**TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------oOo---------



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN CƠ SỞ DỮ LIỆU NÂNG CAO**

**TÊN ĐỀ TÀI: FIREBASE/FIRESTORE**

**GVHD: LƯƠNG TRẦN HY HIẾN**

**NHÓM THỰC HIỆN: NHÓM 13**

**Thành viên nhóm:**

1. Nguyễn Thanh Huy Hoàng – 43.01.104.056
2. Phan Huỳnh Văn Ngọc – 43.01.104.116
3. Lê Tấn Khôi – 43.01.104.084
4. Lê Trường Thanh – 43.01.104.162

MỤC LỤC

[I. Giới thiệu Firebase: 9](#_Toc24960960)

[Giới thiệu Cloud Firesore: 11](#_Toc24960961)

[Ưu điểm: 11](#_Toc24960962)

[Cách hoạt động: 12](#_Toc24960963)

[II. Bắt đầu với Cloud Firestore: 13](#_Toc24960964)

[Tạo cơ sở dữ liệu Cloud Firestore: 13](#_Toc24960965)

[Thiết lập môi trường phát triển của bạn 14](#_Toc24960966)

[Khởi tạo Cloud Firestore 14](#_Toc24960967)

[Thêm dữ liệu 15](#_Toc24960968)

[Đọc dữ liệu 16](#_Toc24960969)

[Bảo mật dữ liệu 16](#_Toc24960970)

[III. Hiểu rõ Cloud Firestore 18](#_Toc24960971)

[So sánh Cloud Firesore và Cơ sở dữ liệu thời gian thực 18](#_Toc24960972)

[Chọn cơ sở dữ liệu cho project 18](#_Toc24960973)

[Mô hình dữ liệu (data model) 18](#_Toc24960974)

[Hỗ trợ thời gian thực và ngoại tuyến 19](#_Toc24960975)

[Truy vấn 19](#_Toc24960976)

[Ghi và giao tác 19](#_Toc24960977)

[Độ tin cậy và hiệu suất 20](#_Toc24960978)

[Bảo mật 20](#_Toc24960979)

[Giá cả 21](#_Toc24960980)

[Sử dụng Cloud Firestore và cơ sở dữ liệu thời gian thực 21](#_Toc24960981)

[Cloud Firestore Data model 21](#_Toc24960982)

[Documents 22](#_Toc24960983)

[Collections 23](#_Toc24960984)

[Tham chiếu 24](#_Toc24960985)

[Dữ liệu phân cấp 24](#_Toc24960986)

[Subcollections 25](#_Toc24960987)

[Các kiểu dữ liệu được hỗ trợ 27](#_Toc24960988)

[Kiểu dữ liệu: 27](#_Toc24960989)

[Thứ tự của các giá trị kiểu dữ liệu 28](#_Toc24960990)

[SDK và client libraries 29](#_Toc24960991)

[Mobile và web SDKs 29](#_Toc24960992)

[References và resources 29](#_Toc24960993)

[Server client libraries 30](#_Toc24960994)

[Firebase Admin SDKs 30](#_Toc24960995)

[References và resources 31](#_Toc24960996)

[Google Cloud client libraries 32](#_Toc24960997)

[References và resources 32](#_Toc24960998)

[Tích hợp thư viện của bên thứ ba 32](#_Toc24960999)

[Các loại chỉ mục trong Cloud Firestore 32](#_Toc24961000)

[Một chỉ mục đằng sau mọi truy vấn 32](#_Toc24961001)

[Quản lý chỉ mục ít hơn, phát triển ứng dụng nhiều hơn 33](#_Toc24961002)

[Các loại chỉ mục 33](#_Toc24961003)

[Chỉ mục trường đơn 33](#_Toc24961004)

[Chỉ mục hỗn hợp 34](#_Toc24961005)

[Chế độ chỉ mục và phạm vi truy vấn 35](#_Toc24961006)

[Ví dụ lập chỉ mục 35](#_Toc24961007)

[Các truy vấn được hỗ trợ bởi các chỉ mục trường đơn 37](#_Toc24961008)

[Các truy vấn được hỗ trợ bởi các chỉ mục tổng hợp 38](#_Toc24961009)

[Các truy vấn được hỗ trợ bởi các chỉ mục nhóm collection 39](#_Toc24961010)

[Lập chỉ mục và chi phí 40](#_Toc24961011)

[Lợi dụng hợp nhất lập chỉ mục 40](#_Toc24961012)

[Giới hạn chỉ mục 42](#_Toc24961013)

[Mẫu tốt nhất cho lập chỉ mục 43](#_Toc24961014)

[Vị trí cơ sở dữ liệu của Cloud Firestore 43](#_Toc24961015)

[Default GCP resource location 44](#_Toc24961016)

[Các loại vị trí 44](#_Toc24961017)

[Vị trí đa vùng 45](#_Toc24961018)

[Vị trí khu vực 45](#_Toc24961019)

[Vị trí SLA 46](#_Toc24961020)

[Chi phí vị trí 46](#_Toc24961021)

[Xem cài đặt của vị trí dự án của bạn 46](#_Toc24961022)

[IV. Thêm và quản lý dữ liệu: 46](#_Toc24961023)

[Xây dựng dữ liệu 46](#_Toc24961024)

[Dữ liệu lồng nhau trong document 47](#_Toc24961025)

[Các collection con 47](#_Toc24961026)

