DailyApp

-Application de Gestion des Tâches Quotidiennes -

Simplifiez votre organisation au quotidien

ADRAR CELINE
Licence 3 ISEI
Developpement mobile


Caramel
Whisper Pink
Winter Wheat

DailyApp est conçue pour répondre au besoin croissant de gestion efficace des tâches.

Objectifs principaux :

- Faciliter la gestion des tâches quotidiennes
- Offrir une interface utilisateur intuitive et moderne
- Intégrer des fonctionnalités de géolocalisation
- Gérer les rappels et notifications
- Permettre le suivi de l'état des tâches

Fonctionnalités Principales :

- Création et gestion de tâches : Ajouter supprimer ou modifier des taches , ajouter des titres, descriptions, dates et heure.
- Notifications : Rappels automatiques .
- Géolocalisation : Ajout d'emplacements aux tâches.
- États des tâches : En attente, en cours, terminées.

Technologies utilisées :

- Langage : Kotlin
- Architecture : MVVM (Model-View-ViewModel)
- Base de données Room (SQLite)
- Interface utilisateur : Material Design Components
- Services : Google Maps, Google Places API
- Gestion des dépendances : Gradle avec KTS

Prérequis techniques :

- Android Studio (version récente
- JDK 11 ou supérieur
- SDK Android (API 34 minimum)
- Un émulateur Android ou un appareil physique
- Connexion Internet pour la synchronisation avec Google Maps

Structure du projet :

```
app/
├── src/
│   ├── main/
│   │   ├── java/com/example/dailyapp/
│   │   │   ├── data/
│   │   │   ├── ui/
│   │   │   ├── utils/
│   │   │   └── adapters/
│   │   ├── res/
│   │   └── AndroidManifest.xml
│   └── test/
└── build.gradle.kts
```

Architecture et Conception :

I Architecture MVVM :

L'application suit le pattern MVVM (Model-View-ViewModel)

- Model
  - data/model/
    - User.kt : Entité représentant un utilisateur
    - Task.kt : Entité représentant une tâche
    - TaskStatus.kt : Énumération des états possibles d'une tâche

- View
  - ui/
    - MainActivity : Écran principal avec liste des tâches
    - LoginActivity : Gestion de la connexion
    - RegisterActivity : Inscription des utilisateurs
    - AddEditTaskActivity : Ajout/modification des tâches
    - TaskDetailActivity : Détails d'une tâche

- ViewModel
  - ui/tasks/
    - TaskViewModel :
      - Gère la logique métier des tâches
      - Utilise Kotlin Coroutines pour les opérations asynchrones
      - Maintient l'état de l'UI via StateFlow

II Persistance des données :

-Room Database

```
@Database(entities = [User::class, Task::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    abstract fun taskDao(): TaskDao
}
```

-DAOs (Data Access Objects)

- UserDao : Gestion des utilisateurs
- TaskDao : Opérations CRUD sur les tâches
    - CRUD : Create, Read, Update, Delete

III Gestion des sessions :

Utilisation de SessionManager pour :

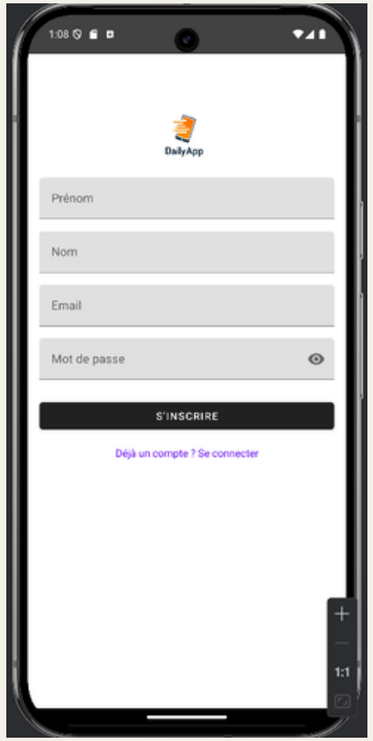
- Stockage sécurisé des informations de connexion (préférences partagées ou chiffrement).
- Gestion de l'état de connexion : détermine si un utilisateur est connecté ou déconnecté.
- Accès rapide aux informations utilisateur pour personnaliser les fonctionnalités et l'interface.

