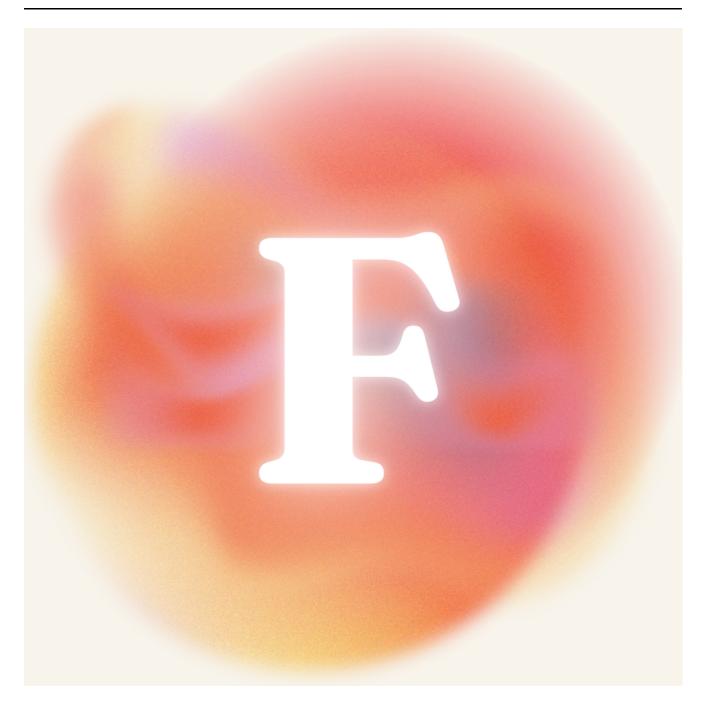
Fluently - Language Learning iOS App



A modern language-learning app built with **Swift** and **VIPER architecture**, designed to help users master new languages through interactive exercises, flashcards, and personalized lessons.

For Users: How to Install the App

Prerequisites

- An **iPhone** running iOS 17+
- A computer with **iMazing** installed (Download iMazing)

Installation Steps

PROFESSEUR: M.DA ROS

- 1. **Download the .ipa file** from this repository.
- 2. Connect your iPhone to your computer and trust the device.
- 3. Open iMazing and select your device.
- 4. Navigate to **Manage Apps** → **Library**.
- 5. Click the **triple dots (···)** in the bottom-right corner.
- 6. Select Install .ipa File and choose the downloaded Fluently.ipa.
- 7. Wait for the installation to complete, then open the app on your iPhone.

trust the developer certificate.



🗎 For Collaborators: Development Guide

Project Structure (VIPER Architecture)

```
Fluently/
AppComponents/
                      # Reusable UI components (viewModifiers,
themes, fonts)
 — Assets.xcassets/ # App icons and colors
 — Cache/
                      # Keychain and local storage
  — Models/
                      # Data models (Cards, Lessons, Exercises)
                      # API services and networking logic
  — Network/
  - Screens/
                      # VIPER modules for each screen
   ├── HomeScreen/ # Home screen (View, Presenter, Interactor,
Router, Builder)
    LessonScreens/ # Interactive exercises
      - LoginScreen/ # Authentication flow
                      # Other screens (Profile, Dictionary, Calendar,
etc)
└─ Tests/
                      # Unit and UI tests
```

Setup & Development

1. Clone the repository:

```
git clone https://github.com/your-repo/ios-app.git
```

- 2. Open Fluently.xcodeproj in Xcode 15+.
- 3. Install dependencies (if any) via Swift Package Manager.
- 4. Build and run on a simulator or physical device $(\mathbb{X} + \mathbb{R})$.

VIPER Workflow

- View: UI components (SwiftUI).
- Interactor: Business logic and data fetching.

- Presenter: Mediates between View and Interactor.
- Entity: Data models.
- Router: Navigation handling.

Example module:

```
Screens/LoginScreen/

LoginView.swift # UI

LoginPresenter.swift # Logic

LoginInteractor.swift # API calls

LoginRouter.swift # Navigation
```

Testing

PROFESSEUR: M.DA ROS

• Unit Tests: Run FluentlyTests (# + U).