Plateforme Cagnotte Participative - Architecture MVC & Web API

Architecture du Projet

```
CagnotteApp/
│
├── CagnotteApp.Domain/          # Couche Domaine
│   ├── Entities/
│   │   ├── Entreprise.cs
│   │   ├── Cagnotte.cs
│   │   ├── Participant.cs
│   │   ├── Participation.cs
│   │   └── TypeCagnotte.cs
│   └── Enums/
│
├── CagnotteApp.Data/            # Couche Accès aux Données
│   ├── Context/
│   │   └── CagnotteContext.cs
│   ├── Configurations/
│   │   └── ConventionConfig.cs
│   └── Migrations/
│
├── CagnotteApp.Service/         # Couche Service (Logique Métier)
│   ├── Interfaces/
│   │   ├── ICagnotteService.cs
│   │   ├── IEntrepriseService.cs
│   │   └── IParticipantService.cs
│   └── Implementations/
│       ├── CagnotteService.cs
│       ├── EntrepriseService.cs
│       └── ParticipantService.cs
│
├── CagnotteApp.Web/             # Application MVC
│   ├── Controllers/
│   │   ├── CagnotteController.cs
│   │   └── EntrepriseController.cs
│   ├── Views/
│   │   ├── Cagnotte/
│   │   │   ├── Index.cshtml
│   │   │   ├── Create.cshtml
│   │   │   └── Details.cshtml
│   │   └── Entreprise/
│   │       └── Details.cshtml
│   └── ViewModels/
│       └── CagnotteViewModel.cs
│
└── CagnotteApp.API/              # Web API
    ├── Controllers/
    │   ├── CagnottesApiController.cs
    │   ├── EntreprisesApiController.cs
    │   └── ParticipantsApiController.cs
    └── DTOs/
        ├── CagnotteDto.cs
        ├── CreateCagnotteDto.cs
        └── EntrepriseDto.cs
```

Partie I: Entités & Entity Framework

1. Création des Entités

```csharp
// CagnotteApp.Domain/Entities/Entreprise.cs
public class Entreprise
{
    public int Id { get; set; }

    [Required]
    [StringLength(100)]
    public string NomEntreprise { get; set; }

    [Required]
    public string MailEntreprise { get; set; }

    // Navigation Properties
    public virtual ICollection<Cagnotte> Cagnottes { get; set; }

    public Entreprise()
    {
        Cagnottes = new HashSet<Cagnotte>();
    }
}
```

```csharp
```

```csharp
// CagnotteApp.Domain/Entities/TypeCagnotte.cs
public enum TypeCagnotte
{
    Evenement,
    Depense,
    ProjetSolidaire
}
```

csharp

```csharp
// CagnotteApp.Domain/Entities/Cagnotte.cs
public class Cagnotte
{
    public int Id { get; set; }

    [Required(ErrorMessage = "Le titre est obligatoire")]
    [StringLength(200)]
    public string Titre { get; set; }

    [DataType(DataType.MultilineText)]
    public string Description { get; set; }

    [Range(1, double.MaxValue, ErrorMessage = "La somme demandée doit être positive")]
    public decimal SommeDemandee { get; set; }

    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime DateLimite { get; set; }

    public TypeCagnotte Type { get; set; }

    // Foreign Keys
    public int EntrepriseId { get; set; }

    // Navigation Properties
    [ForeignKey("EntrepriseId")]
    public virtual Entreprise Entreprise { get; set; }

    public virtual ICollection<Participation> Participations { get; set; }

    public Cagnotte()
    {
        Participations = new HashSet<Participation>();
    }
}
```

csharp

```csharp
// CagnotteApp.Domain/Entities/Participant.cs
public class Participant
{
    public int Id { get; set; }

    [Required]
    [StringLength(100)]
    public string Nom { get; set; }

    [Required]
    [StringLength(100)]
    public string Prenom { get; set; }

    [Required]
    public string MailParticipant { get; set; }

    // Navigation Properties
    public virtual ICollection<Participation> Participations { get; set; }

    public Participant()
    {
        Participations = new HashSet<Participation>();
    }
}
```

