
- **نصيحة:**

أوقات بيطلبوا منك ترسمهم زي بروسيس فلو.

---

40. يعني ايه JWT؟ وليه بنستخدمه؟

- **الإجابة:**

  JSON Web Token. بيستخدموه عشان يحفظ الـ Authentication info جوه Token يتبعت في كل Request.

- **نصيحة:**

  افهم مكونات الـ JWT (Header + Payload + Signature).

---

41. يعني ايه Claims؟

- **الإجابة:**

  هي الداتا اللي بتتحط جوه الـ Token زي (UserId, Role).

- **نصيحة:**

  ممكن تضيف Claims مخصصة حسب المشروع.

---

42. يعني ايه CORS؟

- **الإجابة:**

  Cross-Origin Resource Sharing. بيسمح لأبلكيشن من دومين معين يتكلم مع سيرفر بدومين تاني.

- **نصيحة:**

  في Web API ساعات لازم تضيف app.UseCors().

---

43. ازاي تمنع CORS Error في API؟

- **الإجابة:**

  تضيف Middleware CORS وتحدد السماحية في Program.cs:

```
builder.Services.AddCors();
app.UseCors(x => x.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod());
```

- **نصيحة:**

  في الحقيقي اختار السماحية بناءً على الأمان مش AllowAny لكل حاجة.

---

44. يعني ايه Exception Handling؟

- **الإجابة:**

  طريقة تمسك بيها الأخطاء اللي بتحصل أثناء تنفيذ الكود وتعالجها بشكل نظيف.
- **نصيحة:**

  متخليش الكود يقع قدام اليوزر مهما حصل.

---

45. اذكر طرق التعامل مع Exceptions في Web API؟

- **الإجابة:**
- **Try-Catch جوا الأكشن**
- **Middleware لالتقاط كل الأخطاء**
- **UseExceptionHandler Middleware**
- **نصيحة:**

  ممكن كمان ترجع Custom Error Response.

---

46. يعني ايه Filters في Web API؟

- **الإجابة:**

  أكواد بتتنفذ قبل أو بعد الأكشنز زي AuthorizationFilter، ExceptionFilter.
- **نصيحة:**

  بتفيد في تسجيل الـ Logs أو التحقق قبل تنفيذ الأكشن.

---

47. يعني ايه Global Error Handler؟

- **الإجابة:**

  Middleware بيمسك أي Exception يحصل في أي حتة في التطبيق ويرجع Response مرتب.
- **نصيحة:**

  الـ Production Error Message متبقاش فيها تفاصيل عشان الأمان.

---

48. اشرح الفرق بين 401 و 403 Status Codes؟

- **الإجابة:**
- **401 Unauthorized: مش مسجل دخول.**
- **403 Forbidden: مسجل دخول بس مش عنده صلاحيات يعمل العملية دي.**
- **نصيحة:**

  احفظهم كويس لأنهم Common قوي في الأسئلة.

---

49. يعني ايه DTO؟

- **الإجابة:**

  Data Transfer Object. Object خفيف بنستخدمه بدل الـ Entity عشان ننقل بس البيانات المطلوبة.
- **نصيحة:**

  ميفضلش تبعت الـ Entity كاملة للـ Client.

---

50. الفرق بين DTO و ViewModel؟

- **الإجابة:**
- **DTO: لنقل الداتا بين Layers.**
- **ViewModel: لتحضير الداتا للعرض في View.**
- **نصيحة:**

  مش كل ViewModel يبقى DTO والعكس صحيح.

---

51. يعني ايه Entity Framework Core؟

- **الإجابة:**

  هو ORM (Object Relational Mapper)، بيخليك تتعامل مع قاعدة البيانات بالكود بدل ما تكتب SQL بنفسك.
- **نصيحة:**

  اعرف الفرق بين Code-First و Database-First.

---

52. يعني ايه DbContext؟

- **الإجابة:**

  كلاس أساسي بيربط الكود بتاعك بالـ Database، وبيتحكم في الـ CRUD Operations.
- **نصيحة:**

  تخلي DbContext مرتب وفيه DbSets لكل كيان.

