

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM



BÀI TẬP LỚN

TÊN HỌC PHẦN: LẬP TRÌNH MOBILE

ĐỀ TÀI: XÂY DỰNG APP XEM LỊCH ÂM VÀ LỊCH DƯƠNG

Giáo viên hướng dẫn: Phạm Văn Tiệp

Sinh viên thực hiện:

STT	Mã Sinh Viên	Họ và tên	Lớp
1	1671020282	Nguyễn Văn Tân	CNT1602

Hà Nội, năm 2025

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM**



BÀI TẬP LỚN

TÊN HỌC PHẦN:

ĐỀ TÀI:

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bảng Số	Bảng Chữ
1	1671020282	Nguyễn Văn Tân	20/08/2004		

CÁN BỘ CHẤM THI 1

CÁN BỘ CHẤM THI 2

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ phát triển nhanh chóng, các ứng dụng di động đã trở thành công cụ không thể thiếu trong đời sống hàng ngày. Đặc biệt, với người Việt Nam, việc theo dõi lịch âm bên cạnh lịch dương là một nhu cầu thiết yếu để tổ chức các sự kiện quan trọng và giữ gìn truyền thống văn hóa. Đề tài "Xây dựng ứng dụng xem dương lịch và âm lịch" được thực hiện nhằm đáp ứng nhu cầu này, đồng thời là cơ hội để nhóm chúng tôi áp dụng kiến thức lập trình mobile đã học vào thực tế.

Báo cáo này là kết quả của quá trình nghiên cứu, thiết kế và phát triển ứng dụng trong khuôn khổ môn Lập trình Mobile. Chúng tôi đã nỗ lực xây dựng một ứng dụng tiện ích, kết hợp lịch dương, âm lịch và các tính năng như xem ngày lễ, lưu ngày quan trọng. Dù còn nhiều hạn chế, đây là sản phẩm tâm huyết, thể hiện sự học hỏi và sáng tạo của nhóm.

Chúng tôi xin gửi lời cảm ơn đến giảng viên đã hướng dẫn tận tình, cùng các bạn bè, đồng nghiệp đã hỗ trợ trong suốt quá trình thực hiện. Mong rằng báo cáo này sẽ nhận được sự đóng góp ý kiến để hoàn thiện hơn.

MỤC LỤC

LỜI NÓI ĐẦU.....	3
MỤC LỤC	4
CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI.....	6
1.1. Giới thiệu về đề tài.....	6
1.2. Mục tiêu của đề tài	7
1.3. Phạm vi của đề tài.....	8
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	10
2.1. Tổng quan về Dart	10
2.1.1. Ngôn ngữ Dart là gì?	10
2.1.2. Lịch sử và sự phát triển của Dart.....	10
2.1.3. Những tính năng nổi bật của ngôn ngữ lập trình Dart?.....	11
2.1.4. Ứng dụng của ngôn ngữ lập trình Dart.....	13
2.2. Giới thiệu về Flutter	15
2.2.1 Flutter là gì?.....	15
2.2.2. Ngôn ngữ lập trình của Flutter	15
2.2.3. Các thành phần chính của Flutter	16
2.2.4. Những tính năng của Flutter.....	17
2.3. Firebase.....	19
2.3.1. Firebase là gì?.....	19
2.3.2. Lịch sử các giai đoạn phát triển của Firebase	20
2.3.3. Cách thức hoạt động của Firebase.....	20
3.1. Ý tưởng thiết kế	23

3.2. Sơ đồ ứng dụng.....	24
3.3. Giao diện các trang chính	25
3.3.1. Màn hình chính (CalendarScreen):.....	25
3.3.2. Màn hình Dialog chọn tháng/năm	26
3.3.3. Màn hình Dialog danh sách ngày đã lưu	27
3.4. Code của một số trang chính	28
3.4.1. Main.dart	28
3.4.2 lunar_utils.dart.....	29
3.4.3 holiday_info.dart	33
3.4.4 calendar_screen.dart	34
KẾT LUẬN	36
DANH MỤC TÀI LIỆU THAM KHẢO	37

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu về đề tài

Trong đời sống văn hóa và xã hội của người Việt Nam, lịch âm (lunar calendar) đóng một vai trò không thể thiếu. Đây là hệ thống lịch truyền thống được sử dụng từ hàng ngàn năm trước để xác định các ngày lễ quan trọng như Tết Nguyên Đán (mùng 1 tháng 1 âm lịch), Rằm Tháng Giêng (15 tháng 1 âm lịch), Giỗ Tổ Hùng Vương (10 tháng 3 âm lịch), hay Tết Trung Thu (15 tháng 8 âm lịch). Những ngày này không chỉ mang ý nghĩa văn hóa mà còn là dịp để gia đình sum họp, tưởng nhớ tổ tiên, và thực hiện các nghi lễ truyền thống. Tuy nhiên, trong thời đại hiện nay, lịch dương (Gregorian Calendar) đã trở thành chuẩn mực quốc tế và được sử dụng rộng rãi trên các thiết bị công nghệ như điện thoại thông minh, máy tính bảng và máy tính cá nhân. Điều này dẫn đến sự bất tiện cho người dùng khi cần tra cứu ngày âm lịch để tổ chức các sự kiện quan trọng hoặc đơn giản là để hiểu rõ hơn về các ngày lễ truyền thống.

Đề tài "Xây dựng ứng dụng xem dương lịch và âm lịch" được thực hiện nhằm giải quyết vấn đề trên bằng cách cung cấp một giải pháp công nghệ hiện đại, tích hợp cả hai hệ thống lịch trong một giao diện trực quan và dễ sử dụng. Ứng dụng không chỉ dừng lại ở việc hiển thị ngày tháng mà còn cung cấp thông tin chi tiết về các ngày lễ lớn của Việt Nam, đồng thời cho phép người dùng lưu lại những ngày quan trọng của riêng họ (ví dụ: sinh nhật, kỷ niệm) và đồng bộ hóa dữ liệu qua Firebase Firestore. Với sự phát triển của công nghệ di động và xu hướng sử dụng ứng dụng đa nền tảng, ứng dụng được xây dựng bằng Flutter – một framework mạnh mẽ của Google – để đảm bảo khả năng chạy trên cả Android, iOS và web, đáp ứng nhu cầu đa dạng của người dùng.

Đề tài này không chỉ mang ý nghĩa thực tiễn mà còn có giá trị học thuật. Đây là cơ hội để nhóm thực hiện áp dụng kiến thức về lập trình mobile đã học trong môn học, từ việc phân

tích yêu cầu, thiết kế giao diện, lập trình logic, đến tích hợp cơ sở dữ liệu đám mây. Ngoài ra, quá trình thực hiện đề tài cũng giúp nhóm làm quen với các công cụ phát triển hiện đại như Dart, Flutter, và Firebase, đồng thời rèn luyện kỹ năng làm việc nhóm, quản lý thời gian và giải quyết vấn đề thực tế.

1.2. Mục tiêu của đề tài

Mục tiêu chính: Xây dựng một ứng dụng đa nền tảng cho phép người dùng tra cứu lịch dương và âm lịch, xem thông tin về các ngày lễ truyền thống, và quản lý các ngày quan trọng cá nhân với dữ liệu được lưu trữ trên đám mây. Ứng dụng hướng đến việc cung cấp một công cụ tiện lợi, chính xác và dễ sử dụng cho người Việt Nam trong việc theo dõi ngày tháng theo cả hai hệ thống lịch.

Mục tiêu cụ thể:

- Phát triển giao diện người dùng: Thiết kế một giao diện lịch dạng bảng (calendar) trực quan, hiển thị đồng thời ngày dương và ngày âm, với các thông tin bổ sung như ngày lễ được tô màu nổi bật.
- Tích hợp thuật toán chuyển đổi lịch âm: Xây dựng một hệ thống logic chính xác để chuyển đổi từ ngày dương sang ngày âm dựa trên múi giờ Việt Nam (GMT+7), đảm bảo khớp với lịch âm thực tế được sử dụng tại Việt Nam.
- Hiển thị thông tin ngày lễ: Cung cấp danh sách các ngày lễ quan trọng của Việt Nam (Tết Nguyên Đán, Giỗ Tổ Hùng Vương, Trung Thu, v.v.) dựa trên ngày âm, kèm theo mô tả ngắn gọn về ý nghĩa của từng ngày.
- Quản lý ngày quan trọng: Cho phép người dùng lưu trữ các ngày đặc biệt (ví dụ: sinh nhật, ngày cưới) và xóa chúng khi cần thiết, với dữ liệu được lưu trên Firebase Firestore để đồng bộ hóa giữa các thiết bị.
- Hỗ trợ đa nền tảng: Đảm bảo ứng dụng hoạt động mượt mà trên Android, iOS và web, tận dụng khả năng đa nền tảng của Flutter để giảm thời gian phát triển và tăng khả năng tiếp cận người dùng.

- Tối ưu trải nghiệm người dùng: Tích hợp các tính năng như chọn nhanh tháng/năm, hiển thị danh sách ngày đã lưu, và thông báo kết quả thao tác (lưu/xóa) qua SnackBar.