csharp

```
// CagnotteApp.Domain/Entities/Participation.cs
public class Participation
{
    // Clé primaire composée
    [Key, Column(Order = 0)]
    public int CagnotteId { get; set; }

    [Key, Column(Order = 1)]
    public int ParticipantId { get; set; }

    public decimal Montant { get; set; }

    public DateTime DateParticipation { get; set; }

    // Navigation Properties
    [ForeignKey("CagnotteId")]
    public virtual Cagnotte Cagnotte { get; set; }

    [ForeignKey("ParticipantId")]
    public virtual Participant Participant { get; set; }
}
```

2. DbContext avec Convention Personnalisée

```
csharp
```

```csharp
// CagnotteApp.Data/Context/CagnotteContext.cs
public class CagnotteContext : DbContext
{
    public CagnotteContext(DbContextOptions<CagnotteContext> options)
        : base(options)
    {
    }

    public DbSet<Entreprise> Entreprises { get; set; }
    public DbSet<Cagnotte> Cagnottes { get; set; }
    public DbSet<Participant> Participants { get; set; }
    public DbSet<Participation> Participations { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Configuration de la clé primaire composée
        modelBuilder.Entity<Participation>()
            .HasKey(p => new { p.CagnotteId, p.ParticipantId });

        // Convention personnalisée pour les propriétés commençant par "Mail"
        foreach (var entityType in modelBuilder.Model.GetEntityTypes())
        {
            foreach (var property in entityType.GetProperties())
            {
                if (property.Name.StartsWith("Mail"))
                {
                    property.IsNullable = false;
                }
            }
        }

        // Configuration des relations
        modelBuilder.Entity<Participation>()
            .HasOne(p => p.Cagnotte)
            .WithMany(c => c.Participations)
            .HasForeignKey(p => p.CagnotteId);

        modelBuilder.Entity<Participation>()
            .HasOne(p => p.Participant)
            .WithMany(p => p.Participations)
            .HasForeignKey(p => p.ParticipantId);
    }
}
```

Partie II: Couche Service

Interface des Services

csharp

```csharp
// CagnotteApp.Service/Interfaces/ICagnotteService.cs
public interface ICagnotteService
{
    Task<IEnumerable<Cagnotte>> GetCagnottesEnCoursAsync();
    Task<decimal> CalculerMontantCollecteAsync(int cagnotteId);
    Task<Entreprise> GetEntrepriseAvecPlusDeParticipantsAsync();
    Task<IEnumerable<Cagnotte>> GetAllAsync();
    Task<Cagnotte> GetByIdAsync(int id);
    Task<Cagnotte> CreateAsync(Cagnotte cagnotte);
    Task<IEnumerable<Cagnotte>> SearchByTitreAsync(string titre);
}

public interface IParticipantService
{
    Task<int> GetNombreCagnottesParticipantAsync(int participantId);
    Task<IEnumerable<Participant>> GetAllAsync();
}

public interface IEntrepriseService
{
    Task<IEnumerable<Entreprise>> GetTop2EntreprisesByTypeAsync(TypeCagnotte type);
    Task<IEnumerable<Entreprise>> GetAllAsync();
    Task<Entreprise> GetByIdAsync(int id);
}
```