---

53. الفرق بين Code First و Database First Approach؟

- **الإجابة:**
- **Code First: بتكتب الكود الأول وهو يولدلك قاعدة البيانات.**
- **Database First: عندك قاعدة بيانات، وهو بيولدلك الكود.**
- **نصيحة:**

  Code First مناسب لو أنت لسه بتبدأ المشروع.

---

54. يعني ايه Migration؟

- **الإجابة:**

  طريقة تخليك تحدث التعديلات اللي عملتها في الكود بتاع الموديلات للـ Database.

- **نصيحة:**

  دايمًا بعد ما تعدل كيان، لازم تعمل Migration جديدة.

---

55. اكتب أوامر الـ Migration الأساسية؟

- **الإجابة:**

Add-Migration MigrationName

Update-Database

Remove-Migration

- **نصيحة:**

  متنساش تسمي الـ Migration اسم بيعبر عن التغيير.

---

56. يعني ايه LINQ؟

- **الإجابة:**

  لغة بتسهل عليك كتابة استعلامات (Queries) عالـ Data Collections زي الـ Database أو Lists.

- **نصيحة:**

  فيه LINQ to Objects و LINQ to Entities.

---

57. اكتب مثال على LINQ Query تجيب كل المستخدمين؟

- **الإجابة:**

```
var users = _context.Users.ToList();
```

- **نصيحة:**

  دايما حط AsNoTracking لو مش ناوي تعدل عالبيانات.

---

58. الفرق بين IQueryable و IEnumerable؟

- **الإجابة:**
- **IQueryable: بيجهز الاستعلام ومبيتنفذش غير لما تحتاج الداتا.**
- **IEnumerable: بيحمل الداتا كلها في الذاكرة.**
- **نصيحة:**

  في الـ Database Operations، دايمًا اشتغل بـ IQueryable.

---

59. يعني ايه Lazy Loading و Eager Loading؟

- **الإجابة:**
- **Lazy Loading: بيحمل البيانات المرتبطة وقت لما تطلبها.**
- **Eager Loading: بيحملها مع الاستعلام الأساسي باستخدام Include.**
- **نصيحة:**

  استخدم Eager Loading عشان توفر Calls زيادة.

---

60. اكتب مثال على Include في EF Core؟

- **الإجابة:**

```
var orders = _context.Orders.Include(x => x.Customer).ToList();
```

- **نصيحة:**

  تقدر Include كذا Table ورا بعض كمان.

---

61. يعني ايه Tracking vs NoTracking في EF Core؟

- **الإجابة:**
- **EF Tracking: بيتابع التغييرات اللي بتحصل عالأوبجكت.**
- **EF NoTracking: مبيتابعش وبالتالي الأداء بيبقى أسرع.**
- **نصيحة:**

  لو شغال قراءة بس، خلي AsNoTracking().

---

62. اكتب مثال علي SaveChanges()؟

- **الإجابة:**

```
_context.SaveChanges();
```

- **نصيحة:**

بعد أي إضافة أو تعديل لازم SaveChanges.

---

63. يعني ايه Stored Procedure؟

- **الإجابة:**

كود SQL محفوظ جوا قاعدة البيانات، تقدر تنفذه مرة ورا مرة بدل ما تكتب نفس الكود كل مرة.

- **نصيحة:**

في السيستمات الكبيرة، بيحبوا Stored Procedures للأداء والأمان.

---

64. ازاي تنفذ Stored Procedure من EF Core؟

- **الإجابة:**

```
_context.Users.FromSqlRaw("EXEC GetAllUsers").ToList();
```

- **نصيحة:**

خلي بالك من SQL Injection لما تستخدم FromSqlRaw.

🎯 [(Frontend Basics (HTML, CSS, JavaScript, TypeScript, Bootstrap]

65. يعني ايه HTML؟

- **الإجابة:**

هي لغة وصفية (Markup Language) بتبني بيها الهيكل الأساسي للصفحة (Structure).