Dépendances principales :

```
dependencies {
    // Core Android
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.constraintlayout)

    // Architecture Components
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.ktx)
    implementation(libs.androidx.lifecycle.livedata.ktx)

    // Room Database
    implementation(libs.androidx.room.runtime)
    implementation(libs.androidx.room.ktx)
    ksp(libs.androidx.room.compiler)

    // Google Services
    implementation(libs.play.services.maps)
    implementation(libs.google.places)
}
```

Fonctionnalités Principales

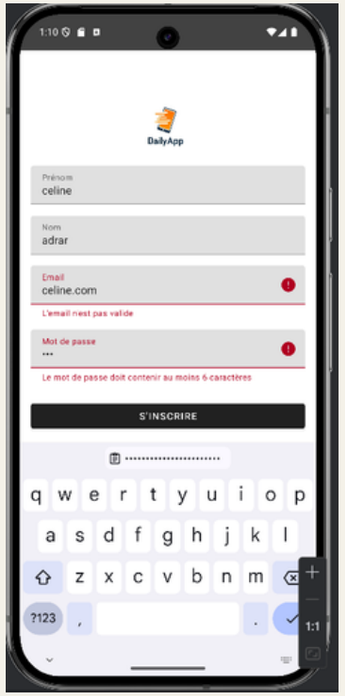
Inscription et connexion :



interface inscription

Il faut remplir toutes les informations demandées

```
// RegisterActivity.kt
class RegisterActivity {
    fun register(firstName: String, lastName: String, email:
String, password: String) {
        // Validation des entrées
        // Vérification de l'unicité de l'email
        // Création du compte utilisateur
    }
}
```



interface inscription

l'inscription ne peut pas s'effectuer en cas d'erreur

Fonctionnalités Principales

Inscription et connexion :

interface de connexion

Après avoir effectuer l'inscription un message s'affiche en disant : "inscription réussie", on peut désormais se connecter avec ses identifiants.

```
// LoginActivity.kt
class LoginActivity {
    fun login(email: String, password: String) {
        // Vérification des identifiants
        // Création de la session utilisateur
        // Redirection vers MainActivity
    }
}
```

interface de connexion :

//activity_login.xml

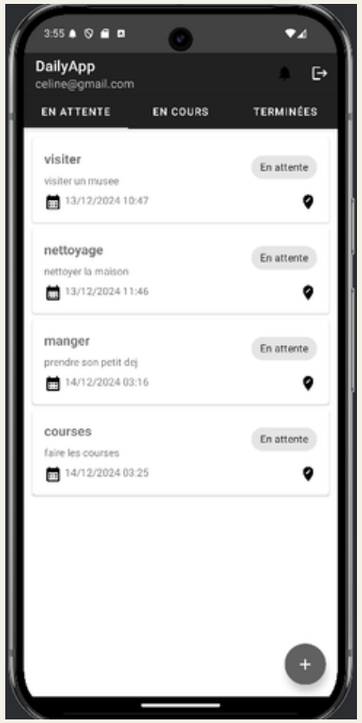
```
<androidx.constraintlayout.widget.ConstraintLayout>
    <!-- Logo -->
    <ImageView android:id="@+id/appLogo"/>

    <!-- Champs de saisie -->
    <TextInputLayout android:id="@+id/emailLayout"/>
    <TextInputLayout android:id="@+id/passwordLayout"/>

    <!-- Boutons -->
    <Button android:id="@+id/loginButton"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

interface de connexion

erreur : email ou mot de passe incorrecte

Fonctionnalités principales

Ecran principal :



écran principale

Ici on retrouve la liste des tâches.

//activity_main.xml

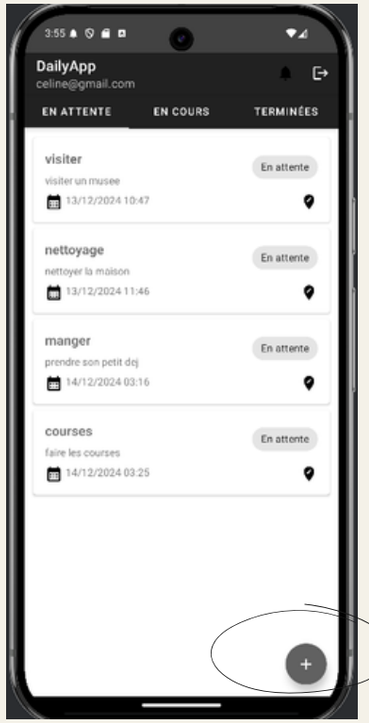
```
<CoordinatorLayout>
    <!-- Barre d'outils -->
    <AppBarLayout>
        <MaterialToolbar/>
        <TabLayout/>
    </AppBarLayout>

