Maui

▼ Technologie

- Site web
  - on peut utiliser l’accéléromètre
- Natif spécifique (compiler pour le hardware cible)
- Natif lié (Kivy, Kotlin)
- Natif avec runtime: code tourne sur une sorte de VM (.NET MAUI, React, Flutter)

▼ Structure code

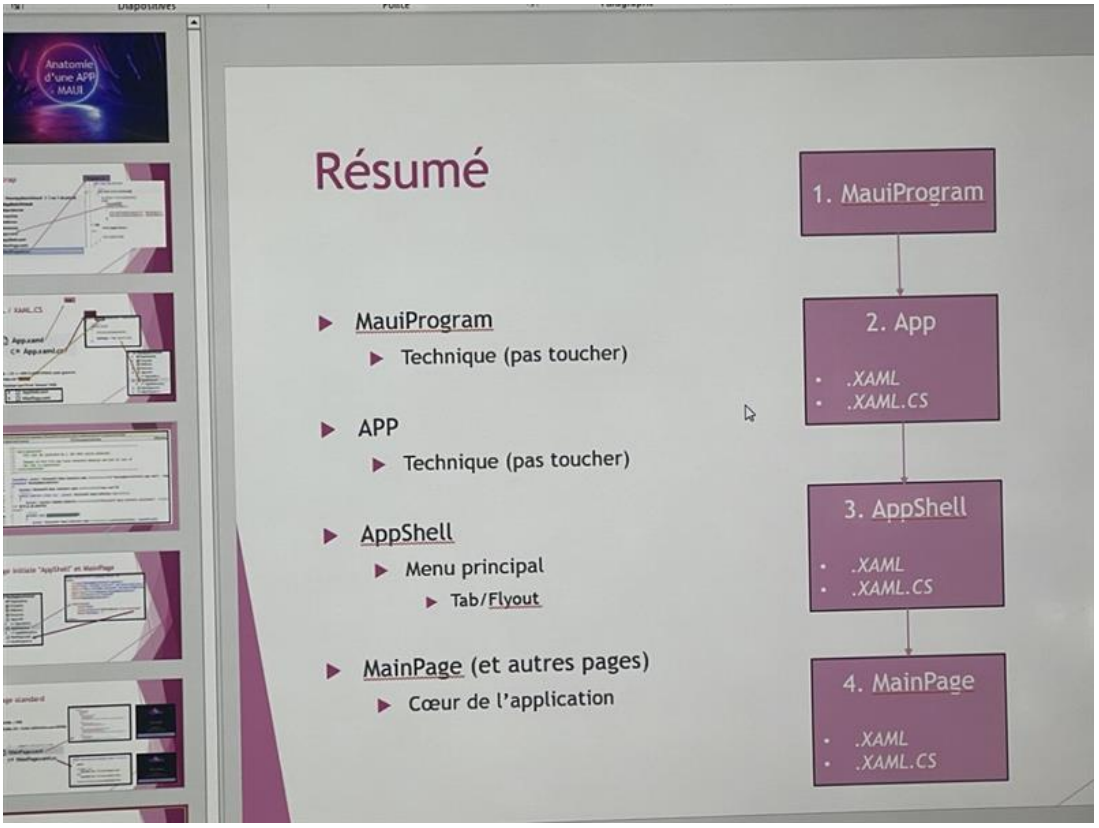
Une classe est composé de :

- un fichier XAML (Frontend)
- un fichier CS (Backend)
- Les 2 formes une classe final auto générer

Il y a le mot clé partial qui permet de fusionner un fichier XAML et CS en une et même classe

Partial : permet de découper une class

▼ structure des fichiers



▼ XAML

- Ressemble au HTML
- ▼ utilise des balises
  - ça utilise des attributs
    - <ingredient quantite=200 />
- ▼ exemples
  - <four></four>
  - <four />
- Il faut utiliser des balises définies par les namespaces XML

▼ AppShell

- Dans le AppShell on peut choisir les types de pages
  - Tabulation
    - dans tab, il y a l'option route qui est l'url de la poage
  - on peut masquer des entrée
- Menu Hamburger (Flyout)

- ne pas oublié le flyoutDisplayOptions="AsMultipleItems"

