

RAPPORT DE PROJET

AP2 BTS SIO



Application web
à utilité des
Médecins

Projet réalisé dans le cadre de la formation
BTS SIO SLAM

SOMMAIRE

1_ Contexte	p.3
2_ Besoins et Objectifs	p.6
3_ Analyse fonctionnelle	p.7
4_ Choix techniques	p.9
5_ La Base de Données	p.10
6_ Organisation du code	p.15
7_ Explication détaillée	p.16
- Le code de mon application médicale en ASP.NET Core MVC (<i>p.15</i>)	
- La méthode de contact de l'application (<i>p.19</i>)	
-	
8_ Annexe	p.34

CONTEXTE

→ Création d'une Application de Gestion des Ordonnances Médicales pour l'entreprise *MedManager*

La gestion des dossiers médicaux et des prescriptions est une tâche essentielle mais complexe pour les professionnels de santé. Dans un contexte où les médecins doivent jongler entre de nombreuses consultations, des suivis de patients variés et une montagne de données médicales, la nécessité d'un outil numérique fiable, performant et intuitif est devenue incontournable. C'est pour répondre à ces enjeux que l'entreprise *MedManager* a commandé un projet ambitieux : une application de gestion des ordonnances médicales pensée pour optimiser le quotidien des médecins.

Le projet vise à centraliser toutes les informations relatives aux patients et à leurs prescriptions dans une interface unique, sécurisée et facile d'utilisation. En intégrant des fonctionnalités clés comme l'historique médical, la gestion des allergies, l'analyse des interactions médicamenteuses et la création d'ordonnances, *MedManager* se positionne comme une solution moderne répondant aux besoins croissants des cabinets médicaux.

Les défis rencontrés par les médecins

→ Complexité administrative : La gestion des dossiers patients et des prescriptions nécessite un suivi rigoureux, souvent chronophage.

→ Prévention des risques : Les allergies ou les interactions médicamenteuses représentent un danger important, demandant une vigilance accrue lors des prescriptions.

→ Traçabilité et sécurité : Le stockage et le traitement des données sensibles exigent des mesures strictes pour respecter la confidentialité et la réglementation en vigueur.

→ Amélioration de la qualité des soins : Les médecins cherchent des outils pour optimiser leurs décisions cliniques tout en gagnant du temps.

Objectifs du projet

L'application *MedManager* a pour mission de fournir une solution complète et performante pour relever ces défis. Elle s'inscrit dans une volonté d'améliorer la qualité des soins tout en simplifiant la charge de travail des praticiens. Avec cette application, les médecins pourront non seulement mieux gérer leurs patients, mais aussi réduire les erreurs médicales grâce à une automatisation intelligente des alertes et des analyses de risques.

En capitalisant sur les technologies modernes telles que C# et ASP.NET Core MVC, l'application garantit une robustesse technique, une évolutivité, et une navigation fluide entre les différentes fonctionnalités. Elle se distingue par son approche centrée sur l'utilisateur, offrant une interface claire et accessible même aux praticiens peu familiers avec les outils numériques.

Ce projet répond à des enjeux réels du domaine médical en mettant l'accent sur la simplicité, la sécurité et l'efficacité. Il traduit une vision ambitieuse d'une médecine plus connectée et plus sûre, au service des patients comme des professionnels.

EXPRESSION DES BESOINS & OBJECTIFS

L'entreprise MedManager a identifié des besoins cruciaux dans le domaine médical, nécessitant une solution numérique performante pour répondre aux attentes des professionnels de santé. Voici les principaux besoins :

→ Gestion centralisée des données médicales :

Fournir une plateforme unique pour centraliser les informations des patients, leurs antécédents médicaux, leurs allergies, et leurs prescriptions. Cela permet de réduire les erreurs liées à la dispersion des données et d'assurer une traçabilité complète.

→ Sécurisation des informations sensibles :

Garantir la confidentialité et l'intégrité des données médicales conformément aux réglementations en vigueur (comme le RGPD). Une authentification robuste et un contrôle des accès sont nécessaires pour protéger les données.

→ Prévention des risques médicaux :

Offrir un système d'alerte capable de détecter en temps réel les interactions médicamenteuses dangereuses ou les contre-indications liées aux allergies ou aux antécédents des patients.

→ Gain de temps pour les médecins :

Simplifier les tâches administratives quotidiennes, comme la gestion des ordonnances, la recherche d'informations sur les médicaments ou la consultation des dossiers patients. Une interface intuitive réduit le temps passé sur ces tâches et permet aux médecins de se concentrer sur leurs consultations.

→ Automatisation et amélioration des soins :

Permettre aux médecins de générer des ordonnances précises et complètes en quelques clics grâce à des formulaires intelligents. Les rapports générés sur les prescriptions ou les médicaments les plus utilisés aident aussi à optimiser les pratiques médicales.

→ Accès à des statistiques et rapports :

Aider les médecins et les gestionnaires à prendre des décisions éclairées grâce à des statistiques sur les prescriptions, les médicaments les plus prescrits, ou les patients les plus suivis.