[Collection cấp gốc (root-level) 48](#_Toc24961027)

[Thêm dữ liệu vào Cloud Firestore 49](#_Toc24961028)

[Tạo document 49](#_Toc24961029)

[Cập nhật một document 51](#_Toc24961030)

[Giao dịch và tập ghi 55](#_Toc24961031)

[Cập nhật dữ liệu với các giao dịch 55](#_Toc24961032)

[Các tập ghi 57](#_Toc24961033)

[Xác thực dữ liệu cho các hoạt động vi mô 58](#_Toc24961034)

[Xóa dữ liệu khỏi Cloud Firestore 61](#_Toc24961035)

[Xóa document 61](#_Toc24961036)

[Xóa các trường 61](#_Toc24961037)

[Xóa collection 62](#_Toc24961038)

[Xóa dữ liệu với Firebase CLI 62](#_Toc24961039)

[Quản lý Cloud Firestore với Bảng điều khiển Firebase 63](#_Toc24961040)

[Xem dữ liệu 63](#_Toc24961041)

[Quản lý dữ liệu 65](#_Toc24961042)

[Quản lý quy tắc bảo mật đám mây 67](#_Toc24961043)

[Quản lý chỉ mục 67](#_Toc24961044)

[Giám sát việc sử dụng 67](#_Toc24961045)

[Xuất và nhập dữ liệu 67](#_Toc24961046)

[Trước khi bắt đầu 68](#_Toc24961047)

[Xuất dữ liệu 69](#_Toc24961048)

[Nhập dữ liệu 70](#_Toc24961049)

[Quản lý hoạt động xuất nhập dữ liệu 71](#_Toc24961050)

[Thanh toán và giá cả cho các hoạt động xuất nhập 73](#_Toc24961051)

[Xuất sang BigQuery 73](#_Toc24961052)

[Di chuyển dữ liệu giữa các dự án 74](#_Toc24961053)

[Trước khi bắt đầu 74](#_Toc24961054)

[Xuất dữ liệu từ dự án nguồn 75](#_Toc24961055)

[Nhập dữ liệu vào dự án đích 77](#_Toc24961056)

[Bắt đầu một hoạt động nhập khẩu 78](#_Toc24961057)

[V. Đọc dữ liệu: 79](#_Toc24961058)

[Lấy dữ liệu 1 lần 79](#_Toc24961059)

[Lấy 1 document 79](#_Toc24961060)

[Lấy nhiều document từ 1 collection 80](#_Toc24961061)

[Liệt kê các subcollection của 1 document 80](#_Toc24961062)

[Truy vấn dữ liệu 81](#_Toc24961063)

[Truy vấn đơn giản 81](#_Toc24961064)

[Truy vấn phức 81](#_Toc24961065)

[Một số hạn chế truy vấn 81](#_Toc24961066)

[Phân loại và giới hạn dữ liệu 82](#_Toc24961067)

[Phân trang dữ liệu với cursor 82](#_Toc24961068)

[Định nghĩa một cursor 82](#_Toc24961069)

[Sử dụng document snapshot để định nghĩa cursor 83](#_Toc24961070)

[Phân trang truy vấn 83](#_Toc24961071)

[Thiết lập nhiều điều kiện cho cursor 84](#_Toc24961072)

[Truy cập dữ liệu khi ngoại tuyến 84](#_Toc24961073)

[Quản lý chỉ mục (index) 84](#_Toc24961074)

[Thời gian xây dựng chỉ mục 85](#_Toc24961075)

[Lỗi xây dựng chỉ mục 85](#_Toc24961076)

[VI. Bảo mật và xác nhận dữ liệu: 85](#_Toc24961077)

[Cloud firestore Security rules 85](#_Toc24961078)

[Quy tắc bảo mật version 2 (Security rules version 2) 85](#_Toc24961079)

[Viết Rules: 86](#_Toc24961080)

[Cấu trúc của Cloud Firestore Security Rules 88](#_Toc24961081)

[Phục vụ và khai báo cơ sở dữ liệu: 88](#_Toc24961082)

[Quy tắc Read/write cơ bản 88](#_Toc24961083)

[Toán tử chia nhỏ (Granular operations) 89](#_Toc24961084)

[Dữ liệu Phân cấp (hierarchical data) 90](#_Toc24961085)

[Kí tự đại diện Đệ quy (Recursive wildcards) 91](#_Toc24961086)

[Chồng lấp match 92](#_Toc24961087)

[Viết điều kiện cho Quy tắc bảo mật Cloud Firestore 93](#_Toc24961088)

[Tính xác thực 93](#_Toc24961089)

[Xác nhận dữ liệu (Data validation) 94](#_Toc24961090)

[Truy cập các document khác 94](#_Toc24961091)

[Giới hạn truy xuất 95](#_Toc24961092)

[Gọi truy vấn và chi phí 95](#_Toc24961093)

[Chức năng tùy chỉnh (Custom functions) 95](#_Toc24961094)

[Quy tắc không phải là bộ lọc (Rules are not filters) 96](#_Toc24961095)

[Thay đổi các quy tắc không an toàn 98](#_Toc24961096)

[(Fix insecure rules) 98](#_Toc24961097)

[Hiểu quy tắc bảo mật Cloud Firestore của bạn 98](#_Toc24961098)

[Các tình huống phổ biến với các quy tắc không an toàn 98](#_Toc24961099)