Implémentation des Services

csharp

```
// CagnotteApp.Service/Implementations/CagnotteService.cs
public class CagnotteService : ICagnotteService
{
    private readonly CagnotteContext _context;

    public CagnotteService(CagnotteContext context)
    {
        _context = context;
    }

    // 1. Retourner les cagnottes en cours
    public async Task<IEnumerable<Cagnotte>> GetCagnottesEnCoursAsync()
    {
        return await _context.Cagnottes
            .Include(c => c.Entreprise)
            .Where(c => c.DateLimite >= DateTime.Now)
            .ToListAsync();
    }

    // 2. Calculer le montant collecté pour une cagnotte
    public async Task<decimal> CalculerMontantCollecteAsync(int cagnotteId)
    {
        var montant = await _context.Participations
            .Where(p => p.CagnotteId == cagnotteId)
            .SumAsync(p => p.Montant);

        return montant;
    }

    // 5. Retourner l'entreprise avec la cagnotte ayant le plus de participants
    public async Task<Entreprise> GetEntrepriseAvecPlusDeParticipantsAsync()
    {
        var entreprise = await _context.Cagnottes
            .Include(c => c.Entreprise)
            .Include(c => c.Participations)
            .OrderByDescending(c => c.Participations.Count)
            .Select(c => c.Entreprise)
            .FirstOrDefaultAsync();

        return entreprise;
    }

    public async Task<IEnumerable<Cagnotte>> GetAllAsync()
    {
        return await _context.Cagnottes
            .Include(c => c.Entreprise)
            .ToListAsync();
    }

    public async Task<Cagnotte> GetByIdAsync(int id)
    {
        return await _context.Cagnottes
            .Include(c => c.Entreprise)
            .Include(c => c.Participations)
            .FirstOrDefaultAsync(c => c.Id == id);
    }

    public async Task<Cagnotte> CreateAsync(Cagnotte cagnotte)
    {
        _context.Cagnottes.Add(cagnotte);
        await _context.SaveChangesAsync();
        return cagnotte;
    }

    public async Task<IEnumerable<Cagnotte>> SearchByTitreAsync(string titre)
    {
        if (string.IsNullOrEmpty(titre))
            return await GetAllAsync();

        return await _context.Cagnottes
            .Include(c => c.Entreprise)
            .Where(c => c.Titre.Contains(titre))
            .ToListAsync();
    }
}
```

csharp

```
// CagnotteApp.Service/Implementations/ParticipantService.cs
public class ParticipantService : IParticipantService
{
    private readonly CagnotteContext _context;

    public ParticipantService(CagnotteContext context)
    {
        _context = context;
    }

    // 3. Calculer le nombre de cagnottes pour un participant
    public async Task<int> GetNombreCagnottesParticipantAsync(int participantId)
    {
        var nombre = await _context.Participations
            .Where(p => p.ParticipantId == participantId)
            .CountAsync();

        return nombre;
    }

    public async Task<IEnumerable<Participant>> GetAllAsync()
    {
        return await _context.Participants.ToListAsync();
    }
}
```

csharp

```
// CagnotteApp.Service/Implementations/EntrepriseService.cs
public class EntrepriseService : IEntrepriseService
{
    private readonly CagnotteContext _context;

    public EntrepriseService(CagnotteContext context)
    {
        _context = context;
    }

    // 4. Top 2 entreprises par type de cagnotte
    public async Task<IEnumerable<Entreprise>> GetTop2EntreprisesByTypeAsync(TypeCagnotte type)
    {
        var entreprises = await _context.Entreprises
            .Include(e => e.Cagnottes)
            .Select(e => new
            {
                Entreprise = e,
                NombreCagnottes = e.Cagnottes.Count(c => c.Type == type)
            })
            .OrderByDescending(x => x.NombreCagnottes)
            .Take(2)
            .Select(x => x.Entreprise)
            .ToListAsync();

        return entreprises;
    }

    public async Task<IEnumerable<Entreprise>> GetAllAsync()
    {
        return await _context.Entreprises.ToListAsync();
    }

    public async Task<Entreprise> GetByIdAsync(int id)
    {
        return await _context.Entreprises
            .Include(e => e.Cagnottes)
            .FirstOrDefaultAsync(e => e.Id == id);
    }
}
```