ANALYSE FONCTIONNELLE

Analyse fonctionnelle et choix technologiques

La plateforme est une application de gestion des ordonnances médicales, ainsi, chaque médecin doit pouvoir effectuer différentes tâches à travers cette plateforme. Cette application peut être pensée en deux parties distinctes, celle qui est utilisée par les médecins et celle qui permet le bon fonctionnement des interfaces. La deuxième partie n'est pas utile aux médecins, c'est un centre de contrôle des paramètres de l'application.

I- Partie de l'application utilisée par les médecins

-Authentification et gestion des utilisateurs

- Page de connexion du médecin
- Page de gestion du profil du médecin

-Tableau de bord du médecin

- Vue d'ensemble des patients récents
- Ordonnances en cours de chaque patient

-Base de données des médicaments

- Avoir accès à la liste des médicaments et pouvoir chercher dans cette même liste à l'aide d'une barre de recherche
- Chaque médicament possède une fiche détaillée avec les informations sur les contre-indications et les interactions
- Possibilité d'ajouter des médicaments à la liste

-Gestion des patients

- Liste des patients avec la possibilité de rechercher dans cette liste
- Chaque patient possède une fiche détaillée qui comprends une section pour les antécédents médicaux et une autre pour les allergies
- Pouvoir ajouter des patients
- Pouvoir modifier les informations de chaque patient

-Gestion des allergies et antécédents

- Interface permettant l'ajout ou la modification d'allergies et d'antécédents médicaux

-Création des ordonnances

- Interface de sélection des médicaments
- Pouvoir saisir de la posologie et de la durée du traitement
- Possibilité d'ajouter des instructions spéciales
- Mise en place d'un système d'alerte pour les interactions médicamenteuses et contre-

indications

-Gestion des ordonnances

- **Avoir accès à la liste des ordonnances**
- Possibilité de créer une nouvelle ordonnance
- Possibilité de modifier une ordonnance existante
- Visualisation détaillée d'une ordonnance
- Pouvoir utiliser la fonction d'export d'ordonnance en format PDF

II- Partie de l'application liée aux paramètres généraux

-Rapports et statistiques

- Rapport sur les prescriptions
- Statistiques sur les médicaments les plus prescrits
- Liste d'ordonnances en cours

-Paramètres de l'application

- Configuration des alertes
- Gestion des droits d'accès

-Aide et support

- Pouvoir accéder à un guide d'utilisation ainsi qu'à une page de FAQ
- Formulaire de contact pour le support technique

L'objectif est donc de regrouper tous les besoins dans une seule plateforme afin de permettre aux médecins de gérer et traiter les dossiers médicaux efficacement. Cette application est alors facile d'utilisation pour les médecins, ce qui rend efficace les manipulations.

De plus, la sécurité des données des patients est assurée.

