ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Кваліфікаційна робота
Перший(бакалаврський) рівень

Кросплатформний мобільний застосунок для менеджменту та аналізу задач

Виконав:
Данил МИРГОРОДСЬКИЙ
ст.гр. КІУКІ-21-5

Керівник:
Юлія АНДРУСЕНКО
асистент кафедри ЕОМ

Task Manager

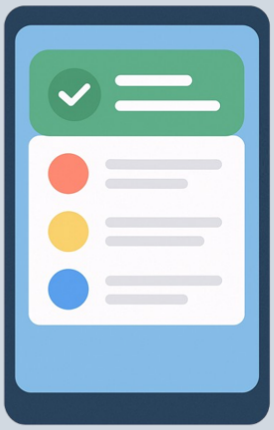
Task Manager, або менеджер задач – це інструмент призначений для управління завданнями. Він допомагає створювати, організовувати, відстежувати та виконувати завдання, як у повсякденному житті, так і в робочих проектах.

Це корисний інструмент як для особистого використання, так і для командної роботи.

Існуючі аналоги менеджерів задач

Todoist	Any.do	Microsoft To Do	Tick Tick
+ Розумне додавання задач + Потужна система фільтрів, тегів і пріоритетів + Інтеграція з популярними сервісами – Більшість функцій доступні лише в платній версії –Інтерфейс може бути складним	+ Інтуїтивний, мінімалістичний інтерфейс + Підтримка голосового введення + Повна кросплатформність – Відсутні теги та фільтри – Обмежений функціонал у безкоштовній версії.	+ Повністю безкоштовний + Інтеграція з екосистемою Microsoft + Простий і зручний інтерфейс. – Відсутні фільтри, теги та аналітика – Обмежені функції кастомізації –Мінімум можливостей для командної роботи	+ Багатий функціонал + Підтримка Pomodoro + Зручне тижневе планування – Частина функцій обмежені або не доступні у free-версії – Може здатися складним для нових користувачів

Постановка задачі

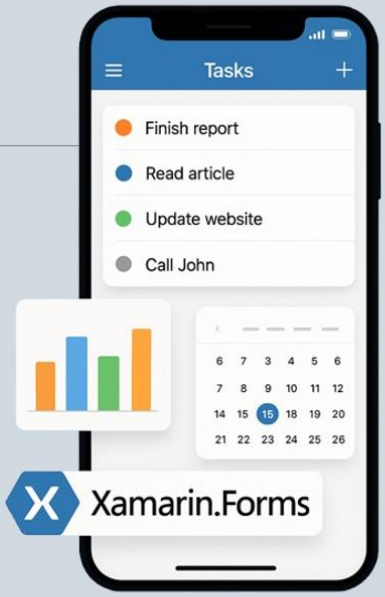


Метою даної роботи стало створення зручного, функціонального та адаптивного таскменеджера, який дозволяє не лише швидко фіксувати задачі, але й аналізувати ефективність, відстежувати повторювані справи, працювати автономно та залишатися простим у використанні.

- •Керування завданнями
- •Аналіз та статистика
- •Персоналізація
- • Повністю автономна робота та сповіщення

Реалізація застосунку

- Xamarin.Forms
- Мова програмування C#
- SQLite
- Microcharts та Xamarin.Plugin.Calendar



TaskItem	
Id	INTEGER
Title	TEXT
Description	TEXT
CategoryIcon	TEXT
DueDate	INTEGER
IsCompleted	INTEGER

