OKKT - Osztálykirándulás Tervező Alkalmazás

Fejlesztői dokumentáció

Készítette: a ;expected csapata

Dátum: 2025. október 17.

Verzió: 1.0

Tartalomjegyzék

1. Projekt áttekintés

1.1. Alkalmazás célja

Az OKKT egy mobilalkalmazás, amelynek célja az osztálykirándulások tervezésének, költségmegosztásának és pénzügyi előkészületeinek megkönnyítése. Az alkalmazás lehetővé teszi pedagógusok és szülők számára a kirándulások költségeinek hatékony kezelését, a diákok zsebpénzének elemzését és a kiadások dokumentálását.

1.2. Elérhetőség

A projekt forráskódja és további információi elérhetők a GitHubon: https://github.com/zsoltikv/

1.3. Főbb funkciók

- **Költségtervezés**: Részletes kedvezménykezeléssel.
- **Zsebpénz elemzés**: Egyéni és csoportos módban.
- **Vizuális jelentések**: Diagramokkal.
- **Intelligens javaslatok**: Költségfedezeti problémák megoldására.
- **Fotókezelés**: Számlák dokumentálásához.
- **PDF export**: Professzionális jelentésekhez.
- **Adatmentés**: Lokális tárolásban.

1.4. Célközönség

- Osztályfőnökök és tanárok
- Szülők és iskolai koordinátorok
- Diákcsoportok vezetői

2. Technológiai stack

2.1. Fő technológiák

- **.NET MAUI 8.0**: Platformfüggetlen keretrendszer
- **C# 11**: Backend programozási nyelv
- **XAML**: UI definíciók
- **MVVM minta**: Architektúra

2.2. Külső könyvtárak

```
<PackageReference Include="Microsoft.Maui.Controls"
   Version="$(MauiVersion)" />
<PackageReference Include="Microsoft.Extensions.Logging.Debug"
   Version="9.0.0" />
<PackageReference Include="PdfSharpCore" Version="1.3.67" />
```

2.3. Platform támogatás

- **Android**: API 21+
- **Windows**: 10.0.19041.0+

3. Architektúra és design minták

3.1. Alkalmazás szerkezet

```
OKKT/
├── Models/                  // Adatmodellek
│   ├── TripData.cs
│   ├── TripSummary.cs
│   ├── CostItem.cs
│   └── MauiEntryExtensions.cs
├── Views/                  // UI oldalak
│   ├── MainPage.xaml
│   ├── PastTripsPage.xaml
│   ├── TripDetailPage.xaml
│   └── ImageViewPage.xaml
├── ViewModels/            // Üzleti logika
├── Resources/            // Erőforrások
└── Services/            // Szolgáltatások
```

3.2. Design minták

3.2.1. MVVM (Model-View-ViewModel)

- **Model**: TripData, CostItem, TripSummary osztályok
- **View**: XAML fájlok (MainPage.xaml, stb.)
- **ViewModel**: Code-behind fájlok üzleti logikával

3.2.2. Data Binding

```
<Label Text="{Binding TripName}"
       FontSize="18"
       FontAttributes="Bold"
       TextColor="#FFD700"/>
```

3.2.3. Command Pattern

```
public ICommand TripTappedCommand { get; }
TripTappedCommand = new Command<TripSummary>(async (trip) => await
   OpenTripAsync(trip));
```

4. Adatmodel leírás

4.1. TripData osztály

```
public class TripData
{
    // Kirándulás alapadatok
    public string TripName { get; set; } = string.Empty;
    public string TripDestination { get; set; } = string.Empty;
    public DateTime TripDateStart { get; set; } = DateTime.Now;
    public DateTime TripDateEnd { get; set; } = DateTime.Now;

    // Részvételi adatok
    public int Participants { get; set; }
    public bool IsPerPersonMode { get; set; }

    // Pénzügyi adatok
    public List<CostItem> Costs { get; set; } = new List<CostItem>();
    public List<double> PocketMoney { get; set; } = new
       List<double>();
    public double AveragePocketMoney { get; set; }

    // űMszaki adatok
    public List<string> PhotoPaths { get; set; } = new
       List<string>();
    public DateTime LastSaved { get; set; } = DateTime.Now;
    public bool Calculated { get; set; } = false;
}
```