CHOIX TECHNOLOGIQUES

Analyse fonctionnelle et choix technologiques

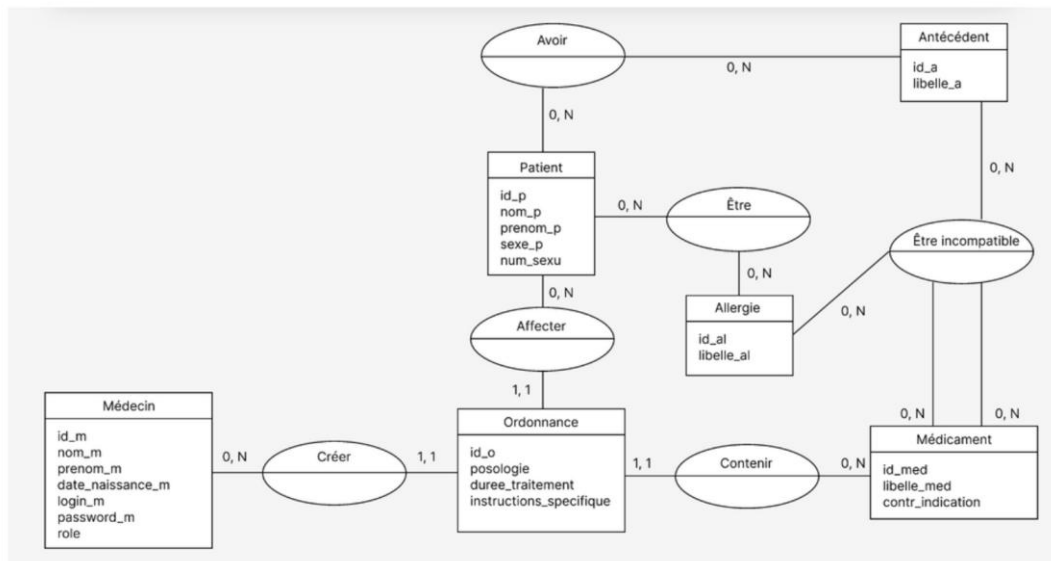
Choix des technologies utilisées

1. Langages de programmation :
 - C# : Utilisé pour sa robustesse, sa compatibilité avec .NET, et son support des architectures MVC.
 - HTML/CSS/JavaScript : Pour la conception des vues et une interface utilisateur moderne.
2. Framework :
 - ASP.NET Core MVC : Offre une architecture modulaire et performante pour la gestion des modèles, des vues, et des contrôleurs. Il garantit une évolutivité et une maintenance aisée du projet.
3. Base de données :
 - PHPMyAdmin: Permet de gérer efficacement les relations complexes entre les patients, les ordonnances, et les médicaments grâce à sa puissance et sa fiabilité.
4. Technologies front-end :
 - Bootstrap : Facilite la création d'interfaces responsives, intuitives et esthétiques, optimisées pour tous types d'écrans.
5. Export PDF :
 - iTextSharp : Utilisé pour générer des ordonnances téléchargeables en format PDF.
6. Sécurité :
 - Identity Framework : Fournit une authentification sécurisée pour protéger les données des utilisateurs.
7. Système d'alerte :
 - Algorithmes d'analyse intégrées au back-end pour détecter les risques liés aux allergies ou interactions médicamenteuses.

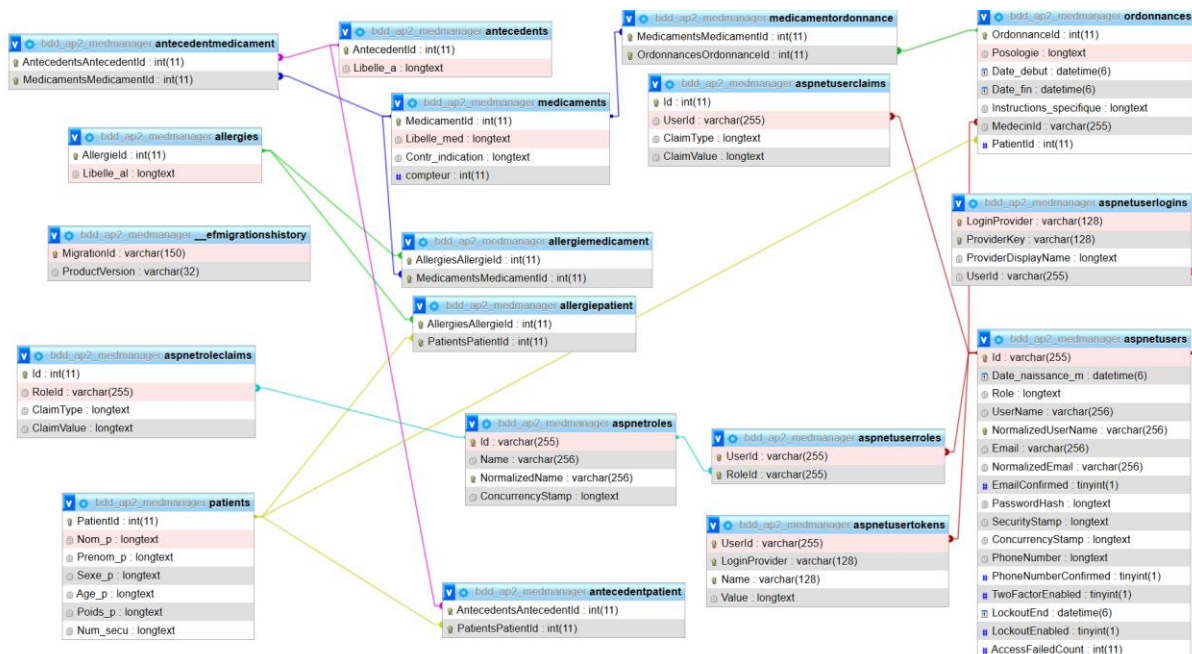
Le choix de ces technologies permet de répondre aux besoins exprimés par l'entreprise tout en garantissant une application performante, évolutive, et conforme aux standards du secteur médical.

LA BASE DE DONNÉES

Aspect visuel



MCD de notre application



Tables et Relations

Création de la Base de Données avec DbContext dans une Application ASP.NET

I- Définir les Modèles:

Les modèles sont des classes qui représentent les tables dans la base de données. Chaque propriété d'un modèle correspond à une ligne de la table associée.