▼ Navigation

Il y a plusieurs niveau de navigation (maximum 3)

si tab ou flyout ne se souvient pas des anciennespages

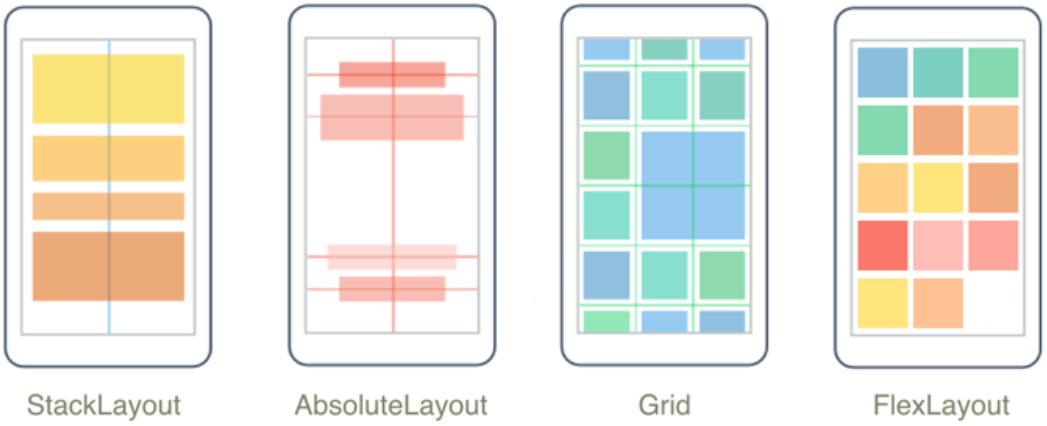
un peut utiliser des boutons customs et va empiler les pages

il y a la navigation modale pour forcer les personnes a réaliser (pas de boutons back)

▼ Layout

Le layout permet de positionner des éléments

▼ 4 types



on peut utiliser une exstension

▼ Interactions utilisateurs

Code interactions :

- Code behind (code de manière pas propre)
- M V VM (Model, View, View-Model) (librairies MVVM: Toolkit)
- MVU
- Blazor (C# / HTML)

▼ M V VM

Model

- C#

- data source / DB
- Travaille avec View-Model
- Logique model (règles du métier)
- Peut être une api

View-Model

- C#
- logique code
- Travaille avec Model et View
- Navigation (utiliser Shell.Current pour récupérer la page actuelle)

View

- XAML
- partie graphique
- Travaille avec View-Model

▼ CRUD (create, read, update, delete)

- un modèle
  - une classe c
  - Un moyen de stockage (SQLite)
  - un ORM (Entity framework)
- une vue
  - Entry + button + CollectionView

▼ DbContext

DbContext permet de décrire les tables comme Sequelize

pour forcer créer la création/mise à jour de la db

- this.Database.EnsureCreated();

La méthode OnConfiguring permet de configurer le sgbd (elle est exécutée automatiquement)

lors de l'utilisation, on utilise using

pour sauvgarder dans la db dbContext.SaveChangesAsync()

▼ Animations

Le code de l’animation est sur la vue mais le déclenchement de l’animation est dans le view model

▼ animation de base sans plugin :

- fade
- rotation
- scale
- translate

pour le code :

- on a besoin d’un binding sur les éléments de la vue

▼ Switch / toggle

ils n’ont pas de “command”

on peut utiliser une fonction partial OnToggledChanged et sur la vue on utilise IsToggled

▼ Accéléromètre

Il prend en compte plusieurs axe (x, y, z)

3 phases :

1. Je m’intéresse aux données de capteurs
2. je récupère périodiquement les données (par défaut 200ms)
3. je ne m’intéresse plus aux données du capteur

▼ Test unitaire

▼ Structure des test

- Contexte
- Action à tester
- Vérification

▼ Limitation de MAUI

on ne peut pas tester uniquement un projet MAUI (on doit créer un projet de type MAUI Library)

▼ Test d’intégration

C’est des scénario sur une feuille, un modèle est dispo sur moodle

using () {} permet d’allouer et désallouer de la mémoire (relâcher les ressources)

lorsqu’on veut passe une fonction en paramètre on peut utiliser Func ou Action (Func<Task>) comme type

on peut associer une fonction dans un xaml.cs à une variable (ex: test) vm et pour appeler la fonction, on peut faire test.Invoke()