4.2. CostItem osztály

```
public class CostItem
{
    public string Type { get; set; } = string.Empty;
    public double Amount { get; set; }
    public int NumberOfPeople { get; set; }
    public bool HasDiscount { get; set; }
    public double DiscountAmount { get; set; }
    public int DiscountNumberOfPeople { get; set; }
}
```

4.3. TripSummary osztály

```
public class TripSummary
{
    public string FileName { get; set; }
    public string TripName { get; set; }
    public string TripDestination { get; set; }
    public DateTime TripDateStart { get; set; }
    public DateTime TripDateEnd { get; set; }
    public DateTime LastSaved { get; set; }
    public int Participants { get; set; }
    public double TotalCost { get; set; }
}
```

5. Felhasználói felület és navigáció

5.1. Alkalmazás szerkezet

5.1.1. AppShell navigáció

```
<TabBar>
    <ShellContent
        Title="Új kirándulás"
        Icon="travel.png"
        ContentTemplate="{DataTemplate local:MainPage}" />
    <ShellContent
        Title="Tervezetek"
        Icon="folder.png"
        ContentTemplate="{DataTemplate local:PastTripsPage}" />
</TabBar>
```

5.1.2. Fő oldalak

- **MainPage**: Új kirándulás létrehozása
- **PastTripsPage**: Mentett kirándulások böngészése
- **TripDetailPage**: Kirándulás részletes nézete
- **ImageViewPage**: Kép megtekintése és kezelése

5.2. UI/UX Design elvek

5.2.1. Színpaletta

- **Primary**: #121212 (háttér)
- **Secondary**: #1E1E1E (kártyák)
- **Accent**: #FFD700, #FF9800, #FFA500 (kiemelések)
- **Text**: #FFFFFF, #C8C8C8 (szövegek)

5.2.2. Tipográfia

- **Betűcsalád**: Noto Sans
- **Méretek**: 11px - 24px
- **Súlyok**: Regular, Bold

6. Üzleti logika részletezés

6.1. Költségszámítás algoritmus

6.1.1. Teljes költség számítás

```
double totalCost = tripData.Costs.Sum(c =>
    (c.Amount * c.NumberOfPeople) +
    (c.DiscountAmount * c.DiscountNumberOfPeople));
```

6.1.2. Fedezettségi elemzés

```
bool canPay = monthlyTotal >= costPerPerson;
double shortage = costPerPerson - monthlyTotal;
```

6.1.3. Intelligens javaslatok

```
// Költségcsökkentés
double neededReduction = totalShortage;

// Több diák fizet
double extraPerPerson = totalShortage / (participants -
   cantPayList.Count);

// Több őid
int neededMonths = (int)Math.Ceiling(costPerPerson /
   pocketMoneyList.Min());
```

6.2. Adatvalidáció

6.2.1. Bemeneti ellenőrzések

```
if (!double.TryParse(pocketMoneyEntries[i].Text, out double amount)
   || amount < 0)
{
    await ShowError($"Kérlek add meg a {i + 1}. diák érvényes
       zsebpénzét!");
    return;
}
```

6.2.2. Dátum validáció

```
private void OnEndDateChanged(object sender, DateChangedEventArgs e)
{
    if (TripDateEnd.Date < TripDateStart.Date)
    {
        TripDateEnd.Date = TripDateStart.Date;
    }
}
```

7. Adatkezelés és tárolás

7.1. JSON szerializáció

```
public async Task SaveTripDataAsync()
{
    string json = System.Text.Json.JsonSerializer.Serialize(tripData,
        new System.Text.Json.JsonSerializerOptions { WriteIndented =
           true });

    string tripFileName = $"{tripData.TripName}.json";
    string filePath =
       Path.Combine(FileSystem.Current.AppDataDirectory,
       tripFileName);
    await File.WriteAllTextAsync(filePath, json);
}
```

7.2. Fájlkezelés

7.2.1. Mentés helye

```
string appDataDir = FileSystem.Current.AppDataDirectory;
var jsonFiles = Directory.GetFiles(appDataDir, "*.json");
```

7.2.2. Fotókezelés

```
string targetPath = Path.Combine(FileSystem.Current.AppDataDirectory,
                                 Path.GetFileName(result.FullPath));
```

7.3. Adatbetöltés

```
private async void LoadTrips()
{
    trips.Clear();
    var appDataDir = FileSystem.Current.AppDataDirectory;
    var jsonFiles = Directory.GetFiles(appDataDir, "*.json");

    foreach (var file in jsonFiles)
    {
```

```
        var json = await File.ReadAllTextAsync(file, Encoding.UTF8);
        var tripData = JsonSerializer.Deserialize<TripData>(json);
        // Feldolgozás...
    }
}
```