Exemple de Modèle : Patient.cs

```
public class Patient
{
    [Key]
    0 references
    public int PatientId { get; set; }

    [Required(ErrorMessage = "Le nom est requis")]
    0 references
    public required string Nom_p { get; set; }

    [Required(ErrorMessage = "Le prénom est requis")]
    0 references
    public required string Prenom_p { get; set; }

    [Required(ErrorMessage = "Le sexe est requis")]
    0 references
    public required string Sexe_p { get; set; }

    [Required(ErrorMessage = "L'âge est requis")]
    [RegularExpression(@"^\d+$", ErrorMessage = "L'âge doit être un nombre entier")]
    0 references
    public required string Age_p { get; set; }

    [Required(ErrorMessage = "Le poids est requis")]
    [RegularExpression(@"^\d+(\.\d{1,2})?$", ErrorMessage = "Le poids doit être un nombre valide (ex: 70 ou 70.5)")]
    0 references
    public required string Poids_p { get; set; }

    [Required(ErrorMessage = "Le Numéro de sécurité est requis")]
    [RegularExpression(@"^\d{7}$", ErrorMessage = "Le numéro de sécurité doit posséder 7 chiffres")]
    0 references
    public required string Num_secu { get; set; }

    0 references
    public List<Antecedent> Antecedents { get; set; } = new();
    0 references
    public List<Allergie> Allergies { get; set; } = new();
    0 references
    public List<Ordonnance> Ordonnances { get; set; } = new();
}
```

La première partie définit les informations de base du patient (ses propriétés) :

- PatientId : un identifiant unique en nombre entier (int)
- Nom_p : le nom du patient en texte (string)
- Prenom_p : le prénom en texte
- Sexe_p : le sexe en texte
- Num_secu : le numéro de sécurité sociale en texte

Chaque champ a des validations :

- [Required] : indique que le champ est obligatoire avec un message d'erreur en français
- [RegularExpression] : pour le numéro de sécurité, vérifie qu'il contient 7 chiffres

La dernière partie définit les relations avec d'autres tables :

- Une liste d'Antécédents
- Une liste d'Allergies
- Une liste d'Ordonnances

II- Créer le DbContext

Le DbContext est la classe qui gère les interactions entre les modèles et la base de données. Elle permet de réaliser des opérations comme la lecture, l'écriture et la mise à jour des données.

Exemple de DbContext :

```
2 references
public class ApplicationDbContext : IdentityDbContext<Medecin>
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options): base(options)
    {
    }

    0 references
    public DbSet<Patient> Patients => Set<Patient>();
    0 references
    public DbSet<Medecin> Medecins => Set<Medecin>();
    0 references
    public DbSet<Allergie> Allergies => Set<Allergie>();
    0 references
    public DbSet<Ordonnance> Ordonnances => Set<Ordonnance>();
    0 references
    public DbSet<Medicament> Medicaments => Set<Medicament>();
    0 references
    public DbSet<Antecedent> Antecedents => Set<Antecedent>();
}
```

→ Le DbContext est le chef d'orchestre de ma base de données dans une application ASP.NET. Il crée le lien entre mes modèles et ma base de données, en transformant mes classes C# en tables relationnelles. Il s'occupe de toutes les opérations avec la base de données et gère les relations entre les différentes tables grâce aux DbSet que je déclare pour chaque modèle.

→ Sur la deuxième capture on voit la méthode OnModelCreating qui permet de créer les relations entre les tables (les cardinalités) et leurs tables de jointures dans notre base de données.

0 references

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Patient>()
        .HasMany(p => p.Allergies)
        .WithMany(a => a.Patients)
        .UsingEntity(j => j.ToTable("AllergiePatient")); ;

    modelBuilder.Entity<Patient>()
        .HasMany(p => p.Antecedents)
        .WithMany(a => a.Patients)
        .UsingEntity(j => j.ToTable("AntecedentPatient")); ;

    modelBuilder.Entity<Medicament>()
        .HasMany(m => m.Antecedents)
        .WithMany(a => a.Medicaments)
        .UsingEntity(j => j.ToTable("AntecedentMedicament"));

    modelBuilder.Entity<Medicament>()
        .HasMany(m => m.Allergies)
        .WithMany(a => a.Medicaments)
        .UsingEntity(j => j.ToTable("AllergieMedicament"));

    modelBuilder.Entity<Ordonnance>()
        .HasMany(o => o.Medicaments)
        .WithMany(m => m.Ordonnances)
        .UsingEntity(j => j.ToTable("MedicamentOrdonnance"));

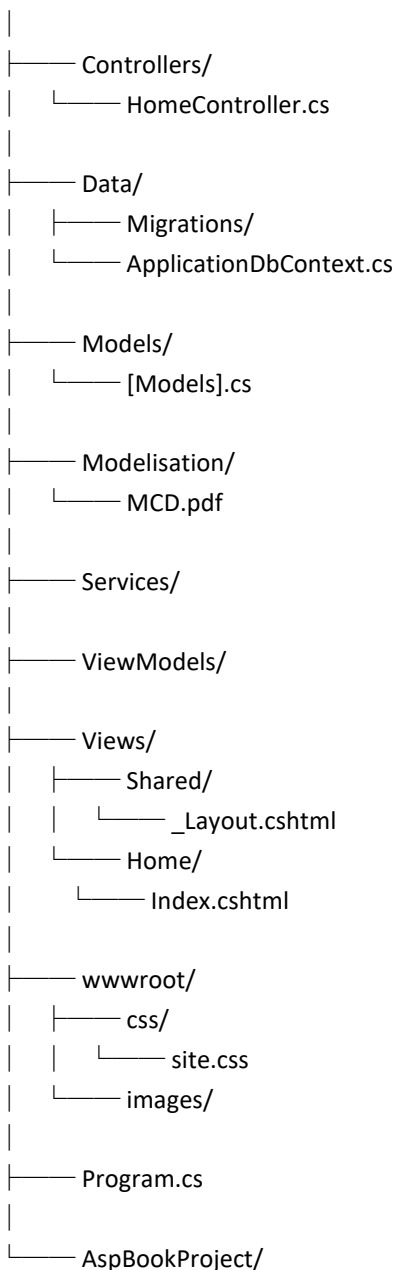
    modelBuilder.Entity<Ordonnance>()
        .HasOne(o => o.Patient)
        .WithMany(p => p.Ordonnances)
        .HasForeignKey(o => o.PatientId);