[Mở truy cập 98](#_Toc24961100)

[Truy cập cho bất kỳ người dùng đã xác thực 99](#_Toc24961101)

[Truy cập đóng 100](#_Toc24961102)

[Kiểm tra các quy tắc bảo mật Cloud Firestore của bạn 101](#_Toc24961103)

[Kiểm tra các quy tắc bảo mật Cloud Firestore của bạn 101](#_Toc24961104)

[Hiểu về quy tắc bảo mật Cloud Firestore 101](#_Toc24961105)

[Install trình giả lập 102](#_Toc24961106)

[Chạy trình giả lập 102](#_Toc24961107)

[Trước khi chạy trình giả lập 103](#_Toc24961108)

[Chạy thử nghiệm cục bộ 103](#_Toc24961109)

[Tạo báo cáo thử nghiệm 105](#_Toc24961110)

[Điểm khác nhau giữa trình giả lập và sản phẩm thật: 106](#_Toc24961111)

[Khắc phục sự cố đã biết: 106](#_Toc24961112)

[\*Hành vi kiểm tra không nhất quán: 106](#_Toc24961113)

[\*Các kiểm thử chỉ qua mỗi lần đầu tiên khi bạn tải trình giả lập: 106](#_Toc24961114)

[\* Thiết lập thử nghiệm rất phức tạp: 107](#_Toc24961115)

[Dữ liệu truy vấn an toàn 107](#_Toc24961116)

[\* Quy tắc không phải là bộ lọc 107](#_Toc24961117)

[\* Truy vấn và quy tắc bảo mật 107](#_Toc24961118)

[Các truy vấn nhóm bộ kết hợp và quy tắc bảo mật (Collection group queries and security rules) 110](#_Toc24961119)

[VII. Demo: 115](#_Toc24961120)

[VIII. Document tham khảo: 115](#_Toc24961121)

# Giới thiệu Firebase:

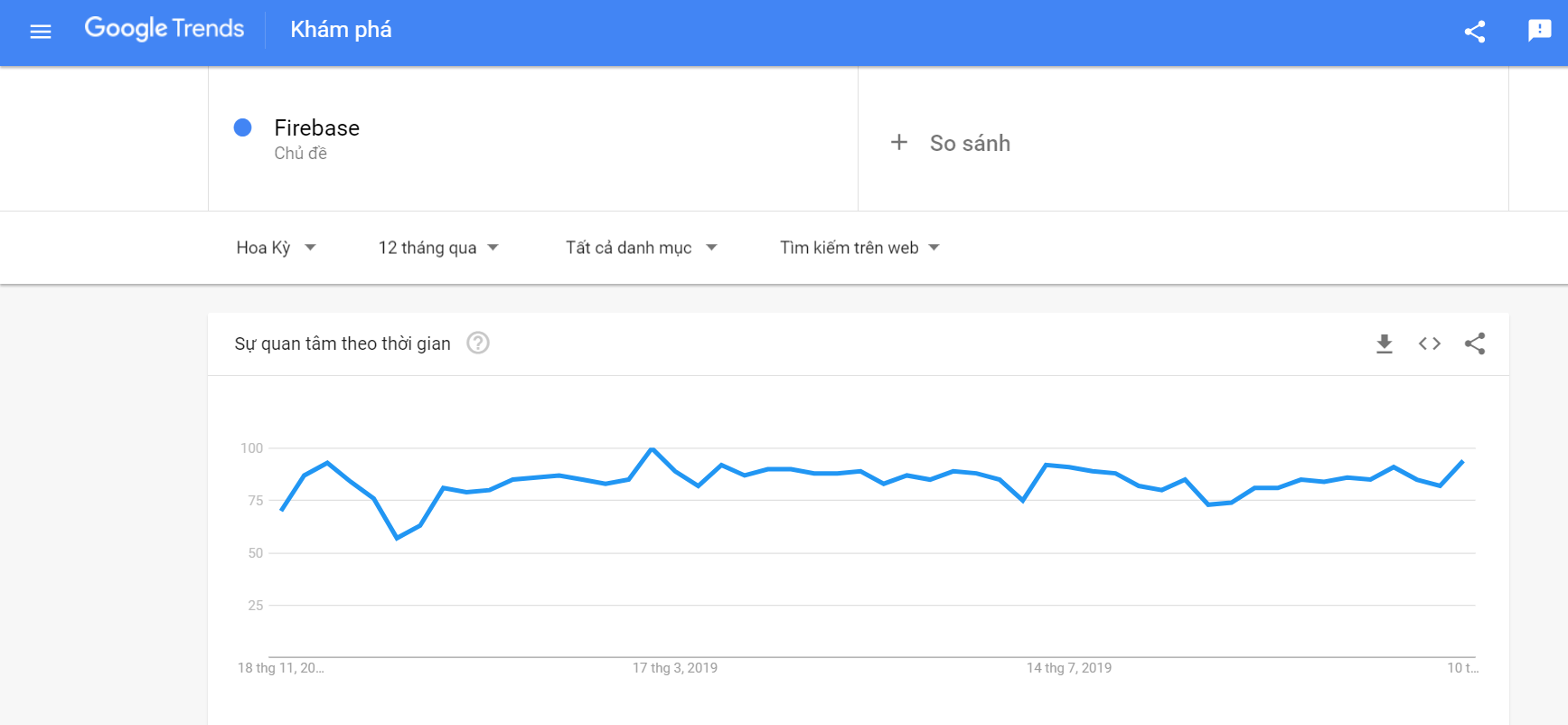
Firebase là một nền tảng phát triển ứng dụng di động và web. Họ cung cấp rất nhiều công cụ và dịch vụ để phát triển ứng dụng chất lượng, rút ngắn thời gian phát triển và phát triển cơ sở người dùng mà không cần quan tâm đến hạ tầng phần cứng.