1.3. Phạm vi của đề tài

Phạm vi của đề tài được giới hạn để đảm bảo tính khả thi trong thời gian thực hiện bài tập lớn, đồng thời vẫn đáp ứng đầy đủ các yêu cầu về chức năng và công nghệ.

- Phạm vi chức năng:

- + Hiển thị lịch dương và âm lịch trong khoảng thời gian từ năm 1900 đến năm 2100, đủ để phục vụ nhu cầu tra cứu quá khứ và tương lai gần của người dùng.
- + Chuyển đổi từ ngày dương sang ngày âm dựa trên múi giờ GMT+7 (Việt Nam), với khả năng xử lý các năm nhuận trong lịch âm.
- + Cung cấp thông tin chi tiết về các ngày lễ lớn của Việt Nam, bao gồm tên lễ, ngày âm lịch, và ý nghĩa văn hóa (ví dụ: Tết Nguyên Đán, Lễ Vu Lan).
- + Cho phép người dùng lưu trữ và xóa các ngày quan trọng, với dữ liệu được quản lý trên Firebase Firestore
- + Hỗ trợ xem danh sách các ngày đã lưu trong một dialog, với khả năng xóa từng mục trực tiếp từ giao diện.

- Phạm vi công nghệ:

- + Sử dụng ngôn ngữ lập trình Dart và framework Flutter để phát triển ứng dụng, tận dụng các widget có sẵn như `TableCalendar` để hiển thị lịch.
- + Tích hợp Firebase Firestore làm cơ sở dữ liệu đám mây để lưu trữ và đồng bộ hóa dữ liệu.
- + Ứng dụng được thiết kế để triển khai trên ba nền tảng chính: Android, iOS và web, với trọng tâm kiểm thử trên web trong quá trình phát triển.

- Phạm vi thời gian: Đề tài được thực hiện trong khuôn khổ bài tập lớn môn Lập trình Mobile, bao gồm các giai đoạn:

- + Phân tích yêu cầu: Xác định các chức năng cần thiết và công nghệ phù hợp.
- + Thiết kế: Lên ý tưởng giao diện và sơ đồ ứng dụng.
- + Phát triển: Viết mã nguồn, tích hợp Firebase, và kiểm thử từng chức năng.
- + Hoàn thiện: Tối ưu hóa ứng dụng và viết báo cáo.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về Dart

2.1.1. Ngôn ngữ Dart là gì?

Ngôn ngữ Dart là một ngôn ngữ lập trình đa mục đích và hướng đối tượng do Google phát triển, được giới thiệu lần đầu tiên vào năm 2011. Dart có cú pháp tương tự ngôn ngữ lập trình C và được biết đến với tốc độ biên dịch nhanh, cú pháp dễ hiểu và khả năng phát triển giao diện người dùng linh hoạt. Sự linh hoạt của Dart được thể hiện qua khả năng xây dựng các ứng dụng web, di động (Android và iOS), máy chủ và máy tính để bàn, nhờ vào tích hợp với Flutter, một framework nổi tiếng của Google. Điều này mang lại nhiều lựa chọn cho các nhà phát triển trong việc tạo ra các ứng dụng hiện đại và hiệu quả.

2.1.2. Lịch sử và sự phát triển của Dart

Ngôn ngữ Dart, được phát triển bởi Google, bắt đầu từ năm 2009, với sự đóng góp chính từ hai kỹ sư Lars Bak và Kasper Lund. Họ là những người đứng sau việc tạo ra Dart, nhằm cung cấp một giải pháp thay thế hoặc bổ sung cho JavaScript trong việc phát triển các ứng dụng web quy mô lớn với mục tiêu tạo ra một ngôn ngữ lập trình web hiện đại, mạnh mẽ và linh hoạt hơn JavaScript.

Dart được giới thiệu lần đầu tiên tại hội nghị GOTO ở Aarhus, Đan Mạch vào tháng 10/2011. Mục tiêu ban đầu của Dart là thay thế JavaScript như một ngôn ngữ lập trình chính cho các ứng dụng web, mang lại một sự cải tiến về hiệu suất và khả năng mở rộng. Trong những năm đầu, Dart chủ yếu được tập trung vào phát triển các ứng dụng web, nhưng dần dần đã mở rộng ra nhiều lĩnh vực khác.

Đến tháng 14/11/2013, Dart SDK 1.0 chính thức được phát hành, với nhiều công cụ hỗ trợ mạnh mẽ bao gồm trình biên dịch Dart-to-JavaScript, Dart Virtual Machine (VM) và công cụ phát triển Dart Editor. Đồng thời, Google cũng giới thiệu Dartium, một phiên bản của trình duyệt Chromium có tích hợp sẵn Dart VM, cho phép chạy trực tiếp mã Dart mà không cần biên dịch sang JavaScript.

Nhưng đến năm 2015, Google quyết định không theo đuổi mục tiêu tích hợp Dart VM vào các trình duyệt chính thống mà tập trung vào việc cải thiện khả năng biên dịch của Dart sang JavaScript. Quyết định này đánh dấu một bước ngoặt quan trọng, định hình lại Dart như một

ngôn ngữ lập trình front-end với mã nguồn được biên dịch sang JavaScript để chạy trên các trình duyệt.

Năm 2017, Dart 2.0 được công bố với nhiều cải tiến quan trọng, nhấn mạnh vào việc phát triển ứng dụng di động. Cùng thời điểm này, Google giới thiệu Flutter, một bộ công cụ UI phát triển ứng dụng di động dựa trên Dart, giúp các nhà phát triển tạo ra các ứng dụng đẹp mắt và hiệu năng cao cho cả Android và iOS từ cùng một mã nguồn. Sự ra đời của Flutter đã đem lại sức sống mới cho Dart, khiến ngôn ngữ này trở nên phổ biến hơn trong cộng đồng phát triển ứng dụng.

Đến năm 2018, Flutter 1.0 được phát hành, mang lại sự chú ý lớn cho Dart, vì Dart là ngôn ngữ chính để phát triển các ứng dụng Flutter. Năm 2020, Dart 2.8 tiếp tục được phát hành với nhiều cải tiến về hiệu năng và công cụ, khẳng định sự cam kết của Google trong việc phát triển Dart và Flutter. Dart và Flutter ngày càng trở nên phổ biến trong cộng đồng phát triển ứng dụng, không chỉ cho di động mà còn cho các ứng dụng web và máy tính để bàn.

2.1.3. Những tính năng nổi bật của ngôn ngữ lập trình Dart?

a) Cú pháp đơn giản và dễ học

Dart có cú pháp tương tự như các ngôn ngữ lập trình C-style khác như Java và JavaScript, giúp lập trình viên dễ dàng tiếp cận và bắt đầu sử dụng Dart nhanh chóng. Điều này giúp giảm thời gian học tập và cho phép các lập trình viên nhanh chóng bắt đầu xây dựng ứng dụng.

b) Hiệu suất cao

Dart được thiết kế để đạt hiệu suất cao khi chạy trên mọi thiết bị và nền tảng. Điều này đạt được nhờ vào khả năng biên dịch Ahead-Of-Time (AOT) của Dart, một phương pháp cho phép biên dịch mã nguồn thành mã máy gốc trước khi ứng dụng được khởi chạy, giúp cải thiện đáng kể tốc độ và hiệu suất của ứng dụng. Điều này đặc biệt quan trọng đối với các ứng dụng di động và ứng dụng yêu cầu hiệu suất cao.

c) Hỗ trợ lập trình hướng đối tượng (OOP)

Dart là một ngôn ngữ lập trình hướng đối tượng (OOP) với đầy đủ các đặc điểm như class-based, tính kế thừa và đa hình. Điều này giúp các lập trình viên tổ chức mã nguồn một cách

rõ ràng và dễ quản lý. Lập trình hướng đối tượng cũng giúp tái sử dụng mã và phát triển các ứng dụng phức tạp hơn một cách hiệu quả.

d) Null Safety

Một trong những tính năng quan trọng và tiên tiến của Dart là null safety, giúp ngăn chặn các lỗi runtime liên quan đến null bằng cách đảm bảo rằng các biến không thể chứa giá trị null trừ khi được khai báo rõ ràng. Null safety không chỉ cải thiện độ an toàn của mã nguồn mà còn giúp các nhà phát triển tránh được những lỗi phổ biến và khó phát hiện. Điều này đặc biệt hữu ích trong các dự án lớn và phức tạp, nơi việc kiểm soát các giá trị null là rất quan trọng để đảm bảo chất lượng và hiệu suất của ứng dụng.

e). Khả năng biên dịch nhanh

Dart hỗ trợ cả hai phương pháp biên dịch là Ahead-Of-Time (AOT) và Just-In-Time (JIT), giúp tối ưu hóa quy trình phát triển ứng dụng. Khả năng biên dịch nhanh của Dart đóng vai trò quan trọng trong việc tăng tốc độ phát triển ứng dụng. Dart có thể biên dịch mã nguồn sang JavaScript, cho phép ứng dụng chạy mượt mà trên các trình duyệt web, đồng thời cũng có khả năng biên dịch sang mã máy gốc để hoạt động trên các thiết bị di động và máy tính để bàn. Việc biên dịch nhanh chóng không chỉ giảm thiểu thời gian chờ đợi của các nhà phát triển mà còn nâng cao hiệu suất làm việc, giúp quá trình phát triển diễn ra suôn sẻ và hiệu quả hơn.