8. Grafikus elemek és diagramok

8.1. Egyéni rajzolás (IDrawable)

8.1.1. PieChartDrawable

```
public class PieChartDrawable : IDrawable
{
    public void Draw(ICanvas canvas, RectF dirtyRect)
    {
        // Kördiagram rajzolása
        DrawPieSlice(canvas, centerX, centerY, radius, -90,
           canPayAngle, Color.FromArgb("#4CAF50"));
    }
}
```

8.1.2. CostBreakdownDrawable

```
public class CostBreakdownDrawable : IDrawable
{
    public void Draw(ICanvas canvas, RectF dirtyRect)
    {
        // Költségmegoszlási diagram
        foreach (var cost in costs)
        {
            float percentage = (float)(costValue / totalCost);
            float sweepAngle = percentage * 360f;
            DrawPieSlice(canvas, centerX, centerY, radius,
               startAngle, sweepAngle, sliceColor);
        }
    }
}
```

8.2. GraphicsView használata

```
public class PieChartView : GraphicsView
{
    public PieChartView(List<double> pocketMoney, double cost, int
       months)
    {
        Drawable = new PieChartDrawable(pocketMoney, cost, months);
    }
}
```

9. PDF export funkcionalitás

9.1. PDF generálás PdfSharpCore-dal

9.1.1. Dokumentum létrehozása

```
using (var document = new PdfSharpCore.Pdf.PdfDocument())
{
    var page = document.AddPage();
    page.Size = PdfSharpCore.PageSize.A4;

    using (var gfx = XGraphics.FromPdfPage(page))
    using (var xImage = XImage.FromFile(tempImagePath))
    {
        gfx.DrawImage(xImage, 0, 0, pageWidth, pageHeight);
    }

    document.Save(filePath);
}
```

9.1.2. Platformfüggetlen útvonalak

```
#if ANDROID
var downloadsPath = Android.OS.Environment
    .GetExternalStoragePublicDirectory(Android.OS.Environment.DirectoryDownloads
    .AbsolutePath;
#elif WINDOWS
var downloadsPath = Path.Combine(
    Environment.GetFolderPath(Environment.SpecialFolder.UserProfile),
    "Downloads");
#endif
```

9.2. Képernyőkép készítése

```
var screenshot = await pdfView.CaptureAsync();
```

10. Fotókezelés és média

10.1. Képfeltöltés lehetőségek

```
string action = await DisplayActionSheet("Fénykép hozzáadása",
   "Mégse", null,
                                          "📷 Kamera", "🖼 Galéria");
```

10.2. Kép nagyítás és navigáció

```
public partial class ImageViewPage : ContentPage
{
```

```
    private double currentScale = 1;
    private double startScale = 1;

    private void OnPinchUpdated(object sender,
       PinchGestureUpdatedEventArgs e)
    {
        // Pinch zoom implementáció
    }
}
```

10.3. Gesture kezelés

```
var pinchGesture = new PinchGestureRecognizer();
pinchGesture.PinchUpdated += OnPinchUpdated;
FullImage.GestureRecognizers.Add(pinchGesture);

var doubleTap = new TapGestureRecognizer { NumberOfTapsRequired = 2
   };
doubleTap.Tapped += (s, e) => ResetZoom();
```

11. Platformfüggő implementációk

11.1. Android specifikus beállítások

11.1.1. StatusBar szín beállítás

```
#if ANDROID
Microsoft.Maui.Handlers.WindowHandler.Mapper.AppendToMapping(
    "StatusBarColor",
    (handler, view) =>
    {
        var activity = Platform.CurrentActivity;
        var window = activity.Window;
        window.SetStatusBarColor(Android.Graphics.Color.ParseColor("#121212"));
    });
#endif
```

11.1.2. RadioButton testreszabás

```
#if ANDROID
radioButton.HandlerChanged += (s, e) =>
{
    if (radioButton.Handler.PlatformView is
       Android.Widget.RadioButton androidRadioButton)
    {
        var colorStateList = new ColorStateList(states, colors);
        androidRadioButton.ButtonTintList = colorStateList;
    }
};
```

```
#endif
```