Firebase là sự kết hợp giữa nền tảng cloud với hệ thống máy chủ cực kì mạnh mẽ của Google. Firebase cung cấp cho chúng ta những API đơn giản, mạnh mẽ và đa nền tảng trong việc quản lý, sử dụng database.

Ví dụ: Với Firebase, chúng ta có thể tự xây dựng một ứng chat thời gian thực như Facebook mesage, Zalo… trong vài ngày, thậm chí vài giờ.

Đơn giản vì giờ chúng ta chỉ cần phải lo phát triển phía Client( Ứng dụng mobile), còn phần backend(server) đã có Firebase lo, chúng ta chỉ cần gọi API là đủ.

Trên thế giới thì xu hướng sử dụng Firebase rất lớn. Theo Google Trend:



Firebase có tiền thân là Envolve. Đây là dịch vụ cung cấp những API để bạn dễ dàng tích hợp tính năng chat vào trang web.

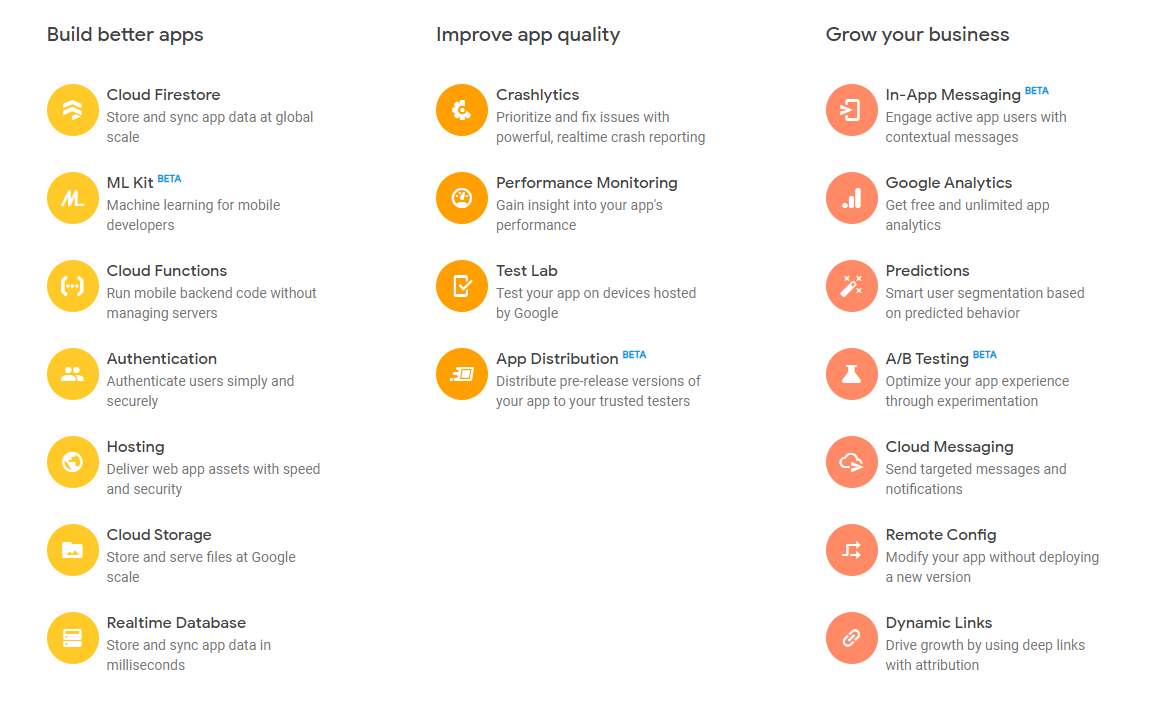
Điều thú vị là người dùng Envolve sử dụng dịch vụ để truyền dữ liệu chứ không đơn thuần cho ứng dụng chat. Họ sử dụng Envolve để đồng bộ dữ liệu của những ứng dụng như game online, danh bạ, lịch…

Nhận biết được điều này, CEO của Envolve đã tách biệt hệ thống chat và đồng bộ dữ liệu thời gian thực thành 2 mảng riêng biệt.

Đến năm 2012, Firebase được thành lập như một công ty cung cấp dịch vụ Backend-as-a-Service theo thời gian thực.

Ngửi thấy mùi tiềm năng, năm 2014, Google lập tức mua lại Firebase với giá không được tiết lộ. Và giờ Google phát triển Firebase thành một dịch vụ đồ sộ như bạn đang thấy.

Hiện nay, danh mục dịch vụ của Firebase rất nhiều. Từ hệ thống chat thời gian thực, đến A/B testing… và cả ML KIT( Bộ công cụ phát triển Machine learning).



Trong đó nhóm sẽ tập trung tìm hiểu về Cloud Firestore.