f). Phát triển đa nền tảng

Dart, kết hợp với Flutter, cho phép phát triển ứng dụng đa nền tảng từ một cơ sở mã nguồn duy nhất. Các nhà phát triển có thể xây dựng ứng dụng cho iOS, Android, web và các ứng dụng desktop mà không cần viết mã lập trình riêng biệt cho từng nền tảng. Điều này giúp tiết kiệm thời gian và tài nguyên, đồng thời đảm bảo tính nhất quán và chất lượng của ứng dụng trên mọi nền tảng.

g). Tính năng hot reload

Một trong những tính năng nổi bật và được yêu thích nhất của Dart là hot reload. Khi sử dụng Flutter, Dart hỗ trợ tính năng hot reload, cho phép lập trình viên ngay lập tức thấy các thay đổi trong mã nguồn mà không cần phải khởi động lại ứng dụng, tăng tốc quá trình phát triển và gỡ lỗi. Dart cũng tích hợp tốt với các IDE phổ biến như IntelliJ IDEA, Android

Studio và Visual Studio Code, cung cấp các tiện ích như tự động hoàn thành mã, gỡ lỗi và phân tích mã nguồn. Với tất cả các tính năng này, Dart là một ngôn ngữ lập trình mạnh mẽ và linh hoạt, giúp lập trình viên phát triển các ứng dụng hiệu quả và dễ bảo trì trên nhiều nền tảng khác nhau.

h). Tích hợp tốt với các công cụ phát triển

Dart có khả năng tích hợp tốt với nhiều công cụ phát triển hiện đại ngày nay như Visual Studio Code, Android Studio và IntelliJ IDEA. Điều này giúp các lập trình viên dễ dàng viết code, debug và triển khai mã nguồn một cách hiệu quả. Các công cụ hỗ trợ này cũng cung cấp các tính năng như gợi ý câu lệnh code, kiểm tra lỗi và quản lý dự án, giúp tăng cường hiệu suất làm việc.

Những tính năng nổi bật này khiến Dart trở thành một ngôn ngữ mạnh mẽ và linh hoạt, phù hợp cho nhiều loại ứng dụng từ web, di động đến desktop, đáp ứng nhu cầu của các nhà phát triển trong môi trường công nghệ hiện đại.

2.1.4. Ứng dụng của ngôn ngữ lập trình Dart

Ngôn ngữ Dart được ứng dụng rộng rãi trong nhiều lĩnh vực phát triển phần mềm, đặc biệt là trong các ứng dụng di động, web và máy tính để bàn. Dưới đây là một số lĩnh vực mà Dart được sử dụng phổ biến và hiệu quả:

a) Phát triển ứng dụng di động

Dart trở nên đặc biệt nổi tiếng nhờ Flutter, framework UI do Google phát triển. Flutter cho phép các nhà phát triển xây dựng các ứng dụng di động đa nền tảng (cross-platform) với hiệu suất cao, giao diện người dùng đẹp mắt và trải nghiệm mượt mà. Bằng cách sử dụng một cơ sở mã nguồn duy nhất, các nhà phát triển có thể triển khai ứng dụng của mình trên cả Android và iOS, giúp tiết kiệm thời gian và công sức so với việc phải viết mã riêng cho từng nền tảng.

b). Phát triển ứng dụng web

Dart ban đầu được thiết kế để thay thế JavaScript trong phát triển web. Mặc dù không đạt được sự phổ biến như mong đợi ban đầu, Dart vẫn là một lựa chọn mạnh mẽ cho phát triển ứng dụng web. Với bộ công cụ như DartPad và các thư viện như AngularDart, các nhà phát

triển có thể xây dựng các ứng dụng web tương tác và hiệu quả. Dart cũng có khả năng biên dịch thành JavaScript, giúp tương thích với mọi trình duyệt hiện đại.

c). Phát triển ứng dụng desktop

Cùng với sự phát triển của Flutter, Dart không chỉ giới hạn ở các ứng dụng di động và web mà còn mở rộng sang phát triển ứng dụng máy tính để bàn. Flutter for Desktop cho phép các nhà phát triển tạo ra các ứng dụng chạy trên Windows, macOS và Linux từ cùng một cơ sở mã nguồn. Điều này giúp mở rộng phạm vi ứng dụng của Dart, cho phép các nhà phát triển tiếp cận một lượng lớn người dùng trên nhiều nền tảng.

d). Phát triển server-side

Dart không chỉ mạnh mẽ trong phát triển giao diện người dùng mà còn tỏ ra rất hiệu quả trong việc xây dựng các ứng dụng phía máy chủ. Thư viện `dart:io` và các gói mở rộng như `Aqueduct` giúp Dart xây dựng các ứng dụng máy chủ mạnh mẽ và hiệu quả. Dart có thể xử lý các yêu cầu HTTP, quản lý kết nối cơ sở dữ liệu và thực hiện các tác vụ phức tạp phía máy chủ, tương tự như Node.js nhưng tận dụng được các ưu điểm vượt trội của Dart.

e). Phát triển các công cụ

Dart còn được sử dụng để phát triển các công cụ và tiện ích hỗ trợ quá trình phát triển phần mềm. Ví dụ, các công cụ dòng lệnh (CLI) được viết bằng Dart có thể giúp tự động hóa các tác vụ, quản lý dự án và thực hiện các công việc khác một cách hiệu quả.



Hình 1

2.2. Giới thiệu về Flutter

2.2.1 Flutter là gì?

Là một framework mã nguồn mở dùng để phát triển ứng dụng di động và web đa nền tảng, được tạo ra và phát triển bởi Google. Flutter là công cụ cho phép các nhà phát triển tạo ra giao diện người dùng (UI) cho ứng dụng trên nhiều hệ điều hành và nền tảng chỉ bằng một mã nguồn duy nhất.

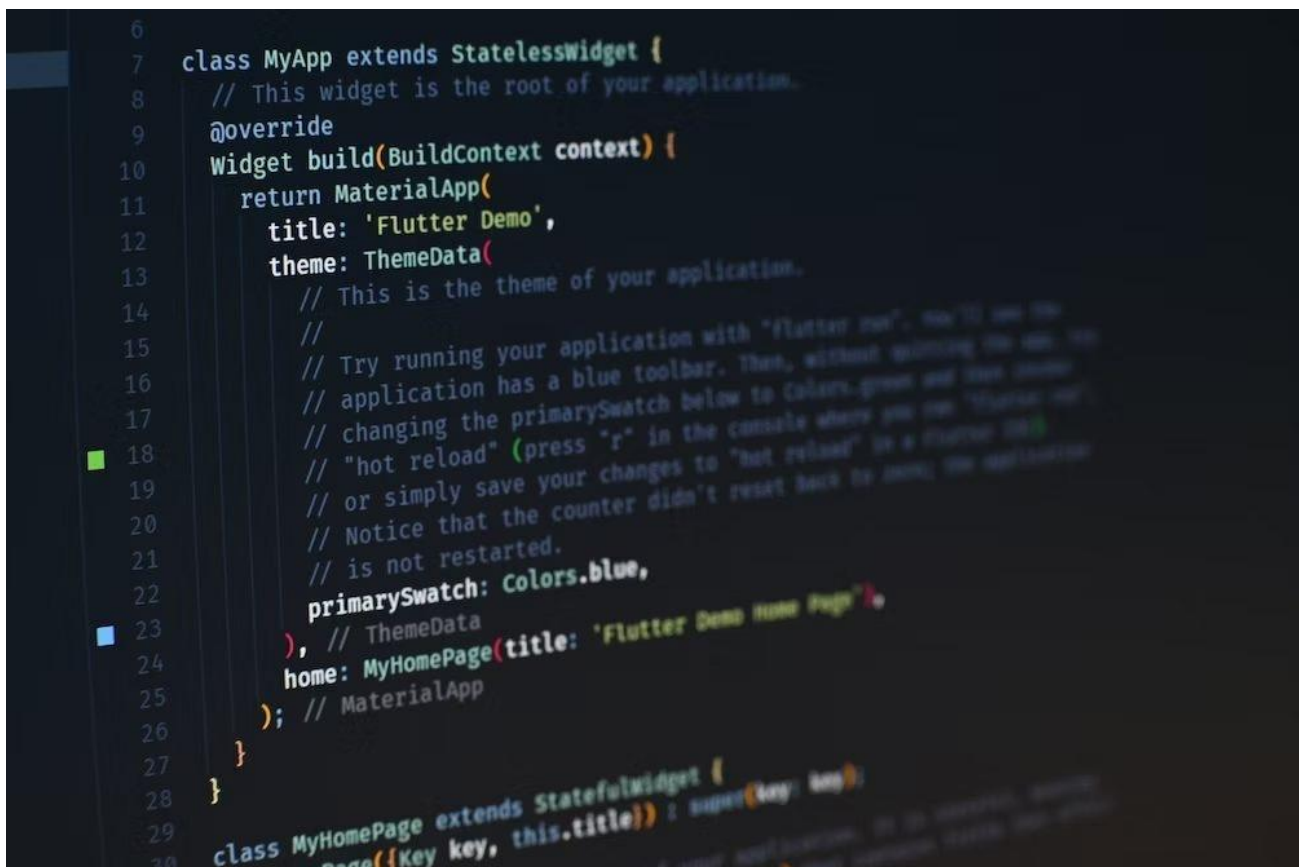


Hình 2

Ban đầu khi vừa mới ra mắt vào năm 2018, Flutter chủ yếu hỗ trợ phát triển ứng dụng di động. Tuy nhiên, từ đó đến nay, Flutter đã mở rộng phát triển ứng dụng trên sáu nền tảng khác nhau, bao gồm: iOS, Android, web, Windows, MacOS và Linux.