    modelBuilder.Entity<Ordonnance>()
        .HasOne(o => o.Medecin)
        .WithMany(m => m.Ordonnances)
        .HasForeignKey(o => o.MedecinId);

    base.OnModelCreating(modelBuilder);
}
```

ORGANISATION DU CODE

AP2/



Cette architecture suit le pattern MVC (Model-View-Controller) avec des couches supplémentaires pour une meilleure séparation des responsabilités :

- Services pour isoler la logique métier
- ViewModels pour adapter les données à l'affichage
- Data pour la gestion de la base de données

EXPLICATIONS DÉTAILLÉES

Le code de mon application médicale en ASP.NET Core MVC

Je vais vous expliquer en détail la partie de mon code qui gère l'ajout de médicaments dans l'application. Cette fonctionnalité permet aux médecins d'enregistrer de nouveaux médicaments dans la base de données...

Pour démarrer :

1. ViewModel : structure des données
2. Controller : logique de traitement
3. Vue : affichage et interaction utilisateur

```
public class OrdonnanceViewModel
{
    0 references
    public int? OrdonnanceId { get; set; }

    [StringLength(100)]
    [Required(ErrorMessage = "La posologie est requise")]
    0 references
    public string Posologie { get; set; }

    [DataType(DataType.Date)]
    [Required(ErrorMessage = "Le date de début est obligatoire")]
    0 references
    public DateTime Date_debut { get; set; }

    [DataType(DataType.Date)]
    [Required(ErrorMessage = "Le date de fin est obligatoire")]
    0 references
    public DateTime Date_fin { get; set; }

    0 references
    public string? Instructions_specifique { get; set; }

    [Required(ErrorMessage = "Veuillez sélectionner un patient")]
    0 references
    public int? PatientId { get; set; }

    1 reference
    public Patient? Patient { get; set; }

    0 references
    public Medecin? Medecin { get; set; }
    0 references
    public List<Patient> Patients { get; set; } = new List<Patient>();
    0 references
    public List<Medicament>? Medicaments { get; set; }
    0 references
    public List<int> SelectedMedicamentId { get; set; } = new List<int>();
}
```


1. Dans mon contrôleur, j'ai deux méthodes Add pour gérer l'ajout d'une ordonnance :

- La première méthode (HTTP GET) initialise le formulaire :