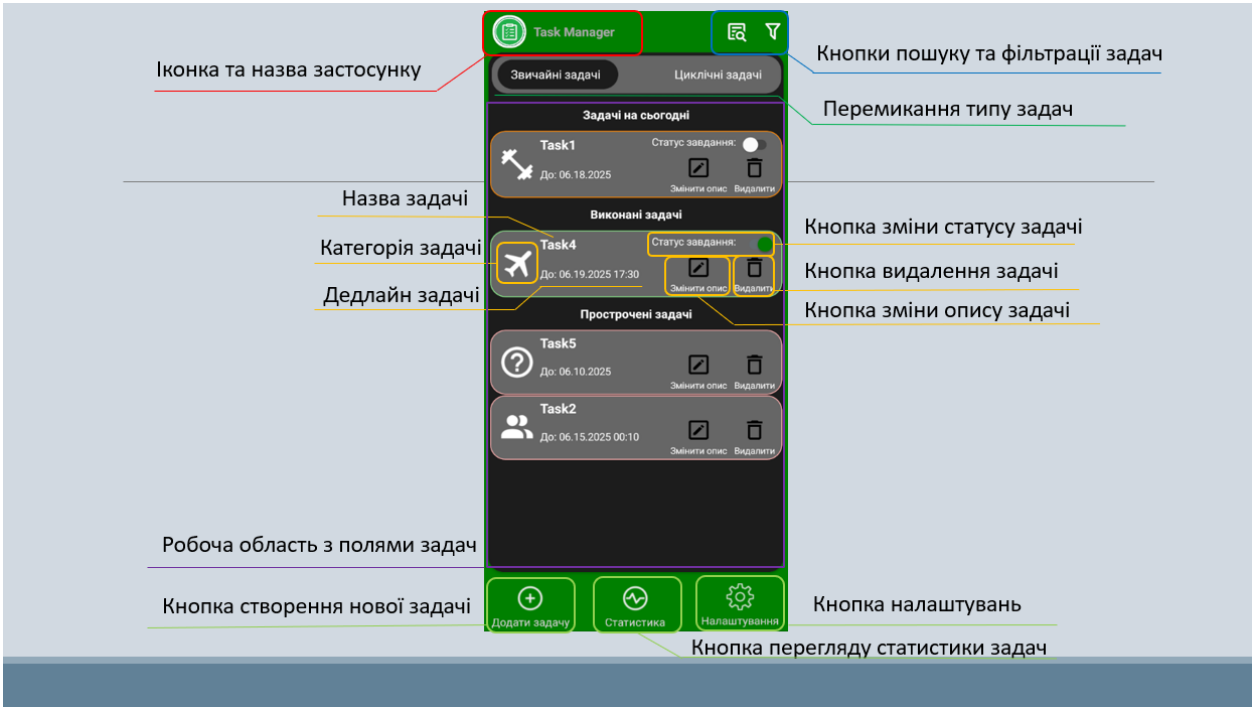
RecurringTaskItem	
Id	INTEGER
Title	TEXT
Description	TEXT
CategoryIcon	TEXT
StartDate	INTEGER
RepeatInterval	INTEGER
RepeatNumber	INTEGER
RepeatCount	INTEGER
IsActive	INTEGER
CompletionStatesSerialized	

Збереження даних

Для збереження використовується SQLite + SQLite.Net-PCL локальна база даних, що не потребує інтернету

TaskItem / RecurringTaskItem – моделі задач із повноцінною логікою обробки

Переваги: повна автономність, швидкодія, простота інтеграції, підтримка SQL



Операції над задачами

Над самими задачами передбачено виконання наступних операцій:

- Створення нової задачі
- Редагування будь-яких параметрів існуючої задачі
- Перегляд детальної інформації по задачі
- Фільтрування задач по назві та категоріям
- Видалення задачі

Перегляд статистики задач

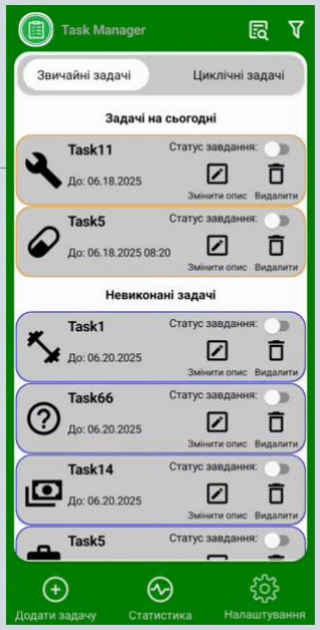
Перегляд статистики відбуваються через спеціалізоване вікно статистики.

- Призначення: Аналіз ефективності користувача
- Особливості:
  - Кругові та стовпчикові діаграми
  - Інтерактивний календар
  - Поради на основі активності
- Технічна реалізація:
  - Сторонні бібліотеки: Microcharts, Xamarin.Plugin.Calendar
  - Дані з SQLite
  - Свайп-навігація між сторінками

Налаштування застосунку

Налаштування в застосунку відбуваються через спеціалізоване вікно налаштувань

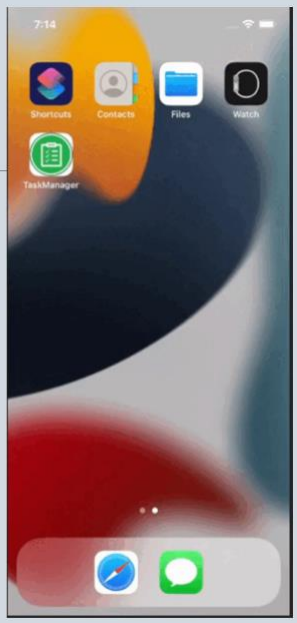
- Призначення: Зміна теми, мови, сповіщень
- Особливості:
  - Попап вибору теми та кольрової схеми застосунку
  - Зміна мови через локалізацію
  - Увімкнення сповіщень → поява TimePicker
  - Панель збереження змін
- Технічна реалізація:
  - MVVM + Preferences
  - Анімації через AnimateHeight()



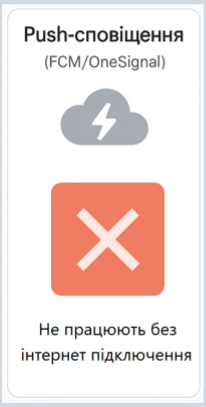
Реалізація на iOS

Застосунок повністю підтримується на пристроях з операційною системою iOS.

