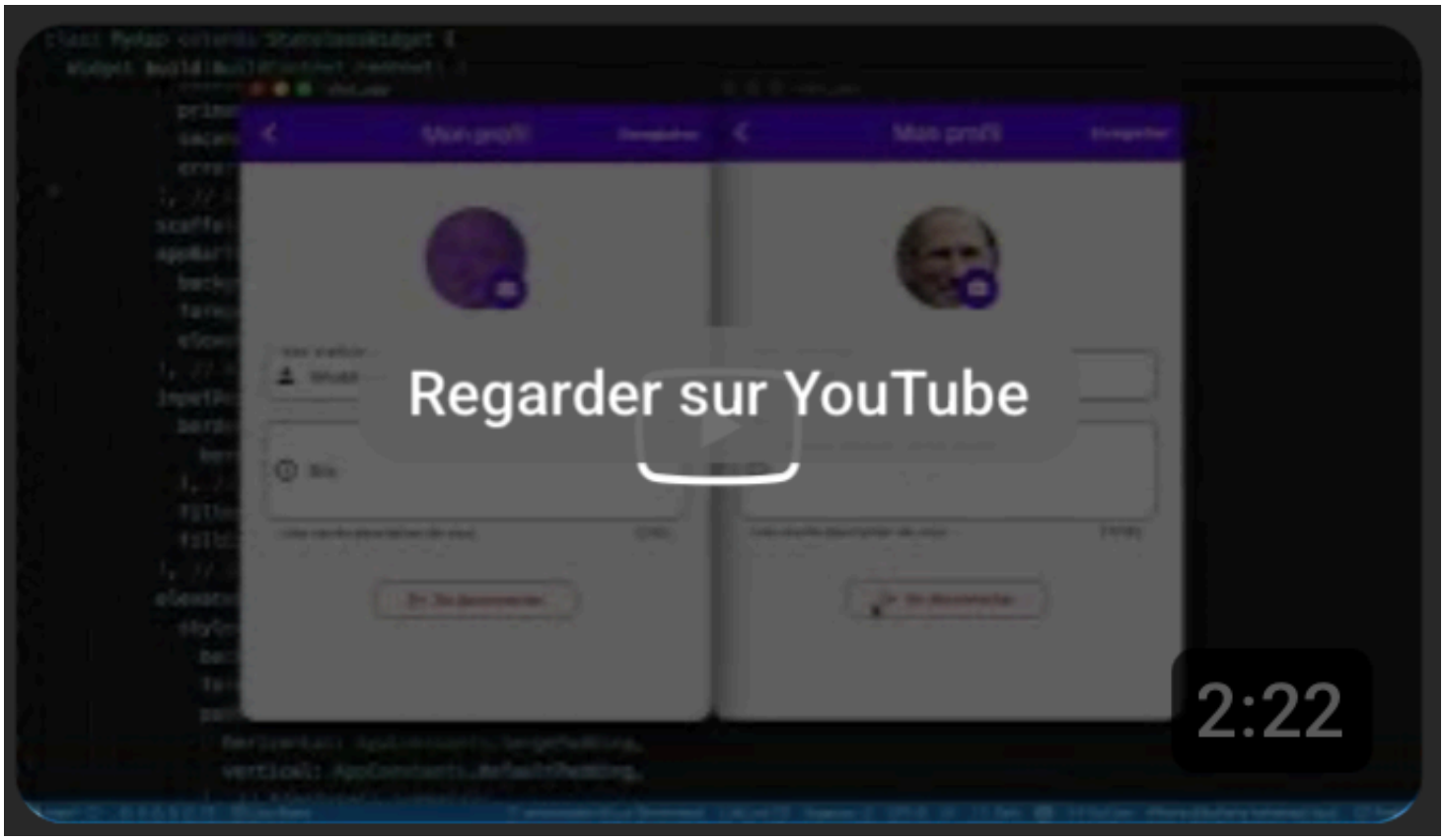
# Chat App - Application de messagerie Flutter




Lien vers vidéo de présentation :
https://youtu.be/AGlTNIfGhUk

> **Projet réalisé dans le cadre du cours Flutter & Firebase**
> Université de Corse - Master 2 DFS - 2025/2026
> Enseignant : Edouard Chevenslove

## 📱 Présentation

Chat App est une application de messagerie instantanée développée en Flutter, intégrant Firebase pour l'authentification et la gestion des données en temps réel. Le projet suit une architecture MVVM (Model-View-ViewModel) pour garantir une séparation claire des responsabilités et une maintenabilité optimale.

## ✨ Fonctionnalités implémentées

## 🔐 Authentification

- ✅ Inscription avec email/mot de passe
- ✅ Connexion sécurisée via Firebase Authentication
- ✅ Persistance de la session avec SharedPreferences
- ✅ Écran de bienvenue (Splash Screen)
- ✅ Déconnexion avec confirmation

## 💬 Messagerie

- ✅ Liste des utilisateurs en temps réel
- ✅ Conversations individuelles
- ✅ Envoi et réception de messages instantanés
- ✅ Messages groupés par date (Aujourd'hui, Hier, etc.)
- ✅ Interface chat moderne avec bulles de messages
- ✅ Horodatage des messages

## 👤 Profil utilisateur

- ✅ Modification du nom d'affichage
- ✅ Ajout/modification de la bio
- ✅ Upload de photo de profil
- ✅ **Stockage base64 des images** (car c'est payant)
- ✅ Avatars affichés dans toute l'application