```
public async Task<IActionResult> Ajouter()
{
    string? MedecinId = _userManager.GetUserId(User);
    Medecin? med = _userManager.FindByIdAsync(MedecinId).Result;
    if (med == null)
    {
        return RedirectToAction("Logout", "Account");
    }
    var viewModel = new OrdonnanceViewModel
    {
        Date_debut = DateTime.Now,
        Date_fin = DateTime.Now.AddDays(1),
        Patients = await _dbContext.Patients.ToListAsync(),
        Medicaments = await _dbContext.Medicaments.ToListAsync(),
        SelectedMedicamentId = new List<int>(),
    };
    return View(viewModel);
}
```

La méthode Ajouter() prépare les données nécessaires à l'affichage de la vue d'ajout d'une ordonnance pour un médecin connecté. Elle vérifie l'authenticité de l'utilisateur, charge les listes de patients et médicaments depuis la base de données, initialise les dates, puis retourne ces informations dans un modèle à la vue correspondante.

- La deuxième méthode (HTTP POST) traite la soumission du formulaire et s'occupe de sauvegarder le nouveau médicament avec ses relations.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<ActionResult> Ajouter(OrdonnanceViewModel viewModel)
{
    if (!ModelState.IsValid)
    {
        viewModel.Patients = await _dbContext.Patients.ToListAsync();
        viewModel.Medicaments = await _dbContext.Medicaments.ToListAsync();
        return View(viewModel);
    }
    string? MedecinId = _userManager.GetUserId(User);
    Medecin med = _userManager.FindByIdAsync(MedecinId).Result;
    int result = DateTime.Compare(viewModel.Date_debut, viewModel.Date_fin);
    if (result > 0)
    {
        ModelState.AddModelError("", "Vérifier les dates");
        viewModel.Patients = await _dbContext.Patients.ToListAsync();
        viewModel.Medicaments = await _dbContext.Medicaments.ToListAsync();
        return View(viewModel);
    }
    var patient = await _dbContext.Patients
        .Include(p => p.Antecedents)
        .Include(p => p.Allergies)
        .FirstOrDefaultAsync(p => p.PatientId == (int)viewModel.PatientId);

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

    Ordonnance ordonnance = new Ordonnance
    {
        Posologie = viewModel.Posologie,
        Date_debut = viewModel.Date_debut,
        Date_fin = viewModel.Date_fin,
        Instructions_specifique = viewModel.Instructions_specifique,
        MedecinId = MedecinId,
        Medecin = med,
        PatientId = (int)viewModel.PatientId,
        Patient = patient,
        Medicaments = new List<Medicament>()
    };
};
```

```
if (viewModel.SelectedMedicamentId != null && viewModel.SelectedMedicamentId.Count > 0)
{
    var selectedMedicament = await _dbContext.Medicaments
        .Where(a => viewModel.SelectedMedicamentId.Contains(a.MedicamentId))
        .Include(m => m.Allergies)
        .Include(m => m.Antecedents)
        .ToListAsync();
    foreach (var medicament in selectedMedicament)
    {
        ordonnance.Medicaments.Add(medicament);
        medicament.compteur += 1;
    }
}
else
{
    viewModel.Patients = await _dbContext.Patients.ToListAsync();
    viewModel.Medicaments = await _dbContext.Medicaments.ToListAsync();
    ModelState.AddModelError("", "Veuillez choisir au moins 1 médicament");
    return View(viewModel);
}
if (!VerifyImpossibility(ordonnance))
{
    viewModel.Patients = await _dbContext.Patients.ToListAsync();
    viewModel.Medicaments = await _dbContext.Medicaments.ToListAsync();
    ModelState.AddModelError("", "Le patient ne peut pas avoir ces médicaments");
    return View(viewModel);
}

await _dbContext.Ordonnances.AddAsync(ordonnance);
await _dbContext.SaveChangesAsync();

return RedirectToAction(nameof(Index));
```

Ce bloc de code traite la validation et l'enregistrement d'une ordonnance. Il vérifie que l'utilisateur a sélectionné au moins un médicament, charge les données nécessaires (y compris les allergies/antécédents), vérifie les incompatibilités avec le patient, puis enregistre l'ordonnance en base de données. En cas d'erreur, des messages appropriés sont affichés à l'utilisateur via le ModelState.

3. Maintenant, je vais vous montrer la vue qui affiche le formulaire d'ajout :

```
<h1>Créer une nouvelle ordonnance</h1>

<form asp-action="Ajouter" asp-controller="Ordonnance" method="post">
  <div asp-validation-summary="ModelOnly" class="text-danger"></div>
  <input type="hidden" asp-for="OrdonnanceId" />
  <div class="form-group">
    <label asp-for="Patient"></label>
    <select asp-for="PatientId" class="form-control">
      <option value="">Sélectionné le patient</option>
      @foreach (var pat in Model.Patients)
      {
        <option value="@pat.PatientId">@pat.Nom_p</option>
      }
    </select>
    <span asp-validation-for="PatientId" class="text-danger"></span>
  </div>
  <div class="form-group">
    <label asp-for="Date_debut"></label> <input asp-for="Date_debut" class="form-control" type="date" />
    <span asp-validation-for="Date_debut" class="text-danger"></span>
  </div>

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

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

```

<h4>Medicament</h4>
<div class="form-group">
  @foreach (var med in Model.Medicaments)
  {
    <div class="form-check">
      <input class="form-check-input" type="checkbox" name="SelectedMedicamentId" value="@med.MedicamentId"
        @(Model.SelectedMedicamentId != null && Model.SelectedMedicamentId.Contains(med.MedicamentId) ?
          "checked" : "") />
      <label class="form-check-label">@med.Libelle_med</label>
    </div>
  }
</div>

<br />
<input type="submit" value="Créer une ordonnance" class="btn btn-primary" />
</form>