Partie III: Application MVC

Configuration (Program.cs)

```csharp
// CagnotteApp.Web/Program.cs
var builder = WebApplication.CreateBuilder(args);

// Services MVC
builder.Services.AddControllersWithViews();

// DbContext
builder.Services.AddDbContext<CagnotteContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Enregistrement des services
builder.Services.AddScoped<ICagnotteService, CagnotteService>();
builder.Services.AddScoped<IEntrepriseService, EntrepriseService>();
builder.Services.AddScoped<IParticipantService, ParticipantService>();

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Cagnotte}/{action=Index}/{id?}");

app.Run();
```

ViewModel

```csharp
```

```
// CagnotteApp.Web/ViewModels/CagnotteViewModel.cs
public class CagnotteViewModel
{
    public int Id { get; set; }

    [Required(ErrorMessage = "Le titre est obligatoire")]
    public string Titre { get; set; }

    [DataType(DataType.MultilineText)]
    public string Description { get; set; }

    [Required]
    [Range(1, double.MaxValue, ErrorMessage = "La somme doit être positive")]
    [Display(Name = "Somme Demandée")]
    public decimal SommeDemandee { get; set; }

    [Required]
    [DataType(DataType.Date)]
    [Display(Name = "Date Limite")]
    public DateTime DateLimite { get; set; }

    [Required]
    [Display(Name = "Type")]
    public TypeCagnotte Type { get; set; }

    [Required]
    [Display(Name = "Entreprise")]
    public int EntrepriseId { get; set; }

    // Pour les listes déroulantes
    public SelectList Entreprises { get; set; }
    public SelectList Types { get; set; }
}
```

Contrôleur MVC

```
csharp
```

```
// CagnotteApp.Web/Controllers/CagnotteController.cs
public class CagnotteController : Controller
{
    private readonly ICagnotteService _cagnotteService;
    private readonly IEntrepriseService _entrepriseService;

    public CagnotteController(ICagnotteService cagnotteService,
                             IEntrepriseService entrepriseService)
    {
        _cagnotteService = cagnotteService;
        _entrepriseService = entrepriseService;
    }

    // GET: Cagnotte/Index
    public async Task<IActionResult> Index(string searchTitre)
    {
        IEnumerable<Cagnotte> cagnottes;

        if (!string.IsNullOrEmpty(searchTitre))
        {
            cagnottes = await _cagnotteService.SearchByTitreAsync(searchTitre);
            ViewBag.SearchTitre = searchTitre;
        }
        else
        {
            cagnottes = await _cagnotteService.GetAllAsync();
        }

        return View(cagnottes);
    }

    // GET: Cagnotte/Create
    public async Task<IActionResult> Create()
    {
        var entreprises = await _entrepriseService.GetAllAsync();

        var viewModel = new CagnotteViewModel
        {
            Entreprises = new SelectList(entreprises, "Id", "NomEntreprise"),
            Types = new SelectList(Enum.GetValues(typeof(TypeCagnotte)))
        };

        return View(viewModel);
    }

    // POST: Cagnotte/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create(CagnotteViewModel viewModel)
    {
        if (ModelState.IsValid)
        {
            var cagnotte = new Cagnotte
            {
                Titre = viewModel.Titre,
                Description = viewModel.Description,
                SommeDemandee = viewModel.SommeDemandee,
                DateLimite = viewModel.DateLimite,
                Type = viewModel.Type,
                EntrepriseId = viewModel.EntrepriseId
            };

            await _cagnotteService.CreateAsync(cagnotte);
            return RedirectToAction(nameof(Index));
        }

        // Recharger les listes si erreur
        var entreprises = await _entrepriseService.GetAllAsync();
        viewModel.Entreprises = new SelectList(entreprises, "Id", "NomEntreprise");
        viewModel.Types = new SelectList(Enum.GetValues(typeof(TypeCagnotte)));

        return View(viewModel);
    }

    // GET: Cagnotte/Details/5
    public async Task<IActionResult> Details(int id)
    {
        var cagnotte = await _cagnotteService.GetByIdAsync(id);
        if (cagnotte == null)
            return NotFound();

        return View(cagnotte);
    }
}
```

```
csharp
```

```csharp
// CagnotteApp.Web/Controllers/EntrepriseController.cs
public class EntrepriseController : Controller
{
    private readonly IEntrepriseService _entrepriseService;

    public EntrepriseController(IEntrepriseService entrepriseService)
    {
        _entrepriseService = entrepriseService;
    }

    // GET: Entreprise/Details/5
    public async Task<IActionResult> Details(int id)
    {
        var entreprise = await _entrepriseService.GetByIdAsync(id);
        if (entreprise == null)
            return NotFound();

        return View(entreprise);
    }
}
```