# Giới thiệu Cloud Firestore:

Cloud Firestore là một cơ sở dữ liệu linh hoạt, có thể mở rộng để phát triển máy chủ, di động và web từ Firebase và Google Cloud Platform. Giống như Firebase Realtime Database, nó giữ dữ liệu của bạn đồng bộ hóa trên các ứng dụng khách hàng thông qua realtime listeners và cung cấp hỗ trợ ngoại tuyến cho thiết bị di động và web để bạn có thể xây dựng các ứng dụng hoạt động bất kể độ trễ mạng hoặc kết nối Internet. Cloud Firestore cũng cung cấp tích hợp liền mạch với các sản phẩm của Firebase và Google Cloud Platform khác, bao gồm cả Cloud Functions

## Ưu điểm:

Linh hoạt: Mô hình dữ liệu Cloud Firestore hỗ trợ các cấu trúc dữ liệu linh hoạt, phân cấp. Lưu trữ dữ liệu của bạn trong các documents, được tổ chức thành các collections. Các documents có thể chứa các đối tượng lồng nhau phức tạp bên cạnh các subcollections.

Truy vấn mạnh mẽ: Trong Cloud Firestore, bạn có thể sử dụng các truy vấn để truy xuất các documents riêng lẻ, cụ thể hoặc để truy xuất tất cả các documents trong collections khớp với các tham số truy vấn của bạn. Các truy vấn của bạn có thể bao gồm nhiều, chuỗi bộ lọc và kết hợp lọc và sắp xếp. Chúng cũng được liệt kê theo mặc định, vì vậy hiệu suất truy vấn tỷ lệ thuận với kích thước của tập kết quả của bạn chứ không phải tập dữ liệu của bạn.

Cập nhật thời gian thực: Cloud Firestore sử dụng đồng bộ hóa dữ liệu để cập nhật dữ liệu trên mọi thiết bị được kết nối. Tuy nhiên, nó cũng được thiết kế để thực hiện các truy vấn tìm nạp một lần đơn giản một cách hiệu quả.

Hỗ trợ ngoại tuyến: Cloud Firestore lưu trữ dữ liệu mà ứng dụng của bạn đang sử dụng, vì vậy ứng dụng có thể viết, đọc, nghe và truy vấn dữ liệu ngay cả khi thiết bị ngoại tuyến. Khi thiết bị trở lại trực tuyến, Cloud Firestore sẽ đồng bộ hóa mọi thay đổi cục bộ trở lại Cloud Firestore.

Được thiết kế theo quy mô lớn: Cloud Firestore mang đến cho bạn cơ sở hạ tầng mạnh mẽ nhất của Google Cloud Platform: sao chép dữ liệu đa vùng tự động, đảm bảo tính nhất quán mạnh mẽ, điều khiển hàng loạt nguyên tử và hỗ trợ giao dịch thực sự. Cloud Firestore được thiết kế để xử lý khối lượng công việc cơ sở dữ liệu khó khăn nhất từ các ứng dụng lớn nhất thế giới.

## Cách hoạt động:

Cloud Firestore là một cơ sở dữ liệu NoQuery được lưu trữ trên đám mây mà các ứng dụng web, iOS và web của bạn có thể truy cập trực tiếp thông qua SDK gốc. Cloud Firestore cũng có sẵn trong SDK Node.js, Java, Python và Go SDK, bệnh cạnh đó là API REST và RPC.

Theo mô hình dữ liệu NoQuery của Cloud Firestore, bạn lưu trữ dữ liệu trong các documents có chứa ánh xạ trường tới các giá trị. Các documents được lưu trữ trong các collections, là nơi cho các documents của bạn mà bạn có thể sử dụng để sắp xếp dữ liệu của mình và xây dựng các truy vấn. Document hỗ trợ nhiều loại dữ liệu khác nhau, từ các chuỗi và số đơn giản, đến các đối tượng phức tạp, lồng nhau. Bạn cũng có thể tạo các subcollections trong các documents và xây dựng các cấu trúc dữ liệu phân cấp có quy mô khi cơ sở dữ liệu của bạn phát triển. Mô hình dữ liệu Cloud Firestore hỗ trợ mọi cấu trúc dữ liệu hoạt động tốt nhất cho ứng dụng của bạn.

Ngoài ra, truy vấn trong Cloud Firestore là mạnh mẽ, hiệu quả và linh hoạt. Tạo các truy vấn đơn giàn để truy xuất dữ liệu ở cấp độ document mà không cần truy xuất toàn bộ collection hoặc bất kỳ tập hợp con lồng nhau nào. Thêm sắp xếp, lọc và giới hạn cho các truy vấn hoặc con trỏ của bạn để phân trang kết quả của bạn. Để giữ dữ liệu trong ứng dụng của bạn hiện tại, mà không cần truy xuất toàn bộ cơ sở dữ liệu của bạn mỗi khi có cập nhật, hãy thêm trình nghe thời gian thực. Thêm realtime listeners vào ứng dụng của bạn sẽ thông báo cho bạn bằng cách lưu nhanh dữ liệu bất cứ khi nào dữ liệu mà ứng dụng khách của bạn đang phản hồi các thay đổi, chỉ truy xuất các thay đổi mới.

Bảo vệ quyền truy cập vào dữ liệu của bạn trong Cloud Firestore với Firebase Authentication và Cloud Firestore Security Rules cho Android, iOS và JavaScript hoặc Quản lý truy cập và nhận dạng (Identity and Access Management: IAM) cho các ngôn ngữ phía máy chủ.