    <!-- Liste des tâches -->
    <RecyclerView android:id="@+id/tasksRecyclerView"/>

    <!-- Bouton d'ajout -->
    <FloatingActionButton android:id="@+id/addTaskFab"/>
</CoordinatorLayout>
```

Item de tâche (item_task.xml)

```
<MaterialCardView>
    <ConstraintLayout>
        <TextView android:id="@+id/taskTitle"/>
        <Chip android:id="@+id/taskStatus"/>
        <TextView android:id="@+id/taskDescription"/>
        <TextView android:id="@+id/taskDueDate"/>
    </ConstraintLayout>
</MaterialCardView>
```

thème et styles

```
<style name="Theme.DailyApp" parent="Theme.MaterialComponent
s.DayNight.NoActionBar">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryVariant">@color/primary_variant</
item>
    <item name="colorOnPrimary">@color/white</item>
    <item name="colorSecondary">@color/secondary</item>
</style>
```



Bouton ajout tâche

On clique sur le bouton + si on veut ajouter une tâche

Fonctionnalités principales

gestion des tâches :


```
data class Task(
    val id: Long = 0,
    val userId: Long,
    val title: String,
    val description: String,
    val dueDate: Date,
    val status: TaskStatus = TaskStatus.PENDING,
    val address: String? = null,
    val latitude: Double? = null,
    val longitude: Double? = null,
    val notified: Boolean = false
)
```


```
enum class TaskStatus {
    PENDING,        // En attente
    IN_PROGRESS,    // En cours
    COMPLETED       // Terminée
}
```

interface ajout de tâche

On ajoute la nouvelle tache avec toutes ses informations

état des la tâches

Filtrer les tâches par statut (En attente/En cours/Terminées)

Fonctionnalités principales

Ajout tâche :

interface ajout de tâche

On ajoute la nouvelle tache en remplissant les champs obligatoires (titre, description, date)

Ajout date

on ajoute une date limite pour la tache

Ajouter heure

on ajoute une heure à laquelle la tache doit être faite

ajout adresse

On peut ajouter une adresse si on le souhaite (optionnelle)

Fonctionnalités principales

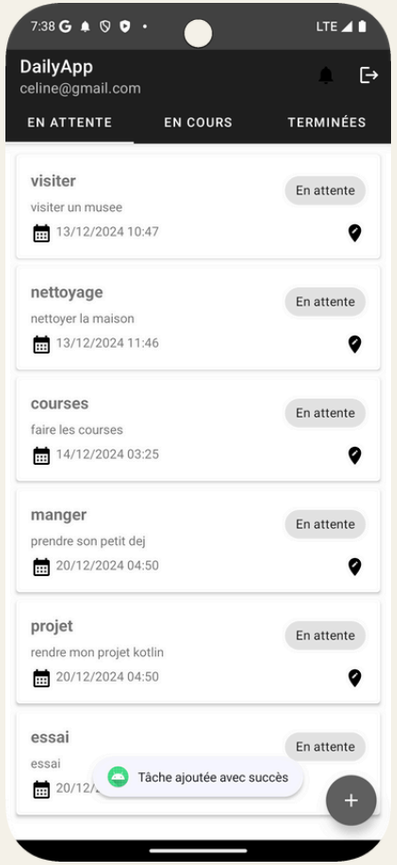
Ajout tâche :

ajout adresse

On peut ajouter une adresse si on le souhaite

Sélection d'adresse :