2.2.2. Ngôn ngữ lập trình của Flutter

Sau khi đã biết Flutter là gì, liệu bạn có thắc mắc Flutter sử dụng ngôn ngữ lập trình nào không? Ngôn ngữ lập trình Dart chính là ngôn ngữ được Flutter sử dụng để phát triển ứng dụng. Dart là một ngôn ngữ hiện đại, hướng đối tượng, được phát triển bởi Google, nổi bật với khả năng phát triển các ứng dụng web, di động một cách nhanh chóng và linh hoạt. Dart không chỉ là ngôn ngữ cơ sở mà còn là công cụ chính trong việc xây dựng framework Flutter.



```

6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Flutter Demo',
13      theme: ThemeData(
14        // This is the theme of your application.
15        //
16        // Try running your application with "flutter run". You'll see the
17        // application has a blue toolbar. Then, without quitting the app, try
18        // changing the primarySwatch below to Colors.green and then invoke
19        // "hot reload" (press "r" in the console where you ran "flutter run",
20        // or simply save your changes to "hot reload" in a flutter IDE).
21        // Notice that the counter didn't reset back to zero; the application
22        // is not restarted.
23        primarySwatch: Colors.blue,
24      ), // ThemeData
25      home: MyHomePage(title: 'Flutter Demo Home Page'), // MaterialApp
26    ); // MaterialApp
27  }
28 }
29
30 class MyHomePage extends StatefulWidget {
31   // ...
32 }

```

Hình 3

Khi sử dụng Dart trong Flutter, các nhà phát triển có thể tạo ra các ứng dụng di động đa nền tảng với giao diện đẹp, khả năng tùy chỉnh cao và hiệu suất ổn định. Điều này giúp tối ưu hóa quá trình phát triển, mang lại trải nghiệm người dùng tốt hơn trên nhiều loại thiết bị và hệ điều hành khác nhau.

2.2.3. Các thành phần chính của Flutter

Tiếp nối phần giải đáp "Flutter là gì" và ngôn ngữ mà công cụ này sử dụng sẽ là các thành phần chính của Flutter. Flutter có hai thành phần chính quan trọng như sau:

- **Framework (Thư viện giao diện người dùng dựa trên widgets):** Framework trong Flutter cung cấp một tập hợp các thành phần giao diện, cho phép người dùng tái sử dụng những mã code giữa các module và dự án khác nhau trên framework một cách dễ dàng, thuận tiện hơn. Điều này giúp tối ưu hóa quá trình phát triển ứng dụng, tiết kiệm thời gian và giúp tạo ra các ứng dụng có thể tùy chỉnh linh hoạt theo nhu cầu cụ thể của người dùng.

- **SDK (Bộ kit phát triển phần mềm):** SDK là bộ công cụ quan trọng hỗ trợ người dùng trong việc phát triển ứng dụng, bao gồm một loạt các công cụ như trình biên dịch, thư viện, và các tiện ích hỗ trợ khác. Nhờ vào SDK, các nhà phát triển có thể tạo ra ứng dụng trên nhiều nền tảng khác nhau như iOS, Android, web, Windows, MacOS và Linux. SDK cung cấp các công cụ để biên dịch mã nguồn thành mã gốc (native code) tương ứng với từng hệ điều hành, giúp ứng dụng hoạt động mượt mà và hiệu quả.



Hình 4

2.2.4. Những tính năng của Flutter

- Dễ sử dụng: Flutter sử dụng ngôn ngữ lập trình Dart, một ngôn ngữ đơn giản, linh hoạt và dễ tiếp cận. Các nhà phát triển rất dễ dàng để học và sử dụng Dart để xây dựng ứng dụng.
- Tính năng Hot Reload: Hot Reload trong Flutter cho phép nhà phát triển xem kết quả ngay lập tức sau khi thực hiện các thay đổi trong mã nguồn, giúp tối ưu hóa quá trình phát triển bằng cách cung cấp khả năng chỉnh sửa, thử nghiệm và sửa lỗi nhanh chóng.

- Các widget tích hợp sẵn: Flutter cung cấp các widget tích hợp sẵn với thiết kế đẹp mắt và phong phú, giúp tạo ra giao diện người dùng mượt mà, tự nhiên. Những widget này hỗ trợ việc xây dựng giao diện đa dạng và tương thích trên nhiều nền tảng.



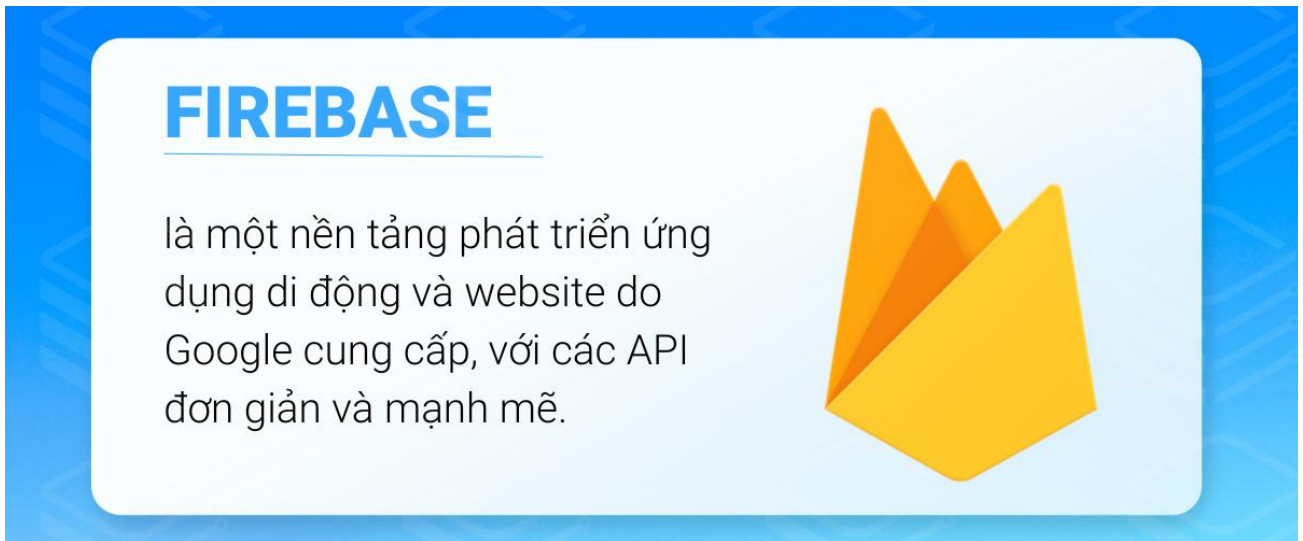
Hình 5

- Hiệu năng cao: Flutter tập trung vào việc cải thiện hiệu năng của ứng dụng, giúp ứng dụng không bị lag lúc sử dụng, ngay cả trên các thiết bị có cấu hình thấp.
- Thể hiện cùng một UI trên nhiều nền tảng: Với Flutter, bạn có thể xây dựng một giao diện người dùng đồng nhất và triển khai nó trên nhiều nền tảng khác nhau như iOS, Android, web hay desktop. Điều này giúp tiết kiệm thời gian và tài nguyên phát triển.
- Giải quyết thách thức trong giao diện người dùng: Flutter cung cấp cách tiếp cận linh hoạt để giải quyết các thách thức trong việc thiết kế giao diện người dùng, thông qua việc sử dụng các layout, platform và widget phong phú.

2.3. Firebase

2.3.1. Firebase là gì?

Firebase là một nền tảng phát triển ứng dụng di động và website do Google cung cấp. Nền tảng này cung cấp bộ công cụ toàn diện với các API đơn giản và mạnh mẽ, giúp các nhà phát triển xây dựng ứng dụng mà không cần lo lắng về việc quản lý server hay cơ sở hạ tầng backend phía sau.