Vues Razor

```
html
```

```
<!-- Views/Cagnotte/Create.cshtml -->
@model CagnotteViewModel

@{
    ViewData["Title"] = "Créer une Cagnotte";
}

<h2>@ViewData["Title"]</h2>

<div class="row">
    <div class="col-md-6">
        <form asp-action="Create" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <div class="form-group">
                <label asp-for="Titre" class="control-label"></label>
                <input asp-for="Titre" class="form-control" />
                <span asp-validation-for="Titre" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <textarea asp-for="Description" class="form-control" rows="4"></textarea>
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="SommeDemandee" class="control-label"></label>
                <input asp-for="SommeDemandee" class="form-control" type="number" step="0.01" />
                <span asp-validation-for="SommeDemandee" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="DateLimite" class="control-label"></label>
                <input asp-for="DateLimite" class="form-control" type="date" />
                <span asp-validation-for="DateLimite" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="EntrepriseId" class="control-label"></label>
                <select asp-for="EntrepriseId" asp-items="Model.Entreprises" class="form-control">
                    <option value="">-- Sélectionnez une entreprise --</option>
                </select>
                <span asp-validation-for="EntrepriseId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Type" class="control-label"></label>
                <select asp-for="Type" asp-items="Model.Types" class="form-control">
                    <option value="">-- Sélectionnez un type --</option>
                </select>
                <span asp-validation-for="Type" class="text-danger"></span>
            </div>

            <div class="form-group mt-3">
                <input type="submit" value="Créer" class="btn btn-primary" />
                <a asp-action="Index" class="btn btn-secondary">Retour à la liste</a>
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

```
html
```

```
<!-- Views/Cagnotte/Index.cshtml -->
@model IEnumerable<Cagnotte>

@{
    ViewData["Title"] = "Liste des Cagnottes";
}

<h2>@ViewData["Title"]</h2>

<p>
    <a asp-action="Create" class="btn btn-primary">Créer une nouvelle cagnotte</a>
</p>

<!-- Bloc de recherche -->
<div class="row mb-3">
    <div class="col-md-6">
        <form asp-action="Index" method="get">
            <div class="input-group">
                <input type="text" name="searchTitre"
                       value="@ViewBag.SearchTitre"
                       class="form-control"
                       placeholder="Rechercher par titre..." />
                <button type="submit" class="btn btn-outline-secondary">
                    <i class="bi bi-search"></i> Rechercher
                </button>
                <a asp-action="Index" class="btn btn-outline-secondary">Effacer</a>
            </div>
        </form>
    </div>
</div>