12. Hibakezelés és logolás

12.1. Globális hibakezelés

```
try
{
    // űMvelet végrehajtása
}
catch (Exception ex)
{
    await DisplayAlert("Hiba", $"Nem sikerült a űmvelet:
       {ex.Message}", "OK");
}
```

12.2. Logolás konfiguráció

```
#if DEBUG
builder.Logging.AddDebug();
#endif
```

12.3. Felhasználóbarát hibaüzenetek

```
await DisplayAlert("Hiba", "Kérlek adj meg érvényes költség összeget
   minden tételhez!", "OK");
```

13. Animációk és felhasználói élmény

13.1. Egyszerű animációk

```
private async void AnimateView(View view)
{
    view.Opacity = 0;
    await view.FadeTo(1, 300);
}
```

13.2. Hibajelzés animáció

```
private async Task ShowError(string message)
{
    await DisplayAlert("Hiba", message, "OK");
    await BtnCalculate.TranslateTo(-15, 0, 50);
    await BtnCalculate.TranslateTo(15, 0, 50);
    // Reszponzív shake animáció
}
```

14. Tesztelési stratégia

14.1. Tesztelendő komponensek

14.1.1. Unit tesztek

- Költségszámítás logika
- Adatvalidációk
- JSON szerializáció/deszerializáció

14.1.2. UI tesztek

- Navigáció működése
- Adatkötések helyessége
- Reszponzív design

14.1.3. Integrációs tesztek

- Fájl I/O műveletek
- PDF generálás
- Fotókezelés

14.2. Tesztkörnyezet

- **Android Emulátor**: API 21+
- **Windows Desktop**: 10.0.19041.0+
- **Fizikai eszközök**: Különböző képernyőméretek

15. Bevezetési útmutató

15.1. Fejlesztői környezet beállítása

15.1.1. Előfeltételek

- Visual Studio 2022 17.8 vagy újabb
- .NET 8.0 SDK
- Android SDK (API 21 vagy újabb)
- Windows 10 SDK (19041 vagy újabb)

15.1.2. Projekt betöltése

```
git clone https://github.com/zsoltikv/OKKT
cd OKKT
dotnet restore
```

15.2. Build és deploy

15.2.1. Android

```
dotnet build -f net8.0-android -c Release
dotnet publish -f net8.0-android -c Release
```

15.2.2. Windows

```
dotnet build -f net9.0-windows10.0.19041.0 -c Release
dotnet publish -f net9.0-windows10.0.19041.0 -c Release
```

16. Ismert korlátozások és jövőbeli fejlesztések

16.1. Jelenlegi korlátozások

- Nincs felhőszinkronizáció
- Limitált képméretek kezelése
- Csak lokális adattárolás

16.2. Jövőbeli bővítési lehetőségek

- Felhő integráció: OneDrive, Google Drive
- Többnyelvűség: Angol, német támogatás
- Haladó jelentések: Excel export, statisztikák
- Collaboration: Több felhasználós szerkesztés
- Offline mód: Teljes offline funkcionalitás

17. Biztonsági megfontolások

17.1. Adatvédelem

- Minden adat lokálisan tárolódik
- Nincs adatgyűjtés külső szerverekre
- Fájlok titkosítva tárolhatók jövőben

17.2. Engedélyek

```
<!-- Android engedélyek -->
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

18. Teljesítményoptimalizálás

18.1. Memóriakezelés

```
using (var stream = await screenshot.OpenReadAsync())
using (var fileStream = File.Create(tempImagePath))
{
    await stream.CopyToAsync(fileStream);
}
```

18.2. UI reszponzivitás

```
await Task.Run(() => {
    // CPU intenzív űmveletek
});
```

19. Kódminőség és karbantartás

19.1. Kódstílus és konvenciók

- **C# naming conventions**: camelCase, PascalCase
- **XAML coding standards**: Konzisztens formázás
- **Comment policy**: Komplex logika dokumentálása

20. Összefoglalás

Az OKKT alkalmazás egy átfogó megoldást kínál osztálykirándulások tervezésére és költségkezelésére. A .NET MAUI keretrendszer használata lehetővé teszi a platformfüggetlen fejlesztést, miközben natív felhasználói élményt biztosít. Az alkalmazás moduláris architektúrája, részletes üzleti logikája és felhasználóbarát felülete ideális eszközt kínál pedagógusok és szülők számára a kirándulások hatékony megszervezéséhez.