Hình 6

Firebase không chỉ giúp tiết kiệm thời gian triển khai ứng dụng mà còn hỗ trợ mở rộng quy mô một cách dễ dàng. Nền tảng này cung cấp dịch vụ cơ sở dữ liệu đám mây, được vận hành trên hệ thống máy chủ mạnh mẽ của Google, giúp việc lập trình trở nên đơn giản hơn nhờ tự động hóa nhiều thao tác liên quan đến cơ sở dữ liệu.

Firebase còn cung cấp các giao diện lập trình ứng dụng (API) thân thiện, giúp thu hút nhiều người dùng hơn và tăng khả năng sinh lời cho các nhà phát triển. Đặc biệt, Firebase đảm bảo tính đa năng và bảo mật cao, hỗ trợ cả hai hệ điều hành phổ biến là Android và iOS.

Chính vì những lý do này, Firebase trở thành lựa chọn hàng đầu cho nhiều lập trình viên trên toàn thế giới trong việc phát triển các ứng dụng dành cho hàng triệu người dùng.

2.3.2. Lịch sử các giai đoạn phát triển của Firebase

- Hơn 10 năm trước, Firebase xuất hiện với tên gọi ban đầu là Envolv. Lúc này, Envolv chỉ là một nền tảng cung cấp các API để tích hợp tính năng chat vào website.
- Không chỉ được dùng cho ứng dụng nhắn tin trực tuyến, nền tảng này còn được nhiều người dùng cho hoạt động truyền và đồng bộ hóa dữ liệu trên các ứng dụng khác như game online. Nhận thấy tiềm năng này, các nhà sáng lập của Envolv quyết định phân tách 2 hệ thống nhắn tin trực tuyến và hệ thống đồng bộ dữ liệu thời gian thực.
- Năm 2012, dựa trên những nền móng có sẵn, Firebase chính thức ra mắt với vai trò là nhà cung cấp dịch vụ Backend-as-a-Service. 2 năm sau đó (2014), Google đã mua lại Firebase và biến nền tảng này thành một dịch vụ đa năng, phục vụ hàng triệu người dùng trên toàn thế giới cho đến ngày nay.

2.3.3. Cách thức hoạt động của Firebase

Sau khi được Google mua lại và phát triển, Firebase hiện đang cung cấp nhiều tính năng, trong đó nổi bật có thể kể đến:

a) Firebase Realtime Database

Firestore Realtime Database là một tính năng, cho phép bạn sử dụng một cơ sở dữ liệu thời gian thực ngay sau đăng ký tài khoản và tạo ứng dụng. Dữ liệu trong database này được trình bày dưới dạng JSON và được đồng bộ thời gian đến tất cả các client kết nối.



Hình 7

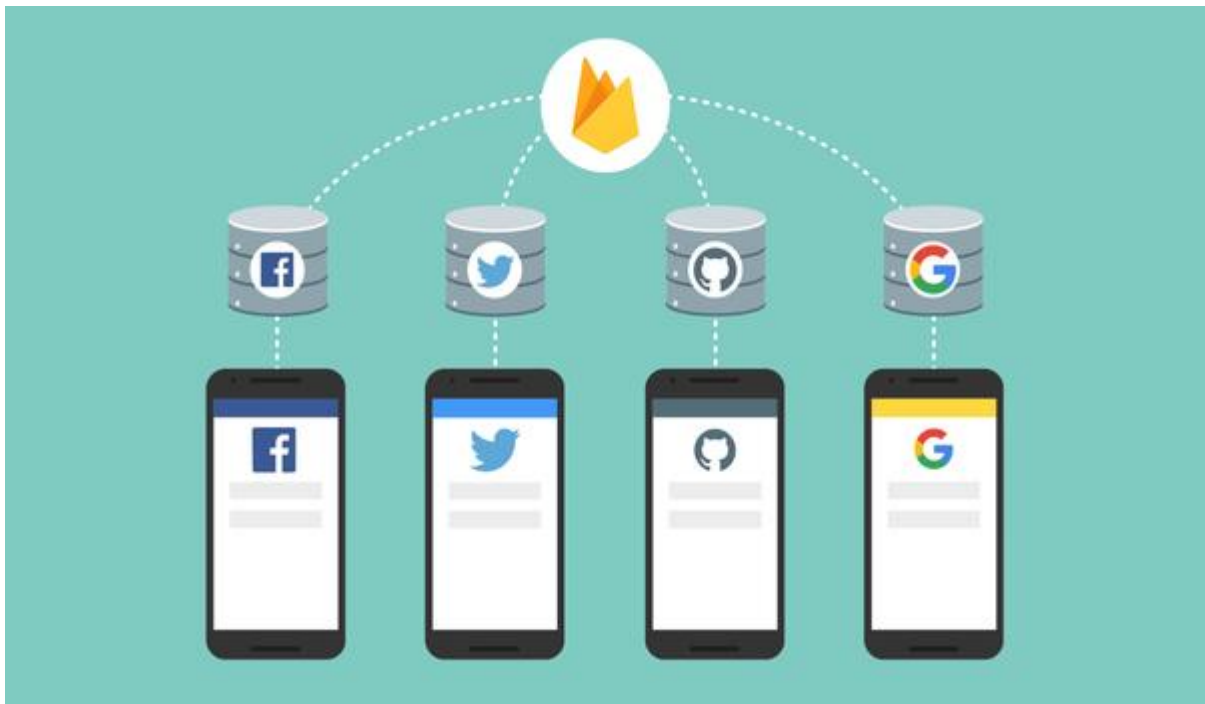
Điều này rất hữu ích cho các ứng dụng hoạt động trên nhiều nền tảng, vì tất cả các client,, đều truy cập cùng một cơ sở dữ liệu. Bất cứ khi nào nhà phát triển thực hiện thay đổi, dữ liệu trong database sẽ được tự động cập nhật và phản ánh trên tất cả các thiết bị. Tất cả thông tin này được truyền đi an toàn qua kết nối SSL với chứng chỉ mã hóa 2048 bit.

Nếu gặp phải trường hợp mất kết nối mạng, dữ liệu sẽ được lưu trữ tạm thời trên thiết bị của bạn (local). Khi mạng được khôi phục, những thay đổi này sẽ tự động đồng bộ lên server của Firebase. Nếu dữ liệu trên thiết bị cũ hơn so với trên server, hệ thống cũng sẽ tự động cập nhật để đảm bảo bạn luôn nhận thông tin mới nhất.

b) Firebase Authentication

Firebase Authentication một trong những tính năng nổi bật của Firebase là hệ thống xác thực người dùng, hỗ trợ nhiều phương thức đăng nhập khác nhau như Email, Facebook, Twitter, GitHub, và Google.

Firebase còn cung cấp khả năng xác thực ẩn danh, rất hữu ích cho các ứng dụng muốn cho phép người dùng trải nghiệm mà không cần đăng nhập ngay lập tức. Tính năng xác thực của Freebase giúp bảo vệ thông tin cá nhân của người dùng, đảm bảo an toàn và ngăn ngừa rủi ro bị đánh cắp tài khoản.



Hình 8

c) Firebase Hosting

Một tính năng quan trọng khác không thể không nhắc đến khi sử dụng Firebase là dịch vụ hosting. Firebase cung cấp các hosting này thông qua một mạng CDN và bảo mật bằng chuẩn SSL.

CDN (Content Delivery Network), là một mạng lưới các máy chủ phân bố rộng khắp toàn cầu. Mỗi máy chủ đều lưu trữ các bản sao của nội dung tĩnh trên website và phân phối chúng đến các PoP (Points of Presence) gần người dùng cuối nhất. Khi một người dùng truy cập vào website, nội dung sẽ được tải từ máy chủ CDN gần họ nhất, giúp tăng tốc độ tải trang và giảm độ trễ.

Nhờ vào dịch vụ hosting trên nền tảng Firebase, các lập trình viên có thể tiết kiệm thời gian trong việc thiết kế, xây dựng và phát triển ứng dụng, bởi vì họ không cần phải lo lắng về cơ sở hạ tầng phân phối nội dung.



Firebase Hosting

Hình 9

CHƯƠNG 3. THIẾT KẾ ỨNG DỤNG

3.1. Ý tưởng thiết kế

Ứng dụng được xây dựng với mục tiêu giúp người dùng dễ dàng xem lịch dương và âm lịch cùng một lúc, đồng thời biết được các ngày lễ quan trọng của Việt Nam. Ý tưởng chính là tạo ra một giao diện lịch dạng bảng, nơi mỗi ngày sẽ hiển thị cả ngày dương và ngày âm, kèm theo thông tin về lễ nếu có. Người dùng cũng có thể lưu lại những ngày đặc biệt của riêng mình và xem lại bất cứ lúc nào.