<table class="table table-striped">
    <thead>
        <tr>
            <th>Titre</th>
            <th>Description</th>
            <th>Somme Demandée</th>
            <th>Date Limite</th>
            <th>Type</th>
            <th>Entreprise</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var cagnotte in Model)
        {
            <tr>
                <td>@cagnotte.Titre</td>
                <td>@cagnotte.Description</td>
                <td>@cagnotte.SommeDemandee.ToString("C")</td>
                <td>@cagnotte.DateLimite.ToString("dd/MM/yyyy")</td>
                <td>@cagnotte.Type</td>
                <td>
                    <a asp-controller="Entreprise"
                       asp-action="Details"
                       asp-route-id="@cagnotte.EntrepriseId">
                        @cagnotte.Entreprise.NomEntreprise
                    </a>
                </td>
                <td>
                    <a asp-action="Details" asp-route-id="@cagnotte.Id" class="btn btn-sm btn-info">Détails</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

```html
```

```
<!-- Views/Entreprise/Details.cshtml -->
@model Entreprise

@{
    ViewData["Title"] = "Détails de l'Entreprise";
}

<h2>@ViewData["Title"]</h2>

<div class="card">
    <div class="card-header">
        <h4>@Model.NomEntreprise</h4>
    </div>
    <div class="card-body">
        <dl class="row">
            <dt class="col-sm-3">Nom de l'entreprise</dt>
            <dd class="col-sm-9">@Model.NomEntreprise</dd>

            <dt class="col-sm-3">Email</dt>
            <dd class="col-sm-9">@Model.MailEntreprise</dd>

            <dt class="col-sm-3">Nombre de cagnottes</dt>
            <dd class="col-sm-9">@Model.Cagnottes.Count</dd>
        </dl>

        <h5 class="mt-4">Cagnottes créées</h5>
        @if (Model.Cagnottes.Any())
        {
            <ul class="list-group">
                @foreach (var cagnotte in Model.Cagnottes)
                {
                    <li class="list-group-item">
                        <a asp-controller="Cagnotte"
                           asp-action="Details"
                           asp-route-id="@cagnotte.Id">
                            @cagnotte.Titre
                        </a>
                        - @cagnotte.SommeDemandee.ToString("C")
                    </li>
                }
            </ul>
        }
        else
        {
            <p>Aucune cagnotte créée</p>
        }
    </div>
    <div class="card-footer">
        <a asp-controller="Cagnotte" asp-action="Index" class="btn btn-secondary">Retour à la liste</a>
    </div>
</div>
```

Partie IV: Web API

Configuration API (Program.cs)

```
csharp
```

```csharp
// CagnotteApp.API/Program.cs
var builder = WebApplication.CreateBuilder(args);

// Services API
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

// CORS
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll",
        builder => builder
            .AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader());
});

// DbContext
builder.Services.AddDbContext<CagnotteContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Services
builder.Services.AddScoped<ICagnotteService, CagnotteService>();
builder.Services.AddScoped<IEntrepriseService, EntrepriseService>();
builder.Services.AddScoped<IParticipantService, ParticipantService>();

var app = builder.Build();

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

app.UseHttpsRedirection();
app.UseCors("AllowAll");
app.UseAuthorization();
app.MapControllers();

app.Run();
```

DTOs

```
csharp
```

```csharp
// CagnotteApp.API/DTOs/CagnotteDto.cs
public class CagnotteDto
{
    public int Id { get; set; }
    public string Titre { get; set; }
    public string Description { get; set; }
    public decimal SommeDemandee { get; set; }
    public decimal MontantCollecte { get; set; }
    public DateTime DateLimite { get; set; }
    public string Type { get; set; }
    public string NomEntreprise { get; set; }
    public int NombreParticipants { get; set; }
}

public class CreateCagnotteDto
{
    [Required]
    public string Titre { get; set; }

    public string Description { get; set; }

    [Required]
    [Range(1, double.MaxValue)]
    public decimal SommeDemandee { get; set; }

    [Required]
    public DateTime DateLimite { get; set; }

    [Required]
    public TypeCagnotte Type { get; set; }

    [Required]
    public int EntrepriseId { get; set; }
}

public class ParticipationDto
{
    [Required]
    public int CagnotteId { get; set; }

    [Required]
    public int ParticipantId { get; set; }

    [Required]
    [Range(1, double.MaxValue)]
```