# Bắt đầu với Cloud Firestore:

Tiếp theo chúng tôi sẽ hướng dẫn bạn cách thiết lập Cloud Firestore, thêm dữ liệu, sau đó xem dữ liệu bạn vừa thêm trong Firebase console.

## Tạo cơ sở dữ liệu Cloud Firestore:

1. Nếu bạn chưa có project Firebase thì hãy tạo một project firebase mới: Trong Firebase console, nhấp vào **Add project**, sau đó làm theo các hướng dẫn trên màn hình để tạo dự án Firebase hoặc thêm dịch vụ Firebase vào dự án GCP hiện có.
2. Từ ngăn điều hướng của console, chọn **Database**, sau đó bấm **Create database** cho Cloud Firestore.
3. Chọn chế độ bắt đầu cho Quy tắc bảo mật Cloud Firestore của bạn:

**Test mode:**

Tốt để bắt đầu với client libraries của web và thiết bị di động, nhưng cho phép mọi người đọc và ghi đè lên dữ liệu của bạn. Sau khi kiểm tra, đảm bảo xem lại phần Bảo mật dữ liệu của bạn.

Để bắt đầu với web, iOS hoặc Android SDK, hãy chọn test mode.

**Locked mode**:

Từ chối đọc và viết từ mobile và web clients. Các máy chủ ứng dụng được xác thực của bạn (C #, Go, Java, Node.js, PHP, Python hoặc Ruby) vẫn có thể truy cập cơ sở dữ liệu của bạn.

Để bắt đầu với C #, Go, Java, Node.js, PHP, Python hoặc Ruby sever client library, hãy chọn locked mode.

1. Chọn một location cho cơ sở dữ liệu của bạn.

* Location setting này là resource location mặc định của Google Cloud Platform (GCP). Lưu ý rằng location này sẽ được sử dụng cho các dịch vụ GCP trong dự án của bạn yêu cầu location setting, cụ thể, Cloud Storage bucket mặc định của bạn và ứng dụng App Engine của bạn (được yêu cầu nếu bạn sử dụng Cloud Scheduler)
* Nếu bạn không chọn một location, thì dự án của bạn đã có một GCP resource location mặc định. Nó được cài đặt trong khi tạo dự án hoặc khi thiết lập một dịch vụ khác yêu cầu location setting.

**Cảnh báo: Sau khi bạn đặt GCP resource location** **mặc định của dự án, bạn không thể thay đổi nó.**

1. Chọn Done.

Cloud Firestore và App Engine: Bạn không thể sử dụng cả Cloud Firestore và Cloud Datastore trong cùng một dự án, điều này có thể ảnh hưởng đến các ứng dụng sử dụng App Engine. Hãy thử sử dụng Cloud Firestore với một dự án khác.

Khi bạn bật Cloud Firestore, nó cũng kích hoạt API trong Cloud API Manager.

## Thiết lập môi trường phát triển của bạn

Thêm các dependencies cần thiết và client libraries vào ứng dụng của bạn.

PHP:

1. Các thư viện khách của máy chủ Cloud Firestore(Java, Node.js, Python, Go, PHP, C # và Ruby) sử dụng Thông tin xác thực mặc định của ứng dụng Google để xác thực.

. Để xác thực từ môi trường phát triển của bạn, hãy đặt biến môi trường GOOGLE\_APPLICATION\_CREDENTIALS để trỏ đến tệp chính của tài khoản dịch vụ JSON. Bạn có thể tạo một tệp chính trên trang API Console Credentials page.

export GOOGLE\_APPLICATION\_CREDENTIALS="path/to/your/keyfile.json"

. Trong môi trường sản xuất của bạn, bạn không cần xác thực nếu bạn chạy ứng dụng của mình trên App Engine hoặc Compute Engine, sử dụng cùng một dự án mà bạn sử dụng cho Cloud Firestore. Nếu không, hãy thiết lập một service account.

1. Cài đặt và kích hoạt phần mở rộng gRPC cho PHP, mà bạn sẽ cần sử dụng thư viện máy khách.
2. Thêm thư viện PHP của Cloud Firestore vào ứng dụng của bạn:

composer require google/cloud-firestore

## Khởi tạo Cloud Firestore

Khởi tạo một phiên bản của Cloud Firestore:

PHP:

use Google\Cloud\Firestore\FirestoreClient;  
  
/\*\*  
 \* Initialize Cloud Firestore with default project ID.  
 \* ```  
 \* initialize();  
 \* ```  
 \*/  
function initialize()  
{  
    // Create the Cloud Firestore client  
    $db = new FirestoreClient();  
    printf('Created Cloud Firestore client with default project ID.' . PHP\_EOL);  
}

## Thêm dữ liệu

Cloud Firestore lưu trữ dữ liệu trong Documents, được lưu trữ trong Collections. Cloud Firestore tạo ra các collections và documents ngay lần đầu tiên bạn thêm dữ liệu vào documents. Bạn không cần phải tạo một cách rõ ràng các collections hoặc documents.

Tạo một collection mới và một document bằng cách sử dụng code ví dụ sau:

PHP:

$docRef = $db->collection('users')->document('lovelace');  
$docRef->set([  
    'first' => 'Ada',  
    'last' => 'Lovelace',  
    'born' => 1815  
]);  
printf('Added data to the lovelace document in the users collection.' . PHP\_EOL);

Bây giờ thêm một document khác vào collection người dùng. Lưu ý rằng document này bao gồm cặp key-value (tên đệm) không xuất hiện trong document đầu tiên. Document trong một collection có thể chứa các bộ thông tin khác nhau.