- **نصيحة:**

اعرف الفرق بين Inline Elements و Block Elements.

---

66. يعني ايه CSS؟

- **الإجابة:**
  هي لغة بتتحكم في الشكل والتنسيق (Style) بتاع صفحة الـ HTML.
- **نصيحة:**
  اعرف الفرق بين Inline Style, Internal Style و External Style.

---

67. يعني ايه JavaScript؟

- **الإجابة:**
  هي لغة برمجة بتخليك تضيف حيوية (Interactivity) لصفحات الويب زي البوتونز اللي تضغط عليها تفتح حاجة.
- **نصيحة:**
  ركز على Events و DOM Manipulation.

---

68. يعني ايه DOM؟

- **الإجابة:**
  Document Object Model. هو تمثيل شجري لكل عناصر صفحة الويب بحيث تقدر تتعامل معاهم بالكود.
- **نصيحة:**
  اعرف ازاي تمسك Element من الـ DOM وتلعب فيه.

---

69. يعني ايه Bootstrap؟

- **الإجابة:**
  فريم وورك CSS جاهز بيساعدك تعمل تصميم مرتب وسريع فيه Grid System و Ready-made Components.
- **نصيحة:**
  ركز على الـ Grid System (12 Columns) كويس.

---

70. يعني ايه Responsive Design؟

- **الإجابة:**
  تصميم بيتغير شكله تلقائيًا عشان يناسب الموبايل، التابلت، والديسكتوب.
- **نصيحة:**
  Bootstrap بيساعدك تعمله بسهولة.

---

71. يعني ايه TypeScript؟

- **الإجابة:**

  هي نسخة مطورة من JavaScript، بتحطلك Strong Typing وكائنات زي الـ Classes والـ Interfaces.

- **نصيحة:**

  كل كود TypeScript في الآخر بيتحول لـ JavaScript.

---

72. اذكر مميزات TypeScript؟

- **الإجابة:**
- **Strong Typing**
- **Better Code Organization**
- **Easier to Debug**
- **نصيحة:**

  في Angular بيستخدموا TypeScript أساسي.

---

73. يعني ايه Event في JavaScript؟

- **الإجابة:**

  أي حاجة بتحصل على الصفحة زي كليك، هوفر، كتابة.

- **نصيحة:**

  اعرف ازاي تعمل Event Listener.

---

74. يعني ايه Promise في JavaScript؟

- **الإجابة:**

  طريقة تدير بيها العمليات الغير متزامنة (Asynchronous) بشكل أفضل.

- **نصيحة:**

  Promises ليها 3 حالات: Pending, Resolved, Rejected.

---

75. يعني ايه async/await؟

- **الإجابة:**

  Syntax حديث للتعامل مع Promises بشكل أبسط وأوضح.

- **نصيحة:**

  تحط كلمة async قدام الفانكشن وتستخدم await جوه الكود.

---

76. ايه هو الـ Clean Architecture؟

- **الإجابة:**

  هو طريقة تصميم بتفصل بين الـ Layers عشان تبقى الكود نظيف وسهل التعديل والتوسيع، بحيث مفيش اعتمادية بين الـ Layers.
- **نصيحة:**

  تخلي كل Layer مسؤول عن حاجة واحدة فقط.

---

77. ايه هي الـ Layers في الـ Clean Architecture؟

- **الإجابة:**
- **(Presentation Layer (UI**
- **(Application Layer (Business Logic**
- **(Domain Layer (Entities**
- **(Infrastructure Layer (Database, External APIs**
- **نصيحة:**

  تخلي الـ Domain Layer خالي من أي تفاصيل خاصة بالبقية، زي الـ Database.

---

78. يعني ايه Dependency Inversion Principle؟

- **الإجابة:**

  إن الكود يتبع مبدأ الـ Inversion of Control، بحيث تعتمد الطبقات العليا على واجهات (Interfaces) مش على الطبقات السفلى.
- **نصيحة:**

  دايماً خلي الـ Dependencies بتاعك في الـ Constructor بدلاً من إنك تخلقها جوه الكود.