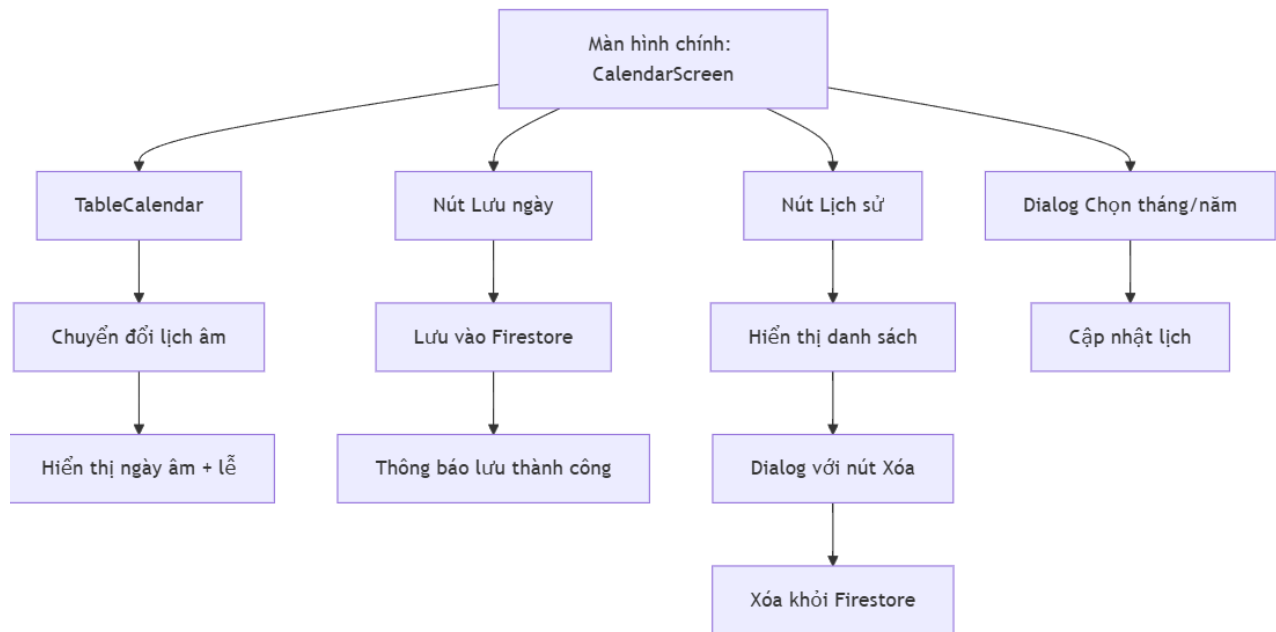
Về phần chuyển đổi lịch âm, đây là một phần quan trọng của ứng dụng. Lịch âm Việt Nam dựa vào chu kỳ của mặt trăng và có thêm tháng nhuận để khớp với năm dương. Để chuyển từ ngày dương sang ngày âm, ứng dụng dùng một số bước đơn giản:

- Đầu tiên, tính số ngày kể từ một mốc thời gian cố định (gọi là ngày Julian) dựa trên ngày, tháng, năm dương.
- Sau đó, tìm ngày trăng mới gần nhất để xác định ngày đầu tháng âm. Từ đây, tính ra ngày âm bằng cách đếm số ngày chênh lệch.
- Tiếp theo, xác định tháng âm và năm âm bằng cách so sánh với các mốc thời gian quan trọng trong năm âm (như ngày 11 tháng 11 âm).
- Cuối cùng, kiểm tra xem có tháng nhuận hay không để điều chỉnh cho đúng.

Phần này được viết trong file `lunar_utils.dart`. Ví dụ, ngày 22/03/2025 sẽ được đổi thành ngày 2/2/2025 âm lịch. Sau khi có ngày âm, ứng dụng sẽ kiểm tra xem ngày đó có phải là ngày lễ không (như Tết Nguyên Đán mừng 1/1 âm) và hiển thị thông tin tương ứng.

Ngoài ra, ứng dụng còn có các ý tưởng khác như: dùng bảng lịch để hiển thị ngày, lưu ngày quan trọng lên Firebase để đồng bộ giữa các thiết bị, và thêm nút để người dùng chọn tháng/năm muốn xem.

3.2. Sơ đồ ứng dụng



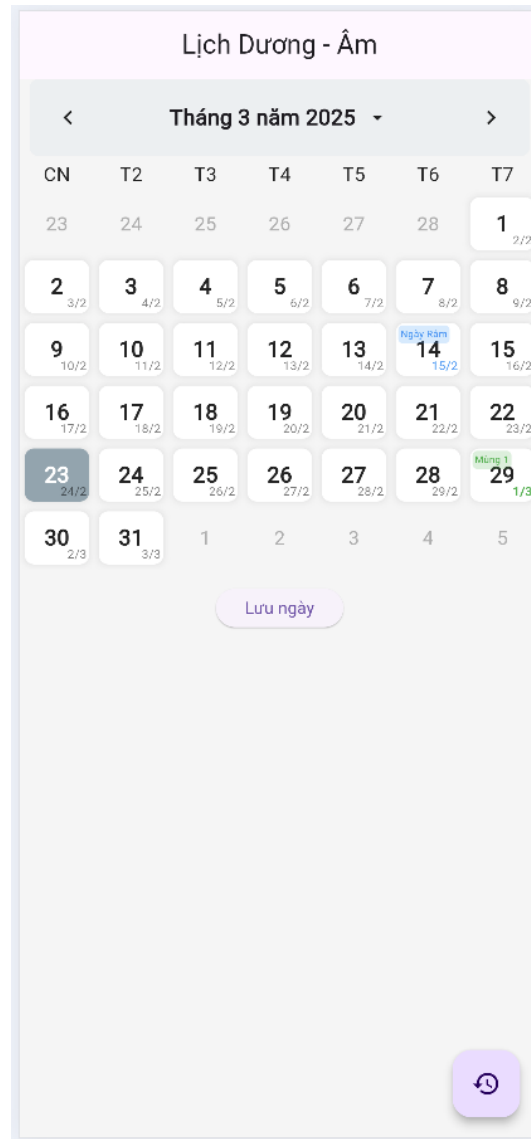
Hình 10: Sơ đồ ứng dụng.

Cách ứng dụng hoạt động được mô tả qua sơ đồ như sau:

- Người dùng mở ứng dụng và vào màn hình chính (CalendarScreen).
- Từ đây, họ có thể:
 - + Xem lịch dương và âm trên bảng lịch.
 - + Nhấn nút "Lưu ngày" để lưu ngày đang chọn lên Firebase.
 - + Nhấn nút "Lịch sử" để xem danh sách ngày đã lưu và xóa nếu muốn.
 - + Mở dialog để chọn tháng và năm cần xem.
- Mọi dữ liệu ngày quan trọng được lưu trên Firebase trong một danh sách gọi là saved_dates. Khi người dùng lưu hoặc xóa ngày, ứng dụng sẽ cập nhật danh sách này và thông báo kết quả.

3.3. Giao diện các trang chính

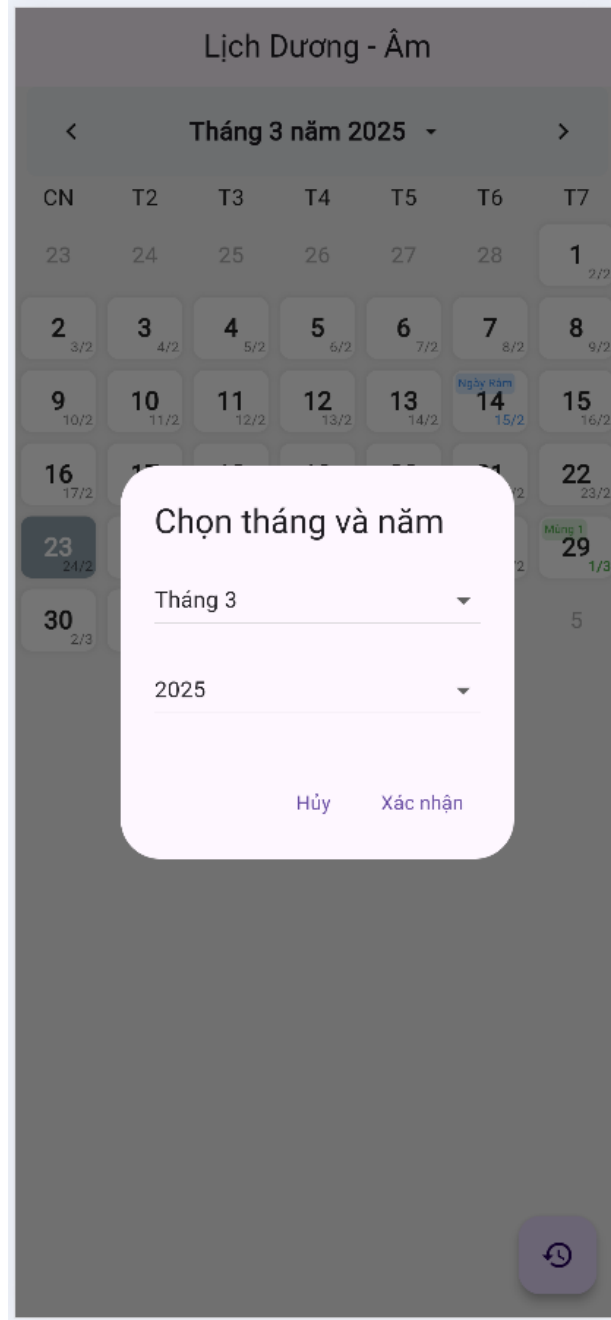
3.3.1. Màn hình chính (CalendarScreen):



Hình 11: Màn hình chính (CalendarScreen)

- Đây là màn hình chính, có bảng lịch hiển thị các ngày trong tháng. Mỗi ô ngày cho thấy ngày dương, ngày âm (góc dưới), và tên lễ (góc trên) nếu có.
- Dưới bảng lịch là nút "Lưu ngày" để thêm ngày đang chọn vào danh sách.
- Góc dưới bên phải có nút hình tròn (FloatingActionButton) để mở danh sách ngày đã lưu.
- Ngày hiện tại có viền xanh, ngày đã lưu có nền vàng, ngày được chọn có nền xám đậm.