```
class AddEditTaskActivity {
    private fun setupLocationPicker() {
        val fields = listOf(
            Place.Field.ID,
            Place.Field.NAME,
            Place.Field.ADDRESS,
            Place.Field.LAT_LNG
        )

        val intent = Autocomplete.IntentBuilder(
            AutocompleteActivityMode.FULLSCREEN, fields)
            .build(this)
    }
}
```

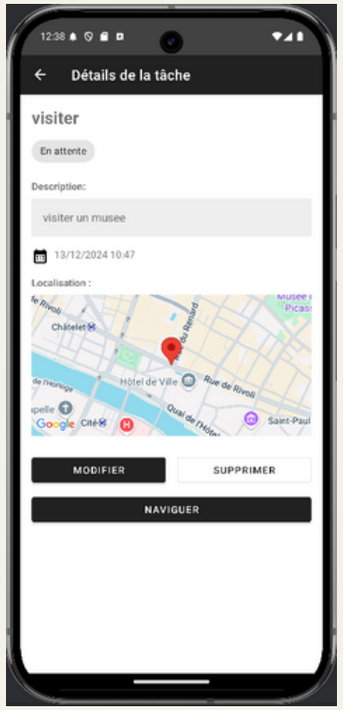


Enregistrer tâche

Un message disant :"tâche ajoutée avec succès " s'affiche

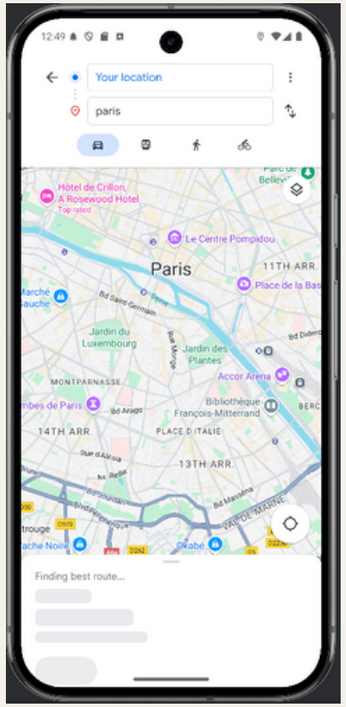
Fonctionnalités avancées

Modification des tâches :



Détail de la tâche

On peut modifier ou supprimer une tâche , ou même naviguer pour voir la carte (en cliquant dessus)



la navigation

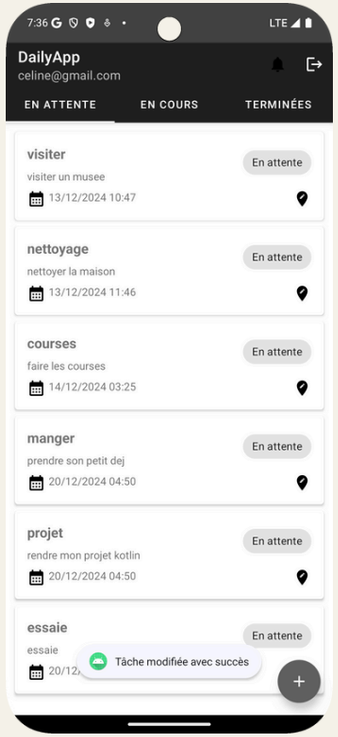
on peut voir la navigation en cliquant sur le bouton naviguer

Configurer la clé API Google Maps dans le manifest :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="VOTRE_CLE_API" />
```

configuration Maps :

```
class TaskDetailActivity {
    private fun setupMap(task: Task) {
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.mapView) as SupportMapFrag
ment

        mapFragment.getMapAsync { googleMap ->
            if (task.latitude != null && task.longitude != nu
ll) {
                val taskLocation = LatLng(task.latitude, tas
k.longitude)
                googleMap.addMarker(MarkerOptions().position
(taskLocation))
                googleMap.moveCamera(CameraUpdateFactory.newL
atLngZoom(taskLocation, 15f))
            }
        }
    }
}
```

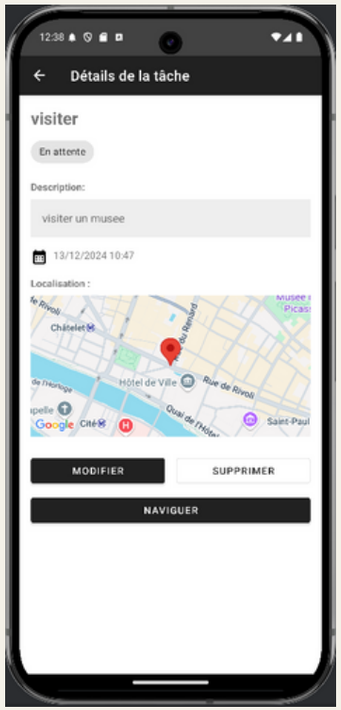


Modification tâche

Après la modification de la tache un message s’affiche en bas de l’écran pour dire qu’elle a été modifier avec succès