Завдяки єдиній кодовій базі, функціональні можливості та логіка роботи програми залишаються ідентичними як для Android, так і для iOS. Користувачі iPhone мають доступ до всіх основних можливостей — керування завданнями, аналітики, зміни теми, локалізації, а також системи щоденних сповіщень.

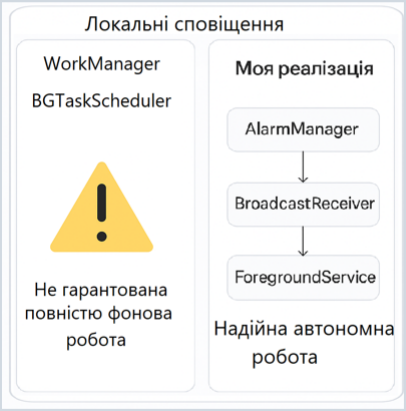


Система сповіщень



Найпоширенішим методом у сучасних мобільних застосунках є використання **push-сповіщень.** Проте цей варіант потребує постійного підключення до інтернету та зовнішнього сервера, що суперечить вимогам автономності застосунку.

Іншим методом є використання **локальних сповіщення**, які генеруються безпосередньо на пристрої користувача. Основною складністю при реалізації даних сповіщень, є їх автономне планування.



Висновки по роботі

В ході виконання роботи було виконано всі поставлені задачі.

Застосунок був протестований активним використанням впродовж останніх декількох тижнів, і в загальному не погано себе зарекомендував з практичної точки зору.

Для даного проекту є не погані перспективи для подальшого розвитку.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

Код реалізованої системи сповіщень

Б.1 Метод ScheduleDailyTaskUpdate

```
public void ScheduleDailyTaskUpdate()
{
    bool notificationsEnabled =
Preferences.Get("NotificationsEnabled", true);

    var alarmIntent = new
Intent(Android.App.Application.Context, typeof(AlarmReceiver));
    alarmIntent.SetAction("com.newmob.TASK_UPDATE_ALARM");

    var pendingIntent = PendingIntent.GetBroadcast(
        Android.App.Application.Context, 0, alarmIntent,
        PendingIntentFlags.UpdateCurrent |
PendingIntentFlags.Immutable);

    var alarmManager =
(AlarmManager)Android.App.Application.Context.GetSystemService(A
ndroid.Content.Context.AlarmService);

    if (notificationsEnabled)
    {
        DateTime now = DateTime.Now;
        DateTime nextRunTime = DateTime.Today.AddHours(1);

        if (nextRunTime < now)
        {
            nextRunTime = nextRunTime.AddDays(1);
        }

        long triggerTimeMillis = (long)(nextRunTime -
DateTime.Now).TotalMilliseconds +
Java.Lang.JavaSystem.CurrentTimeMillis();

Android.App.Application.Context.SendBroadcast(alarmIntent);
        alarmManager.SetRepeating(
            AlarmType.ElapsedRealtimeWakeup,
             SystemClock.ElapsedRealtime(),
              12 * 60 * 60 * 1000, // кожні 8 годин
            pendingIntent);

        System.Diagnostics.Debug.WriteLine("Notification system
enabled and scheduled.");
    }
```

```
    else
    {
        alarmManager.Cancel(pendingIntent);
        var stopIntent = new
Intent(Android.App.Application.Context,
typeof(TaskUpdateForegroundService));
        Android.App.Application.Context.StopService(stopIntent);

        var notificationManager =
NotificationManagerCompat.From(Android.App.Application.Context);
        notificationManager.Cancel(1);


Plugin.LocalNotification.NotificationCenter.Current.Cancel(1002)
;
        System.Diagnostics.Debug.WriteLine("Notification system
disabled.");
    }
}
```

Б.2 Метод AlarmReceiver

```
[BroadcastReceiver(Enabled = true, Exported = false)]
[IntentFilter(new[] { Intent.ActionBootCompleted,
"com.newmob.TASK_UPDATE_ALARM" })]
public class AlarmReceiver : BroadcastReceiver
{
    public override void OnReceive(Context context, Intent
intent)
    {
        System.Diagnostics.Debug.WriteLine($"AlarmReceiver
triggered at: {DateTime.Now}");
        if (intent.Action == Intent.ActionBootCompleted ||
intent.Action == "com.newmob.TASK_UPDATE_ALARM")
        {
            var serviceIntent = new Intent(context,
typeof(TaskUpdateForegroundService));
            if (Build.VERSION.SdkInt >= BuildVersionCodes.O)
            {
                context.StartForegroundService(serviceIntent);
            }
            else
            {
                context.StartService(serviceIntent);
            }
        }
    }
}
```