<div>
  <a asp-action="Index" class="btn btn-secondary">Retour à la liste</a>
</div>

```

Mon formulaire contient des champs texte pour le nom et les contre-indications du médicament, une liste de cases à cocher pour les allergies et antécédents incompatibles, deux boutons pour valider et un pour retourner à la liste.

Quand l'utilisateur soumet le formulaire, les données sont envoyées à la méthode POST du contrôleur que nous avons vue précédemment.

Guide d'utilisation de l'application de gestion médicale

Nous allons voir ensemble comment utiliser les principales fonctionnalités de l'application :

La page de connexion de MedManager est simple et intuitive :

1. Dans le champ "Nom", entrez votre identifiant
2. Dans le champ "Mot de passe", saisissez votre mot de passe
3. Cliquez sur "Se connecter" pour accéder à votre espace

Si vous n'avez pas encore de compte, utilisez le bouton "Créer un compte" en bas du formulaire.

Pour toute assistance, un lien "Nous contacter" est disponible en bas de page.

Connexion

Nom

Mot de passe

☐ Remember me?

Connexion

[Pas de compte ? S'inscrire](#)

Création d'un compte Med Manager

Pour créer votre compte médecin dans l'application :

1. Remplissez vos informations personnelles :
 - Pseudo
 - Nom et prénom
 - Email
2. Choisissez un mot de passe sécurisé
3. Validez avec le bouton "Créer le compte" ou annulez pour revenir à la page de connexion

En cas de besoin, le lien "Nous contacter" reste accessible en bas de page.

Créer un compte

Pseudo

Email

Nom

Prénom

Mots de passe

Confirmer le mots de passe

S'inscrire

Déjà un compte ? Connexion

Liste des Antécédents/Allergies Med Manager

La page de gestion des antécédents vous permet de :

1. Visualiser tous les antécédents médicaux dans un tableau
 - Nom de l'antécédent
 - Bouton "Modifier" pour chaque antécédent
2. Actions disponibles
 - "Ajouter un nouvel antécédent" : créer un nouvelle antécédent
 - "Modifier" : mettre à jour un antécédent existant

Liste des Antécédents

Nom de l'antécédent	Actions
cancer du coeur	 

Ajouter un nouvel antécédent

Édition d'Antécédent Med Manager

Pour modifier un antécédent :

1. Modifiez le nom dans le champ "Nom de l'Antécédent"
2. Cliquez sur "Modifier antécédent" pour valider les modifications

Les modifications seront immédiatement visibles dans la liste des antécédents.

Modifier un antécédent

Antecedent

cancer du coeur

Modifier antécédent

Ajout d'Antécédent Med Manager

Pour ajouter un nouvel antécédent :

1. Saisissez le nom dans le champ prévu
2. Cliquez sur "Créer antécédent" pour valider

Créer un nouvel antécédent

Nom l'antecedent

Les fonctionnalités de gestion des antécédents et d'allergies suivent exactement le même principe avec une liste, des boutons de modification, et des formulaires d'ajout similaires. Pour éviter les répétitions, je ne les détaillerai pas à nouveau.

Liste des Médicaments Med Manager

La liste des médicaments affiche :

























- Nom du médicament

Un boutons est disponible :

- "Ajouter un Médicament" pour créer un nouveaux Médicament

Cette page suit la même logique que celle des antécédents et des allergies.

Liste des Médicaments

Nom du médicament	Actions
Paracétamol	 
Paracétamol	 
Amoxicilline	 
Cétirizine	 
Oméprazole	 
Metformine	 
Clopidogrel	 
Simvastatine	 
Losartan	 
Atorvastatine	 
Tramadol	 
Levothyrox	 

[Ajouter un nouveau médicament](#)

Édition de Médicament Med Manager

Pour modifier un médicament :

1. Section informations
 - Modifiez le nom du médicament
 - Modifier la description

Les modifications sont sauvegardées en cliquant sur "Submit".

Modifier un médicament

Nom du médicament

Paracétamol

Description

Insuffisance hépatique

[Submit](#)

Ajout d'un Médicament Med Manager

Pour ajouter un nouveau médicament :

1. Remplissez les champs obligatoires :
 - Nom du médicament
 - Description

Validez l'ajout avec le bouton "Enregistrer"

Ajouter un médicament

Nom du médicament

Description

Submit

Liste des Patients Med Manager

La liste des patients affiche :

- Informations de base : Nom, Prénom
- Actions disponibles : Modifier, Détails, Supprimer

Pour gérer les patients :

- "Ajouter" : créer un nouveau patient
- "Détails" : voir le dossier complet
- "Modifier" : mettre à jour les informations
- "Supprimer" : retirer un patient de la base

Liste des Patient

Id	Nom	Prénom	Details	Edit	Delete
1	Dupont	Marie			
2	Martin	Jean			
3	Moreau	Sophie			
4	Leroy	Pierre			
5	Bernard	Clara			
6	Rousseau	Emma			
7	Petit	Lucas			
8	Gauthier	Chloé			
9	Lefèvre	Maxime			
10	Fournier	Camille			
11	sandid	lebib			

Ajouter un Patient

Ajout d'un Patient Med Manager

Pour créer un nouveau dossier patient :

1. Informations personnelles