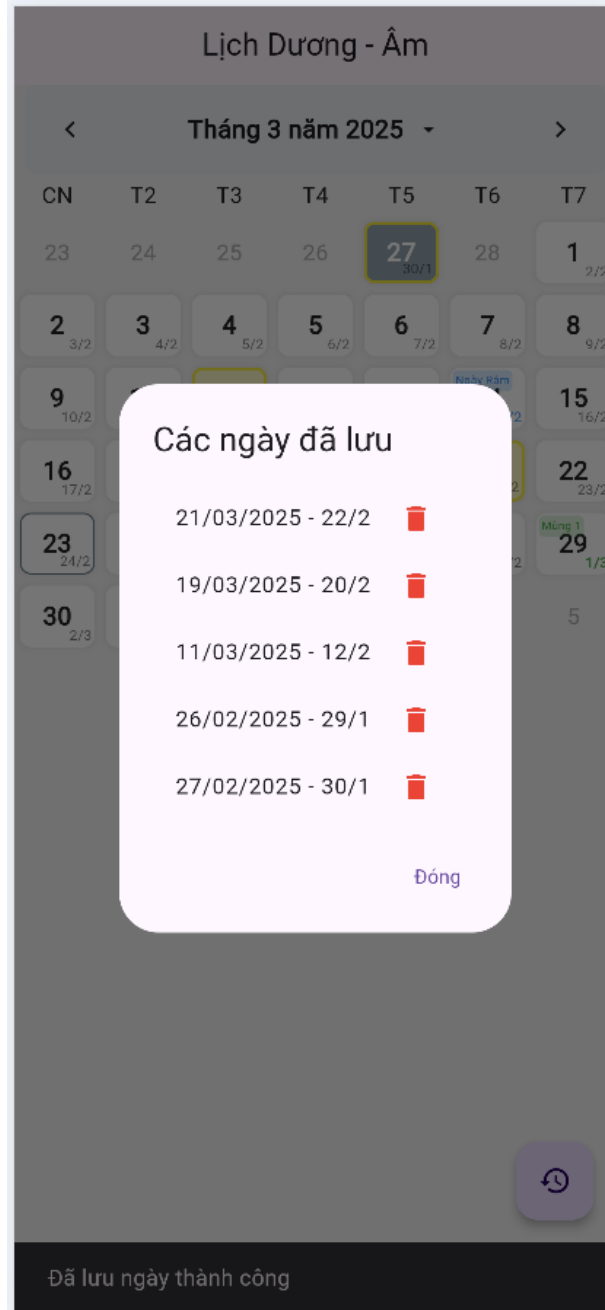
3.3.2. Màn hình Dialog chọn tháng/năm



Hình 12: Màn hình Dialog chọn tháng/năm

- Khi nhấn vào tiêu đề tháng/năm trên lịch, một hộp thoại hiện ra.
- Trong đó có hai danh sách thả xuống: một để chọn tháng (1-12), một để chọn năm (1900-2100).
- Có hai nút: "Hủy" để đóng, "Xác nhận" để đổi lịch sang tháng/năm mới

3.3.3. Màn hình Dialog danh sách ngày đã lưu



Hình 13: Màn hình Dialog danh sách ngày đã lưu

- Mở ra khi nhấn nút "Lịch sử", hiển thị danh sách các ngày đã lưu.
- Mỗi dòng cho thấy ngày dương, ngày âm, và tên lễ (nếu có), kèm nút "Xóa" màu đỏ ở bên phải.
- Dưới cùng có nút "Đóng" để thoát.

3.4. Code của một số trang chính

3.4.1. Main.dart

Đây là file chính khởi chạy ứng dụng Flutter và cấu hình Firebase.



```

1  import 'package:flutter/material.dart';
2  import 'package:firebase_core/firebase_core.dart';
3  import 'calendar_screen.dart';
4  // ignore: depend_on_referenced_packages
5  import 'package:flutter_web_plugins/flutter_web_plugins.dart';
6
7  void main() async {
8    WidgetsFlutterBinding.ensureInitialized();
9
10   // Cấu hình Firebase cho tất cả các nền tảng
11   await Firebase.initializeApp(
12     options: const FirebaseOptions(
13       apiKey: "AIzaSyC410n9giZS44JK1kg1XGUpR8XNzCTPc3g",
14       authDomain: "lunarcalendarweb.firebaseio.com",
15       projectId: "lunarcalendarweb",
16       storageBucket: "lunarcalendarweb.firebaseio.com",
17       messagingSenderId: "745386022738",
18       appId: "1:745386022738:web:df4a7acc19e7f35afc4c8b",
19       measurementId: "G-F5NJBNG636",
20     ),
21   );
22
23   // Loại bỏ ký hiệu # trong URL cho web
24   setUrlStrategy(PathUrlStrategy());
25
26   runApp(const MyApp());
27 }
28
29 class MyApp extends StatelessWidget {
30   const MyApp({super.key});
31
32   @override
33   Widget build(BuildContext context) {
34     return MaterialApp(
35       debugShowCheckedModeBanner: false,
36       theme: ThemeData(
37         primarySwatch: Colors.blueGrey,
38         scaffoldBackgroundColor: Colors.grey[100],
39         textTheme: const TextTheme(bodyMedium: TextStyle(fontSize: 16)),
40       ),
41       home: const CalendarScreen(),
42     );
43   }
44 }
45

```

Hình 14: Main.dart

- Khởi tạo ứng dụng:

- + `WidgetsFlutterBinding.ensureInitialized()` đảm bảo Flutter được khởi tạo trước khi thực hiện các tác vụ bất đồng bộ.

- + `Firebase.initializeApp()` khởi tạo Firebase với các thông số cấu hình như `apiKey`, `projectId`, `appId`, v.v. Đây là bước cần thiết để ứng dụng sử dụng Firestore (lưu trữ ngày đã chọn).
- **Cấu hình cho web:** `setUrlStrategy(PathUrlStrategy())` loại bỏ ký hiệu # trong URL khi ứng dụng chạy trên web, cải thiện trải nghiệm người dùng.
- **Chạy ứng dụng:** `runApp(const MyApp())` khởi chạy widget chính `MyApp`.
- **Widget MyApp:**
 - + Đây là widget gốc của ứng dụng, sử dụng `MaterialApp` để thiết lập giao diện cơ bản.
 - + `debugShowCheckedModeBanner: false` tắt biểu ngữ debug.
 - + `theme` thiết lập giao diện chung với màu chủ đạo là `blueGrey` và nền `grey[100]`.
 - + `home: const CalendarScreen()` đặt màn hình lịch là màn hình chính.

File này chịu trách nhiệm khởi tạo ứng dụng, kết nối với Firebase và định nghĩa giao diện cơ bản.

3.4.2 *lunar_utils.dart*

Thuật toán trong file `lunar_utils.dart` dựa trên các nguyên tắc thiên văn để chuyển đổi từ lịch dương (Gregorian) sang lịch âm (Lunar Calendar) của Việt Nam. Lịch âm Việt Nam là một lịch âm-dương, kết hợp chu kỳ Mặt Trăng (tháng) và chu kỳ Mặt Trời (năm, tiết khí). Dưới đây là giải thích chi tiết:

Nguyên lý cơ bản:

- Chu kỳ Mặt Trăng: Một tháng âm lịch bắt đầu từ ngày trăng mới (new moon), kéo dài khoảng 29.530588853 ngày. Đây là thời gian Mặt Trăng quay quanh Trái Đất.
- Chu kỳ Mặt Trời: Năm âm lịch được điều chỉnh để đồng bộ với năm dương (khoảng 365.25 ngày) bằng cách thêm tháng nhuận khi cần.
- Số ngày Julian (JDN): Hệ đếm ngày liên tục từ năm 4713 TCN, dùng để đơn giản hóa tính toán thiên văn.