Fonctionnalités principales

Suppression tâche :



Détail de la tâche

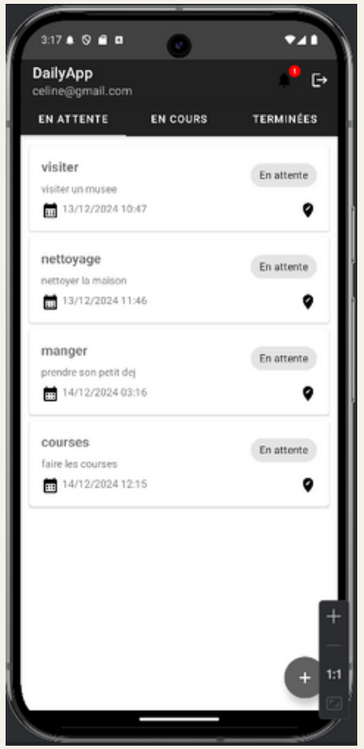
On peut modifier ou supprimer une tâche , ou même naviguer pour voir la carte

suppression

Quand on veut supprimer une tâche un message de comfirmation s’affiche

Fonctionnalités avancées

Affichage notifications :



notification

on reçoit une notification (rappel) avant d’atteindre l’heure limite d’une tâche quelconque

```
class NotificationService : Service() {
    private val taskRunnable = Runnable {
        checkUpcomingTasks()
        updateNotificationCount()
        handler?.postDelayed(taskRunnable, 60000) // Vérifica
tion toutes les minutes
    }

    private fun checkUpcomingTasks() {
        // Vérification des tâches à venir
        // Création de notifications si nécessaire
    }

    private fun showNotification(task: Task) {
        // Configuration du canal de notification
        // Création et affichage de la notification
    }
}
```

en cliquant sur l'icône (Badge de notification) dans la barre d'outils on peut voir l’historique des notifications disponibles

Sécurité et Gestion des Données

Gestion des sessions :

```
object SessionManager {
    private const val PREF_NAME = "DailyAppSession"
    private const val KEY_IS_LOGGED_IN = "isLoggedIn"
    private const val KEY_USER_ID = "userId"
    private const val KEY_USER_EMAIL = "userEmail"

    fun createSession(context: Context, userId: Long, email:
String) {
        // Stockage sécurisé des informations de session
    }

    fun clearSession(context: Context) {
        // Nettoyage des données de session
    }
}
```

Permissions Android :

```
<uses-permission android:name="android.permission.INTERNET" /
>
<uses-permission android:name="android.permission.ACCESS_NETW
ORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COAR
SE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE
_LOCATION" />
<uses-permission android:name="android.permission.POST_NOTIFI
CATIONS" />
```

Sécurité et Gestion des Données

Base de Données Room :

I Migration et Sauvegarde

```
Room.databaseBuilder(
    context.applicationContext,
    AppDatabase::class.java,
    "daily_app_database"
).build()
```

II Règles de Sécurité

```
@Entity(
    tableName = "users",
    indices = [Index(value = ["email"], unique = true)]
)
data class User(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    // ...
)

@Entity(
    foreignKeys = [
        ForeignKey(
            entity = User::class,
            parentColumns = ["id"],
            childColumns = ["userId"],
            onDelete = ForeignKey.CASCADE
        )
    ]
)
data class Task(
    // ...
)
```

Conclusion

"Pour conclure, cette application permet une gestion efficace des tâches quotidiennes grâce à ses fonctionnalités intuitives, telles que les notifications en temps réel et la catégorisation des tâches."

Points forts :

Ce projet a été une excellente opportunité pour découvrir et approfondir mes compétences en développement mobile. C'était ma première expérience avec le langage Kotlin, ce qui m'a permis de développer une application complète. J'ai également appris à intégrer des fonctionnalités avancées telles que les notifications en temps réel, la gestion des données utilisateurs, ainsi que la localisation pour enrichir l'expérience utilisateur.

Perspectives d'Amélioration :

À l'avenir, j'envisage d'ajouter une synchronisation avec le cloud pour permettre aux utilisateurs de sauvegarder et récupérer leurs données à partir de plusieurs appareils.

Ressources :

- Documentation Android : developer.android.com
- Tutoriels Kotlin : kotlinlang.org/docs
- Inspiration pour le design : Material Design Guidelines

Merci pour votre attention !