PHP:

$docRef = $db->collection('users')->document('aturing');  
$docRef->set([  
    'first' => 'Alan',  
    'middle' => 'Mathison',  
    'last' => 'Turing',  
    'born' => 1912  
]);  
printf('Added data to the aturing document in the users collection.' . PHP\_EOL);

## Đọc dữ liệu

Để nhanh chóng xác minh rằng bạn đã thêm dữ liệu vào Cloud Firestore, hãy sử dụng data viewer trong Firebase console.

Bạn cũng có thể sử dụng phương thức "get" để lấy toàn bộ collection.

PHP:

$usersRef = $db->collection('users');  
$snapshot = $usersRef->documents();  
foreach ($snapshot as $user) {  
    printf('User: %s' . PHP\_EOL, $user->id());  
    printf('First: %s' . PHP\_EOL, $user['first']);  
    if (!empty($user['middle'])) {  
        printf('Middle: %s' . PHP\_EOL, $user['middle']);  
    }  
    printf('Last: %s' . PHP\_EOL, $user['last']);  
    printf('Born: %d' . PHP\_EOL, $user['born']);  
    printf(PHP\_EOL);  
}  
printf('Retrieved and printed out all documents from the users collection.' . PHP\_EOL);

## Bảo mật dữ liệu

Nếu bạn đang sử dụng Web, Android hoặc iOS SDK, hãy sử dụng Firebase Authentication và Cloud Firestore Security Rules để bảo mật dữ liệu của bạn trong Cloud Firestore.

Dưới đây là một số rule sets cơ bản bạn có thể sử dụng để bắt đầu. Bạn có thể sửa đổi các security rules của mình trong tab Rules của console.

Auth required:

// Allow read/write access on all documents to any user signed in to the application  
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /{document=\*\*} {  
      allow read, write: if request.auth.uid != null;  
    }  
  }  
}

Locked mode:

// Deny read/write access to all users under any conditions  
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /{document=\*\*} {  
      allow read, write: if false;  
    }  
  }  
}

Test mode:

// Allow read/write access to all users under any conditions  
// Warning: \*\*NEVER\*\* use this rule set in production; it allows  
// anyone to overwrite your entire database.  
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /{document=\*\*} {  
      allow read, write: if true;  
    }  
  }  
}

Nếu bạn đang sử dụng một trong những server SDKs, hãy sử dụng Identity and Access Management (IAM) để bảo mật dữ liệu của bạn trong Cloud Firestore.

# Hiểu rõ Cloud Firestore

## So sánh Cloud Firesore và Cơ sở dữ liệu thời gian thực

Firebase cung cấp hai giải pháp cơ sở dữ liệu mà khách hàng có thể truy cập dựa trên đám mây hỗ trợ đồng bộ hóa dữ liệu theo thời gian thực:

* Cloud Firestore là cơ sở dữ liệu mới nhất của Firebase để phát triển ứng dụng di động. Nó xây dựng dựa trên những thành công của Cơ sở dữ liệu thời gian thực với mô hình dữ liệu mới, trực quan hơn. Cloud Firestore cũng có các truy vấn phong phú hơn, nhanh hơn và mở rộng hơn so với Cơ sở dữ liệu thời gian thực.
* Cơ sở dữ liệu thời gian thực là cơ sở dữ liệu ban đầu của Firebase. Đây là một giải pháp hiệu quả, độ trễ thấp cho các ứng dụng di động yêu cầu trạng thái được đồng bộ hóa giữa các máy khách trong thời gian thực.

### Chọn cơ sở dữ liệu cho project

Chúng tôi khuyên dùng Cloud Firestore cho hầu hết các nhà phát triển bắt đầu một dự án mới. Cloud Firestore cung cấp thêm chức năng, hiệu suất và khả năng mở rộng trên cơ sở hạ tầng được thiết kế để hỗ trợ các tính năng mạnh mẽ hơn trong các phiên bản tương lai. Hy vọng sẽ thấy các loại truy vấn mới, quy tắc bảo mật mạnh mẽ hơn và các cải tiến về hiệu suất trong số các tính năng nâng cao được lên kế hoạch cho Cloud Firestore.

Khi bạn chọn giữa các giải pháp cơ sở dữ liệu, hãy xem xét các khác biệt sau đây giữa Cloud Firestore và Cơ sở dữ liệu thời gian thực.

### Mô hình dữ liệu (data model) Cả cơ sở dữ liệu thời gian thực và Cloud Firestore đều là Cơ sở dữ liệu NoSQL.

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Lưu trữ dữ liệu dưới dạng một cây JSON lớn.   * Dữ liệu đơn giản là rất dễ dàng để lưu trữ. * Dữ liệu phức tạp, phân cấp khó tổ chức hơn ở quy mô lớn | Lưu trữ dữ liệu dưới dạng collections của documents   * Dữ liệu đơn giản dễ dàng lưu trữ trong các document, rất giống với JSON. * Dữ liệu phức tạp, phân cấp dễ tổ chức hơn, sử dụng các subcollections trong documents. |

### Hỗ trợ thời gian thực và ngoại tuyến

Cả hai đều có SDK thời gian thực tiêu chuẩn mobile-first và cả hai đều hỗ trợ lưu trữ dữ liệu cục bộ cho các ứng dụng ngoại tuyến.

