Mémento C# - Guide de Développement

1. Commandes Dotnet Essentielles

1.1 Création de Projet

```
# Créer une nouvelle solution
dotnet new sln -n MonProjet

# Créer un projet Web API
dotnet new webapi -n MonProjet.API

# Ajouter un projet à la solution
dotnet sln add MonProjet.API/MonProjet.API.csproj
```

1.2 Gestion des Dépendances

```
# Ajouter un package NuGet
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package AutoMapper

# Restaurer les packages
dotnet restore

# Mettre à jour les packages
dotnet update
```

1.3 Exécution et Build

```
# Exécuter en mode développement
dotnet run

# Build le projet
dotnet build

# Build en mode release
dotnet build -c Release

# Publier le projet
dotnet publish -c Release
```

1.4 Entity Framework

```
# Ajouter une migration
dotnet ef migrations add InitialCreate

# Appliquer les migrations
dotnet ef database update

# Supprimer la dernière migration
dotnet ef migrations remove
```

2. Architecture et Organisation

2.1 Structure des Namespaces

```
// Organisation par domaine fonctionnel
namespace MonProjet.Models;
namespace MonProjet.DTO;
namespace MonProjet.Controllers;
namespace MonProjet.Services;
```

2.2 Séparation des Couches

```
// Models - Définition des entités
public class Student : User
{
    [Required] public DateOnly BirthDate { get; set; }
    [Required, MaxLength(150)] public string? Address { get; set; }
    // ...
}

// DTOs - Transfert de données
public class CreateStudentDto
{
    [Required, MaxLength(50)] public string? Firstname { get; set; }
    [Required, MaxLength(50)] public string? Lastname { get; set; }
    // ...
}

// Controllers - Gestion des requêtes HTTP
[ApiController]
[Route("API/[controller]")]
public class StudentsController : Controller
{
    // ...
}
```

3. Models et Entités

3.1 Data Annotations

```
public class School
{
    [Key] public int Id { get; set; }
    [Required, MaxLength(50)] public string? Name { get; set; }
    [Required, MaxLength(150)] public string? Address { get; set; }
    [MaxLength(150)] public string? AddressSecondLine { get; set; }
    [Required, MaxLength(5), RegularExpression("[0-9]{5}")] public string? ZipCode { get; set; }
    // ...
}
```

3.2 Relations entre Entités

```
public class Class
{
    [Required] public int YearId { get; set; }
    public Year? Year { get; set; }

    [Required] public int SpecialityId { get; set; }
    public Speciality? Speciality { get; set; }

    public List<Student>? Students { get; set; }
    public List<Course>? Courses { get; set; }
}
```

3.3 Héritage

```
public class User
{
    [Key] public int Id { get; set; }
    [Required, MaxLength(50)] public string? Firstname { get; set; }
    [Required, MaxLength(50)] public string? Lastname { get; set; }
    [Required, MaxLength(100), RegularExpression(@"^\S+@\S+\.[a-z]+$")]
    public string? Email { get; set; }
    public string FullName => $"{Firstname} {Lastname}";
}

public class Student : User
{
    [Required] public DateOnly BirthDate { get; set; }
    // ...
}

public class Teacher : User
{
    public List<Course>? Courses { get; set; }
}
```

4. DTOs (Data Transfer Objects)

4.1 DTOs de Création

```
public class CreateClassDto
{
    [Required] public DateOnly? StartDate { get; set; }
    [Required] public int? Duration { get; set; }
    [Required] public int? YearId { get; set; }
    [Required] public int? SpecialityId { get; set; }
}
```

4.2 DTOs de Lecture

```
public class GetStudentDto
{
    public int Id { get; set; }
    public string? Firstname { get; set; }
    public string? Lastname { get; set; }
    public string FullName => $"{Firstname} {Lastname}";
    public string? Address { get; set; }
    public string? Email { get; set; }
    public DateOnly Birthdate { get; set; }
    public GetClassDto? Class { get; set; }
}
```

4.3 DTOs Légers

```
public class GetLightSchoolDto
{
    public int Id { get; set; }
    public string? Name { get; set; }
}
```

5. Controllers et API REST

5.1 Structure de Base

```
[ApiController]
[Route("API/[controller]")]
public class StudentsController(
    ApplicationContext context,
    IMapper mapper) : Controller
{
    // ...
}
```

5.2 Méthodes CRUD

```
[HttpPost]
public async Task<IActionResult> CreateStudent(CreateStudentDto dto)
{
    Student student = new();
    mapper.Map(dto, student);
    context.Students.Add(student);
    await context.SaveChangesAsync();
    return Created();
}

[HttpGet]
public async Task<IActionResult> GetStudents()
{
    List<GetStudentDto> students = await _getQuery.ToListAsync();
    return Ok(students);
}
```

6. Validation et Contraintes

6.1 Validation des Modèles

```
public class CreateSchoolDto
{
    [Required, MaxLength(50)] public string? Name { get; set; }
    [Required, MaxLength(150)] public string? Address { get; set; }
    [MaxLength(150)] public string? AddressSecondLine { get; set; }
    [Required, MaxLength(5), RegularExpression("[0-9]{5}")]
    public string? ZipCode { get; set; }
    [Required, MaxLength(14), RegularExpression("^([0-9]{2} ){4}[0-9]{2}$")]
    public string? PhoneNumber { get; set; }
    [Required, MaxLength(100), RegularExpression(@"^\S+@\S+\.[a-z]+$")]
    public string? Email { get; set; }
}
```

7. Gestion des Erreurs

7.1 Retours HTTP

```
if (student is null) return NotFound();
if (invalidInput) return BadRequest();
return Ok(result);
```

8. Patterns et Bonnes Pratiques

8.1 Injection de Dépendances

```
public class StudentsController(
    ApplicationContext context,
    IMapper mapper) : Controller
{
    // ...
}
```

8.2 Requêtes LINQ

```
private readonly IQueryable<GetStudentDto> _getQuery = context.Students
    .Include(x => x.Class)
    .Select(x => new GetStudentDto
    {
        Id = x.Id,
        Firstname = x.Firstname,
        Lastname = x.Lastname,
        // ...
    })
    .OrderBy(x => x.Lastname)
    .ThenBy(x => x.Firstname);
```

8.3 Mapping avec AutoMapper

```
// Configuration
public class MappingProfile : Profile
{
    public MappingProfile()
    {
        CreateMap<Student, GetStudentDto>();
        CreateMap<CreateStudentDto, Student>();
    }
}

// Utilisation
mapper.Map<GetStudentDto>(student);
```

9. Types et Structures de Données

9.1 Types Personnalisés

```
public DateOnly StartDate { get; set; }
public TimeOnly StartTime { get; set; }
```

9.2 Collections

```
public List<Student>? Students { get; set; }
public List<Course>? Courses { get; set; }
public List<Teacher>? Teachers { get; set; }
```

10. Configuration de l'Application

10.1 appsettings.json

```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Database=MonProjet;Trusted_Connection=True;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```

10.2 Program.cs

```csharp
var builder = WebApplication.CreateBuilder(args);

// Services
builder.Services.AddControllers();
builder.Services.AddDbContext<ApplicationContext>();
builder.Services.AddAutoMapper(typeof(Program));
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Middleware
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```