## 🎨 Interface utilisateur

- ✅ Design Material Design 3
- ✅ Thème personnalisé (violet/cyan)
- ✅ Navigation fluide entre les écrans
- ✅ Feedback utilisateur (SnackBars, loading indicators)
- ✅ Responsive et adaptatif

## 🏗️ Architecture

### Structure MVVM

```
lib/
├── model/              # Modèles de données
│   ├── chat_user.dart  # Entité utilisateur
│   ├── chat.dart       # Entité conversation
│   └── message.dart    # Entité message
├── viewmodel/          # Logique métier
│   ├── auth_viewmodel.dart         # Authentification
│   ├── chat_user_viewmodel.dart    # Gestion utilisateurs
│   └── chat_viewmodel.dart         # Gestion conversations
├── pages/              # Écrans de l'application
│   ├── splash_page.dart
│   ├── login_page.dart
│   ├── signup_page.dart
│   ├── home_page.dart
│   ├── chat_page.dart
│   └── profile_page.dart
├── utils/              # Utilitaires
│   └── image_helper.dart  # Gestion images base64
├── constants.dart      # Constantes (couleurs, textes)
└── main.dart          # Point d'entrée
```

## Flux de données

```
View (UI) ↔ ViewModel (Logique) ↔ Model (Données) ↔ Firebase
     ↓              ↓                      ↓
  Widgets    ChangeNotifier          Firestore/Auth
```

# 🔧 Technologies utilisées

## Framework & Langage

- **Flutter 3.x** - Framework UI multiplateforme
- **Dart** - Langage de programmation

## Backend & Services

- **Firebase Authentication** - Authentification utilisateur
- **Cloud Firestore** - Base de données NoSQL temps réel
- **Firebase Core** - Configuration Firebase

## Gestion d'état & Architecture

- **Provider** - Injection de dépendances et gestion d'état
- **ChangeNotifier** - Pattern Observer pour la réactivité

## Packages additionnels

- `shared_preferences` - Stockage local de la session
- `image_picker` - Sélection d'images depuis la galerie
- `grouped_list` - Affichage des messages groupés par date
- `intl` - Formatage des dates

# 📊 Structure Firestore

## Collection `users`

```
users/{userId} {
  id: "uid",
  displayName: "Nom complet",
  email: "utilisateur@email.com",
  bio: "Description personnelle",
  avatarBase64: "..."  // Stockage base64
}
```

## Collection `chats`

```
chats/{chatId} {
  id: "user1_user2",
  participants: ["uid1", "uid2"],
  lastMessage: "Dernier message...",
  lastMessageTime: timestamp
}

chats/{chatId}/messages/{messageId} {
  from: "uid_expéditeur",
  to: "uid_destinataire",
  content: "Contenu du message",
  timestamp: timestamp
}
```

# 🔒 Règles de sécurité Firestore

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read: if request.auth != null;
      allow create: if request.auth != null && request.auth.uid == userId;
      allow update, delete: if request.auth != null && request.auth.uid == userId;
    }
    match /chats/{chatId} {
      allow read, write: if request.auth != null;
    }
    match /chats/{chatId}/messages/{messageId} {
      allow read, write: if request.auth != null;
    }
  }
}
```

# 🚀 Installation et configuration

## Prérequis

- Flutter SDK (≥ 3.0.0)
- Dart SDK (≥ 3.0.0)
- Compte Firebase
- Xcode (pour macOS) ou Android Studio (pour Android)

## Étapes d'installation

1. **Cloner le repository**

```
git clone https://github.com/antocreadev/M2DWM-tp-flutter.git
cd M2DWM-tp-flutter
```

2. **Installer les dépendances**

```
flutter pub get
```

3. **Configurer Firebase**

```
# Installer FlutterFire CLI
dart pub global activate flutterfire_cli

# Configurer Firebase (générera firebase_options.dart)
flutterfire configure
```

4. **Activer les services Firebase**

- Dans la Console Firebase :
  - **Authentication** → Activer Email/Password
  - **Firestore Database** → Créer la base de données
  - **Rules** → Copier les règles de sécurité ci-dessus

5. **Lancer l'application**

```
# macOS
flutter run -d macos

# iOS
flutter run -d ios

# Android
flutter run -d android
```

# 💡 Choix techniques importants

## Stockage base64 au lieu de Firebase Storage

**Décision** : Les images de profil sont stockées en base64 directement dans Firestore.

**Avantages** :

- C'est gratuit et évite de mettre la carte bancaire

**Limitations** :

- ⚠️ Taille limitée à ~500KB par image (compression automatique)
- ⚠️ Augmentation de la taille des documents Firestore

## Architecture MVVM avec Provider

**Motivation** : Séparation claire entre UI et logique métier, facilitant les tests et la maintenance.

**Implémentation** :

- `ViewModel` hérite de `ChangeNotifier`
- `Provider` injecte les ViewModels dans l'arbre de widgets
- `Consumer` écoute les changements et reconstruit l'UI

# 📸 Captures d'écran

## Authentification

- Écran de connexion avec validation
- Inscription avec création automatique du profil
- Splash screen avec logo

## Conversations

- Liste des utilisateurs disponibles
- Interface de chat moderne
- Messages groupés par date (Aujourd'hui, Hier)

## Profil

- Modification du nom et bio
- Upload de photo avec prévisualisation
- Confirmation de déconnexion

Ce projet est réalisé dans un cadre pédagogique pour le Master 2 DFS de l'Université de Corse.

# 👨‍💻 Auteur

**Anthony Menghi**
Master 2 DFS - Développement Full Stack
Université de Corse - 2025/2026