---

79. ايه هو الـ SOLID؟

- **الإجابة:**

  مجموعة مبادئ لتصميم الكود بحيث يكون مرن وسهل التوسع:

  1. **S - Single Responsibility Principle**
  2. **O - Open/Closed Principle**
  3. **L - Liskov Substitution Principle**
  4. **I - Interface Segregation Principle**

5. **D - Dependency Inversion Principle**

- **نصيحة:**

  فهم كل مبدأ هيساعدك تطبق الـ Clean Architecture بشكل صحيح.

---

80. اشرح Single Responsibility Principle؟

- **الإجابة:**

  مبدأ إن كل كلاس أو موديول يبقى ليه مسؤولية واحدة بس.

- **نصيحة:**

  لو الكلاس بدأ يتكبر، فكر تقسيمه لعدد من الكلاسات.

---

81. اشرح Open/Closed Principle؟

- **الإجابة:**

  يعني إن الكود يكون مفتوح للتوسيع ومغلق للتعديل. يعني لو عايز تضيف وظيفة جديدة، تضيف كود جديد بدل ما تعدل في الكود القديم.

- **نصيحة:**

  استخدم الـ Interfaces أو الـ Abstract Classes عشان تقدر توسع بدون تعديل.

---

82. اشرح Liskov Substitution Principle؟

- **الإجابة:**

  لو عندك كلاس أساسي وكلاس فرعي، لازم تقدر تستخدم الكلاس الفرعي مكان الكلاس الأساسي بدون ما تسبب مشاكل.

- **نصيحة:**

  تأكد إن الكلاسات الفرعية تحترم سلوك الكلاسات الأساسية.

---

83. اشرح Interface Segregation Principle؟

- **الإجابة:**

  إنه من الأفضل تقسيم الـ Interfaces الكبيرة إلى عدة Interfaces صغيرة، عشان الكلاسات تستخدم بس الـ Methods اللي هي محتاجاها.

- **نصيحة:**

  مفيش كلاس يستخدم أكتر من Interface واحد.

---

84. اشرح Dependency Inversion Principle؟

- **الإجابة:**

  التأكد إن الكود يعتمد على الـ Interfaces أو Abstractions، مش الكلاسات الملموسة (Concrete Classes).
- **نصيحة:**

  لو عندك كود معتمد على كلاس معين، حاول تستخدم Interface عشان تكون مرن أكتر.

---

85. ايه هو الـ Repository Pattern؟

- **الإجابة:**

  هو طريقة لتنظيم الكود بحيث كل عملية مع الـ Database تتم من خلال Repository، وده يخفف التداخل بين الـ Data Layer وبقية الكود.
- **نصيحة:**

  Repository Pattern بيسهل الـ Testing ويخلي الـ Data Access Layer منعزل.

---

86. ايه هو الـ Factory Pattern؟

- **الإجابة:**

  هو طريقة لتكوين كائنات (Objects) من خلال فابريك (Factory) بدلاً من إنشاء الكائن مباشرة.
- **نصيحة:**

  استخدمه لما يكون عندك كائنات معقدة تحتاج تنشأها بطريقة مرنة.

---

87. ايه هو الـ Singleton Pattern؟

- **الإجابة:**

  نمط تصميم بيخلي الكلاس ده يكون ليه نسخة واحدة فقط خلال دورة حياة البرنامج.
- **نصيحة:**

  لو عندك حاجة زي إعدادات النظام أو الـ Database Connection، ممكن تستخدمه.

---

88. ايه هو الـ Observer Pattern؟

- **الإجابة:**

  نمط تصميم يسمح للكائنات المتعددة (Observers) بالاشتراك في التحديثات لما يحصل تغير في الكائن (Subject).
- **نصيحة:**

  مفيد في الأنظمة اللي بتحتاج توافر ملاحظة من عدة أطراف.