Б.3 Метод TaskUpdateForegroundService

```
[Service]
public class TaskUpdateForegroundService : Service
{
    private const string CHANNEL_ID = "task_update_channel";

    public override void OnCreate()
    {
        base.OnCreate();
        CreateNotificationChannel();
    }

    public override IBinder OnBind(Intent intent) => null;

    public override StartCommandResult OnStartCommand(Intent
intent, StartCommandFlags flags, int startId)
    {


System.Diagnostics.Debug.WriteLine($"TaskUpdateForegroundService
started at: {DateTime.Now}");
        var notification = new NotificationCompat.Builder(this,
CHANNEL_ID)
            .SetContentTitle("Task Manager")
            .SetContentText("Система сповіщень активна")
            .SetSmallIcon(Resource.Drawable.mainicon)
            .SetPriority((int)NotificationPriority.Min)
            .SetOngoing(true)
            .Build();

        StartForeground(1, notification);

        OneTimeWorkRequest workRequest = new
OneTimeWorkRequest.Builder(typeof(TaskUpdateWorker)).Build();
        WorkManager.GetInstance(this).Enqueue(workRequest);

        return StartCommandResult.Sticky;
    }

    private void CreateNotificationChannel()
    {
        if (Build.VERSION.SdkInt >= BuildVersionCodes.O)
        {
            var channel = new NotificationChannel(CHANNEL_ID,
"Task Updates",
                NotificationImportance.Min)
            {
                Description = "Канал для оновлення завдань",
                  LockscreenVisibility =
NotificationVisibility.Secret,
```

```
            };
            channel.SetSound(null, null);
            var notificationManager =
(NotificationManager)GetSystemService(NotificationService);

notificationManager.CreateNotificationChannel(channel);
        }
    }
}
```

Б.4 Метод TaskUpdateWorker

```
public class TaskUpdateWorker : Worker
{
    public TaskUpdateWorker(Context context, WorkerParameters
workerParams) : base(context, workerParams) { }

    public override Result DoWork()
    {
        try
        {
            System.Diagnostics.Debug.WriteLine("TaskUpdateWorker
(Hourly): Starting work.");


            var dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment
.SpecialFolder.LocalApplicationData), "tasks.db3");
            var taskDatabase = new TaskDatabase(dbPath);
            var allTasks = taskDatabase.GetItemsAsync().Result;
            var savedTime =
Xamarin.Essentials.Preferences.Get("NotificationTime", "09:00");
            TimeSpan notificationTime =
TimeSpan.Parse(savedTime);

            DateTime now = DateTime.Now;

            DateTime nextRunTime =
DateTime.Today.Add(notificationTime);

            if (nextRunTime < now)
            {
                nextRunTime = nextRunTime.AddDays(1);
            }
            var tasksForCurrentHour = allTasks.Where(t =>
t.DueDate.Date >= now.Date && t.DueDate.Date <
now.Date.AddDays(1)).ToList();
            string message;
            if (!tasksForCurrentHour.Any())
            {
```

```
System.Diagnostics.Debug.WriteLine($"TaskUpdateWorker (Hourly):
No tasks for hour {now.Hour}. Work finished.");
                //return Result.InvokeSuccess();
                 message = "Завдання на поточний день немає";
            }

            else message = "Завдання на поточний день:\n- " +
string.Join("\n- ", tasksForCurrentHour.Select(t => t.Title));

System.Diagnostics.Debug.WriteLine($"TaskUpdateWorker (Hourly):
{message}.");
            System.Diagnostics.Debug.WriteLine($" Time of
not{nextRunTime}");
            var notification = new NotificationRequest
            {
                NotificationId = 1002,
                Title = "Щоденне нагадування",
                Description = message,
                Android = new AndroidOptions { IconSmallName =
new AndroidIcon("mainicon.png"), IconLargeName = new
AndroidIcon("mainicon.png") },
                Schedule = new NotificationRequestSchedule
                {
                    NotifyTime = nextRunTime
                }
            };


NotificationCenter.Current.Show(notification).Wait();

            System.Diagnostics.Debug.WriteLine("TaskUpdateWorker
(Hourly): Notification has been shown. Work finished
successfully.");
            return Result.InvokeSuccess();
        }
        catch (Exception ex)
        {

System.Diagnostics.Debug.WriteLine($"TaskUpdateWorker (Hourly):
An error occurred: {ex.Message}\n{ex.StackTrace}");
            return Result.InvokeFailure();
        }
    }
}
```