Các bước chính trong thuật toán

a) Chuyển đổi ngày dương sang số ngày Julian (*jdFromDate*)

- **Input:** Ngày (dd), tháng (mm), năm (yy) dương lịch.
- **Output:** Số ngày Julian (JDN).
- **Công thức:**

+ Điều chỉnh tháng và năm để tính đúng:

```
int a = floor((14 - mm) / 12);
int y = yy + 4800 - a;
int m = mm + 12 * a - 3;
```

+Tính JDN:

```
int jd = dd + floor((153 * m + 2) / 5) + 365 * y + floor(y / 4) -
floor(y / 100) + floor(y / 400) - 32045;
```

+ Điều chỉnh cho các ngày trước lịch Gregorian (15/10/1582):

```
if (jd < 2299161) {
    jd = dd + floor((153 * m + 2) / 5) + 365 * y + floor(y / 4) -
    32083;
}
```

Ý nghĩa: JDN là cơ sở để tính ngày trăng mới và xác định ngày âm lịch.

b) Tìm ngày trăng mới (*getNewMoonDay*)

- **Input:** Số thứ tự chu kỳ trăng mới (k) và múi giờ (timeZone).
- **Output:** JDN của ngày trăng mới.
- **Công thức:**

+ Tính thời gian trung bình của chu kỳ trăng mới:

```
double T = k / 1236.85;

double Jd1 = 2415020.75933 + 29.53058868 * k + 0.0001178 * T2 -
0.000000155 * T3;
```

+ Điều chỉnh sai lệch do dao động quỹ đạo Mặt Trăng và Mặt Trời:

```
double C1 = (0.1734 - 0.000393 * T) * sin(M * dr) + 0.0021 * sin(2 *
dr * M);
```

```
.....
```

```
double JdNew = Jd1 + C1 - deltat;
```

+ Quy đổi về ngày nguyên gần nhất:

```
return floor(JdNew + 0.5 + timeZone / 24);
```

- **Ý nghĩa:** Ngày trăng mới là ngày bắt đầu tháng âm lịch.

c) Chuyển đổi sang lịch âm (convertSolar2Lunar)

Input: Ngày, tháng, năm dương lịch và múi giờ.

Output: Map chứa ngày, tháng, năm âm lịch và cờ tháng nhuận.

Logic:

1. Tính JDN từ ngày dương lịch (jdFromDate).
2. Tìm ngày trăng mới gần nhất (getNewMoonDay) để xác định ngày đầu tháng âm lịch.
3. Tính ngày âm lịch:

```
lunarDay = dayNumber - monthStart + 1;
```

4. Xác định tháng và năm âm lịch:

- Tìm ngày tháng 11 âm lịch (getLunarMonth11) làm mốc.
- Tính số tháng kể từ tháng 11 trước đó:

```
int diff = floor((monthStart - a11) / 29);
```

```
lunarMonth = diff + 11;
```

5. Xử lý tháng nhuận: Nếu năm có hơn 12 tháng (do thêm tháng nhuận), điều chỉnh:

```
if (b11 - a11 > 365) {

    int leapMonthDiff = getLeapMonthOffset(a11, timeZone);

    if (diff >= leapMonthDiff) {

        lunarMonth = diff + 10;

        if (diff == leapMonthDiff) lunarLeap = 1;

    }

}
```

6. Chuẩn hóa tháng (1-12) và điều chỉnh năm nếu cần:

```
if (lunarMonth > 12) lunarMonth -= 12;

if (lunarMonth >= 11 && diff < 4) lunarYear -= 1;
```

d) Xác định tháng nhuận (getLeapMonthOffset)

Logic: Kiểm tra chu kỳ tiết khí (kinh độ Mặt Trời) giữa các ngày trăng mới để tìm tháng không chứa tiết khí (tháng nhuận).

Ý nghĩa: Đảm bảo năm âm lịch đồng bộ với năm dương.

Độ chính xác:

Thuật toán dựa trên các công thức thiên văn chuẩn, được điều chỉnh với múi giờ Việt Nam (7.0), đảm bảo tính chính xác cao cho lịch âm Việt Nam.

Ý nghĩa: File này là "trái tim" của ứng dụng, cung cấp logic chuyển đổi từ lịch dương sang lịch âm, dựa trên các thuật toán thiên văn chính xác.

3.4.3 *holiday_info.dart*

File này định nghĩa thông tin về các ngày lễ truyền thống của Việt Nam theo lịch âm, cung cấp dữ liệu để hiển thị trên giao diện lịch.



Hình 15: *holiday_info.dart*

- Lớp `HolidayInfo`: Chứa thông tin về ngày lễ: name (tên), description (mô tả), color (màu sắc hiển thị).
- Hàm `getHoliday(int lunarDay, int lunarMonth, int isLeap)`:
 - + Trả về thông tin ngày lễ dựa trên ngày và tháng âm lịch.
 - + Nếu `isLeap == 1` (tháng nhuận), không có ngày lễ.

+ Các ngày lễ tiêu biểu:

Tết Nguyên Đán (1/1 âm lịch, màu đỏ).

Rằm Tháng Giêng (15/1 âm lịch, màu tím).

Giỗ Tổ Hùng Vương (10/3 âm lịch, màu đỏ).

Tết Trung Thu (15/8 âm lịch, màu cam).

Ý nghĩa: File này giúp hiển thị các ngày lễ quan trọng của Việt Nam trên lịch, tăng tính thực tế và văn hóa cho ứng dụng.

3.4.4 *calendar_screen.dart*

File này là giao diện chính, hiển thị lịch và xử lý tương tác người dùng.

- Lớp `CalendarScreen` và `_CalendarScreenState`: Là một `StatefulWidget` để quản lý trạng thái (ngày được chọn, ngày đã lưu, v.v.).

- **Biến trạng thái:**

+ `focusedDay`: Ngày đang được hiển thị (tháng/năm).

+ `selectedDay`: Ngày được người dùng chọn.

+ `savedDates`: Danh sách ngày đã lưu trên Firestore.

+ `TIMEZONE`: Múi giờ (7.0 cho Việt Nam).

- **Firestore Firestore:**

+ `_datesCollection`: Collection trên Firestore để lưu ngày.

+ `_saveDate()`: Lưu ngày vào Firestore.

+ `_deleteDate()`: Xóa ngày khỏi Firestore.

+ `_loadSavedDates()`: Tải danh sách ngày đã lưu khi khởi tạo.

- **Giao diện lịch (`TableCalendar`):**

+ Sử dụng package `table_calendar` để hiển thị lịch dương.

+ Tùy chỉnh:

`headerTitleBuilder`: Hiển thị tháng/năm và cho phép chọn qua dialog.

`dowBuilder`: Hiển thị thứ (T2, T3,..., CN).

`defaultBuilder`, `selectedBuilder`, `todayBuilder`: Tùy chỉnh ô ngày với thông tin âm lịch và ngày lễ.

- Hàm `buildDayCell`:

+ Xây dựng giao diện mỗi ô ngày:

+ Hiển thị ngày dương lịch.

+ Ngày âm lịch (dưới góc phải).

+ Tên ngày lễ nếu có (góc trên trái).

+ Đánh dấu ngày đã lưu (viền vàng) và ngày hiện tại (viền xám).

- Dialog chọn tháng/năm: `selectYearMonth`: Hiển thị dialog để người dùng chọn tháng và năm.

- Nút lưu ngày và xem lịch sử:

+ Nút "Lưu ngày" gọi `saveDate`.

+ `FloatingActionButton` mở dialog danh sách ngày đã lưu, cho phép xóa.

Ý nghĩa: File này tích hợp tất cả logic (chuyển đổi lịch, ngày lễ, lưu trữ) vào giao diện người dùng, cung cấp trải nghiệm trực quan và tương tác.

KẾT LUẬN

1. Kết quả đạt được

Đề tài "Xây dựng ứng dụng xem dương lịch và âm lịch" đã hoàn thành với việc tạo ra một ứng dụng hoạt động tốt trên Android, iOS và web. Ứng dụng hiển thị lịch dương và âm lịch chính xác, cung cấp thông tin về các ngày lễ lớn của Việt Nam (như Tết Nguyên Đán, Trung Thu), và cho phép người dùng lưu/xóa ngày quan trọng trên Firebase Firestore. Giao diện thân thiện, dễ sử dụng, đáp ứng đúng mục tiêu đề ra.

2. Nhược điểm

Ứng dụng còn một số hạn chế: chưa có xác thực người dùng, danh sách ngày lễ chưa đầy đủ, chưa tối ưu hiệu suất khi lưu nhiều ngày, và thiếu tính năng thông báo nhắc nhở. Giao diện cũng chưa hoàn toàn phù hợp với màn hình lớn.

3. Hướng phát triển

Trong tương lai, ứng dụng có thể được cải thiện bằng cách thêm xác thực người dùng, mở rộng danh sách ngày lễ, tối ưu hiệu suất với phân trang, tích hợp thông báo, và hỗ trợ đa ngôn ngữ để phục vụ nhiều đối tượng hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

1. <https://aws.amazon.com/vi/what-is/flutter/>
2. <https://tokyotechlab.com/vi/blogs/ngon-ngu-dart-la-gi>
3. <https://lyso.vn/co-ban/thuat-toan-tinh-am-lich-ho-ngoc-duc-t2093/>
4. <https://fptshop.com.vn/tin-tuc/thu-thuat/firebase-la-gi-159218>
5. <https://linhthong.com/doingayamduong>