- Nom et prénom
- Sexe (M/F)
- Numéro de sécurité sociale

Une fois tous les champs remplis, validez la création du dossier patient avec "Enregistrer".

Ajouter un Patient

Nom

Prénom

Sexe

☐ Femme

☐ Homme

☐ Non genre

Âge

Poids

Numéro de sécurité sociale

Ajouter Patient

Édition Patient Med Manager

Pour modifier les informations d'un patient :

1. Informations personnelles
 - Modifiez le nom, prénom, sexe ou numéro de sécurité sociale

Éditer Patient

Nom

Prénom

Sexe

Num_secu

Enregistrer

[Retour à la liste](#)

Détails Patient Med Manager

Cette page affiche toutes les informations du patient :

1. Informations personnelles
 - Nom et prénom
 - Sexe
 - Numéro de sécurité
2. État médical
 - Liste des antécédents médicaux
 - Liste des allergies connues

Utilisez "Retour à la liste" pour revenir aux patients.

Detail du Patient

Nom

Martin

Prenom

Jean

Sexe

M

Numéro_secu

98765432109876

Allergies

Antecedents

[Retour à la liste](#)

Suppression Patient Med Manager

Avant de supprimer un patient, le système affiche une page de confirmation avec :

- Les informations du patient à supprimer
- Deux options :
 - "Supprimer" pour confirmer la suppression définitive
 - "Annuler" pour conserver le patient

Cette étape de confirmation évite les suppressions accidentelles.

Supprimer un Patient

Voulez vous vraiment supprimer un Patient 1 ?

[Retour à la liste](#)[Supprimer](#)

Liste des Ordonnances Med Manager

La liste des ordonnances montre :

- Informations de l'ordonnance : médecin, dates, posologie
- Informations du patient
- Actions disponibles :
 - Modifier l'ordonnance
 - Voir les détails
 - Supprimer

Options disponibles :

- "Ajouter une nouvelle ordonnance" pour en créer une nouvelle

Liste des Ordonnances

Médecin	Patient	Date début	Date fin	Posologie	Instructions spécifiques	Actions		
Goat	Petit	24/11/2024 00:00:00	25/11/2024 00:00:00	2 par jour		Détails	Modifier	Supprimer
Goat	sandid	01/04/2025 00:00:00	02/04/2025 00:00:00	2 par jour		Détails	Modifier	Supprimer

[Ajouter une nouvelle ordonnance](#)

Édition d'Ordonnance Med Manager

Pour modifier une ordonnance existante :

1. Informations principales
 - Posologie : instructions de prise du médicament
 - Dates : début et fin du traitement
 - Instructions spécifiques complémentaires
2. Sélection
 - Patient concerné

- Médicaments prescrits (cases à cocher)

Validez avec "Enregistrer" ou annulez avec "Retour à la liste".

Éditer Ordonnance

Posologie	<input type="text" value="2 par jour"/>
Date_debut	<input type="text" value="24/11/2024"/>
Date_fin	<input type="text" value="25/11/2024"/>
Instructions_specifique	<input type="text"/>
Patient	<input type="text" value="Petit"/>

Médicaments

- ☐ Paracétamol
- ☐ Paracétamol
- ☐ Amoxicilline
- ☐ Cétirizine
- ☐ Oméprazole
- ☒ Metformine
- ☐ Clopidogrel
- ☐ Simvastatine
- ☐ Losartan
- ☐ Atorvastatine
- ☐ Tramadol
- ☐ Levothyrox

Détails de l'Ordonnance Med Manager

La page affiche tous les éléments de l'ordonnance :

1. Traitement
 - Posologie prescrite
 - Dates de début et fin
 - Instructions spécifiques
2. Intervenants
 - Patient concerné
 - Médecin prescripteur
3. Liste des médicaments prescrits

"Retour à la liste" permet de revenir aux ordonnances.

Detail de l'ordonnance

Medecin

Goat

Patient

Lucas

Posologie

2 par jour

Date_debut

24/11/2024

Date_fin

25/11/2024

Instructions_specifique

Medicaments

- Metformine

[Retour à la liste](#)

Nouvelle Ordonnance Med Manager

Pour créer une ordonnance :

- Posologie du traitement
- Dates de début et fin
- Instructions spécifiques si nécessaire
- Sélection du patient dans la liste déroulante
- Choix des médicaments prescrits

Validez avec "Créer une ordonnance" ou annulez avec "Retour à la liste".

Créer une nouvelle ordonnance

Patient

Sélectionné le patient

Date_debut

02/05/2025

Date_fin

03/05/2025

Posologie

Instructions_specifique

Medicament

- ☐ Paracétamol
- ☐ Paracétamol
- ☐ Amoxicilline
- ☐ Cétirizine
- ☐ Oméprazole
- ☐ Metformine
- ☐ Clopidogrel
- ☐ Simvastatine
- ☐ Losartan
- ☐ Atorvastatine
- ☐ Tramadol
- ☐ Levothyrox

Créer une ordonnance

Retour à la liste

Ce rapport vous a présenté l'ensemble des fonctionnalités de Med Manager, de la gestion des patients à la création d'ordonnances. L'interface intuitive et la navigation fluide vous permettent de :

- Gérer efficacement vos patients et leurs dossiers
- Créer et suivre les ordonnances
- Accéder rapidement aux informations médicales
- Assurer un suivi optimal des traitements

Annexe

Annex 1 : Lien du Projet

<https://ap2-medmanager.kemyl.fr/>

Annexe 2 : Git

Ci-dessous un lien pour accéder à la page Git de mon Application complète :

https://github.com/Kemyyl/AP2_MedManager