---

89. ايه هو الـ Strategy Pattern؟

- **الإجابة:**

  نمط تصميم بيخليك تستخدم استراتيجيات مختلفة لأداء وظيفة معينة، ويكون سهل التبديل بين الاستراتيجيات دي في وقت التنفيذ.
- **نصيحة:**

  مفيد لو عندك أكثر من طريقة لتحقيق نفس الهدف.

---

90. ايه هو الـ Adapter Pattern؟

- **الإجابة:**

  نمط تصميم بيخليك تربط بين واجهتين (Interfaces) مختلفتين بدون ما تغير الكود الموجود.
- **نصيحة:**

  استخدمه لما تحتاج تربط بين أنظمة قديمة وجديدة.

---

91. ايه هو الـ MVC Pattern؟

- **الإجابة:**

  نمط تصميم بيقسم التطبيق لثلاث أجزاء:

  1. **Model: البيانات.**
  2. **View: العرض.**
  3. **Controller: المعالجة.**
- **نصيحة:**

  ده النمط المستخدم في ASP.NET MVC.

---

92. الفرق بين ASP.NET MVC و ASP.NET Web API؟

- **الإجابة:**
- **ASP.NET MVC: بيستخدم لعرض الصفحات (HTML).**
- **ASP.NET Web API: بيستخدم لبناء APIs يتعامل مع البيانات.**
- **نصيحة:**

  لو عايز تبني API بس، استخدم Web API.

---

93. ازاي تستخدم Routing في ASP.NET MVC؟

- **الإجابة:**

  بتستخدم Attribute Routing أو Conventional Routing عشان توجه الـ Requests للـ Controllers.

- **نصيحة:**

  خلي الـ Routes بتاعتك واضحة ومنظمة.

---

94. يعني ايه ActionResult في ASP.NET MVC؟

- **الإجابة:**

  هو نوع البيانات اللي بيرجعها الـ Action Method بعد تنفيذ العمليات.

- **نصيحة:**

  ممكن يرجع ViewResult, RedirectToAction, JsonResult، حسب احتياجك.

---

95. ايه الفرق بين POST و GET في الـ Web API؟

- **الإجابة:**
- **GET: لاسترجاع البيانات.**
- **POST: لإرسال بيانات جديدة.**
- **نصيحة:**

  استخدم GET للـ Read و POST للـ Create.

---

96. ايه هو الـ Middleware في ASP.NET Core؟

- **الإجابة:**

  هو كود بيشتغل في الـ Pipeline بتاع الـ HTTP Request وبيمر على كل Request بيجي للتطبيق.

- **نصيحة:**

  ممكن تستخدمه عشان تضيف وظائف زي الـ Logging أو Authentication.

---

97. ازاي تضيف Middleware في ASP.NET Core؟

- **الإجابة:**

```
public void Configure(IApplicationBuilder app)
{
    app.UseMiddleware<MyCustomMiddleware>();
}
```

- **نصيحة:**

  خلي الـ Middleware مرتب عشان يكون سهل تتبع الأخطاء.

---

98. يعني ايه JWT؟

- **الإجابة:**

  JSON Web Token، بيستخدم في الـ Authentication بين العميل والخادم بشكل آمن.
- **نصيحة:**

  تأكد إنك تحفظ الـ Secret Key في مكان آمن، زي البيئة (Environment Variables).

---

99. ازاي تستخدم Dependency Injection في ASP.NET Core؟

- **الإجابة:**

```
services.AddScoped<IMyService, MyService>();
```

- **نصيحة:**

  استخدمه عشان تعزل الكود وتخلي الكائنات قابلة لإعادة الاستخدام.

---

100. ازاي تكتب Unit Test في ASP.NET Core؟

- **الإجابة:**

  تستخدم xUnit أو NUnit وتكتب Test Methods تتأكد إن الكود شغال زي ما هو مفروض.
- **نصيحة:**

  استخدم Mocking في الـ Tests لتجنب الاعتماد على قواعد البيانات أو أي خدمات خارجية.