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Hỗ trợ ngoại tuyến cho máy khách iOS và Android. | Hỗ trợ ngoại tuyến cho máy khách iOS và Android, Web. |

### Truy vấn

Truy xuất, sắp xếp và lọc dữ liệu từ cơ sở dữ liệu thông qua các truy vấn.

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Truy vấn sâu với chức năng sắp xếp và lọc hạn chế.   * Các truy vấn có thể sắp xếp hoặc lọc trên một thuộc tính, nhưng không phải cả hai. * Các truy vấn sâu theo mặc định: chúng luôn trả về toàn bộ cây con. * Các truy vấn có thể truy cập dữ liệu ở bất kỳ mức độ chi tiết nào, xuống đến các giá trị nút lá riêng lẻ trong cây JSON. * Truy vấn không yêu cầu một chỉ mục; tuy nhiên hiệu suất của các truy vấn nhất định sẽ giảm khi tập dữ liệu của bạn tăng lên. | Truy vấn được lập chỉ mục với sắp xếp và lọc phức tạp.   * Bạn có thể xâu chuỗi các bộ lọc và kết hợp lọc và sắp xếp trên một thuộc tính trong một truy vấn duy nhất. * Các truy vấn rất nông: chúng chỉ trả về các documents trong một collection hoặc nhóm collection cụ thể và không trả về dữ liệu cùa subcollection. * Truy vấn phải luôn trả về toàn bộ documents. * Các truy vấn được lập chỉ mục theo mặc định: Hiệu suất truy vấn tỷ lệ thuận với kích thước của tập kết quả của bạn, không phải là tập dữ liệu của bạn. |

### Ghi và giao tác

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Hoạt động ghi và giao tác dữ liệu cơ bản.   * Ghi dữ liệu thông qua các hoạt động thiết lập và cập nhật. * Giao tác là nguyên tử trên một cây con dữ liệu cụ thể. | Hoạt động ghi và giao tác dữ liệu nâng cao.   * Hoạt động ghi dữ liệu thông qua các hoạt động thiết lập và cập nhật cũng như các phép biến đổi nâng cao như toán tử mảng và số. * Các giao tác có thể đọc và ghi dữ liệu từ bất kỳ phần nào của cơ sở dữ liệu. |

### Độ tin cậy và hiệu suất

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Cơ sở dữ liệu thời gian thực là một giải pháp đơn vùng.   * Cơ sở dữ liệu được giới hạn ở tính sẵn có của vùng trong một vùng duy nhất. * Độ trễ cực thấp, tùy chọn lý tưởng cho việc đồng bộ hóa trạng thái thường xuyên. | Cloud Firestore là một giải pháp đa vùng có quy mô tự động.   * Lưu trữ dữ liệu của bạn trên nhiều trung tâm dữ liệu ở các khu vực riêng biệt, đảm bảo khả năng mở rộng toàn cầu và độ tin cậy cao. * Có sẵn trong cấu hình khu vực hoặc đa khu vực trên toàn thế giới. |

**Khả năng mở rộng**

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Mở rộng yêu cầu sharding   * Quy mô lên khoảng 200.000 kết nối đồng thời và 1.000 ghi / giây trong một cơ sở dữ liệu. Mở rộng ra ngoài đòi hỏi phải bảo vệ dữ liệu của bạn trên nhiều cơ sở dữ liệu. * Không có giới hạn cục bộ về tốc độ ghi vào từng mẩu dữ liệu. | Mở rộng tự động   * Mở rộng hoàn toàn tự động. Hiện tại, giới hạn tỷ lệ là khoảng 1 triệu kết nối đồng thời và 10.000 ghi / giây. Được dự định tăng các giới hạn này trong tương lai. * Có giới hạn về tỷ lệ ghi vào các documents hoặc chỉ mục riêng lẻ. |

### Bảo mật

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Quy tắc xếp tầng ngôn ngữ (Cascading rules language) phân tách ủy quyền và xác nhận.   * Đọc và ghi từ mobile SDKs được bảo mật bởi Quy tắc cơ sở dữ liệu thời gian thực (Realtime Database Rules). * Đọc và viết quy tắc xếp tầng. * Bạn xác nhận dữ liệu riêng biệt bằng cách sử dụng quy tắc xác thực. | Quy tắc không xếp tầng (Non-cascading rules) kết hợp ủy quyền và xác nhận.   * Đọc và ghi từ mobile SDKs được bảo mật bởi Quy tắc bảo mật Cloud Firestore * Đọc và ghi từ SDK máy chủ được bảo mật bởi Identity and Access Management (IAM). * Quy tắc không xếp tầng trừ khi bạn sử dụng ký tự đại diện. * Các quy tắc có thể hạn chế các truy vấn: Nếu kết quả của truy vấn có thể chứa dữ liệu mà người dùng không có quyền truy cập, toàn bộ truy vấn sẽ thất bại. |

### Giá cả

Cả hai giải pháp đều có sẵn trên các gói giá Spark, Flame và Blaze.

|  |  |
| --- | --- |
| **Cơ sở dữ liệu thời gian thực** | **Cloud Firestore** |
| Chỉ tính phí cho băng thông và lưu trữ, nhưng với tốc độ cao hơn. | Tính phí chủ yếu cho các hoạt động được thực hiện trong cơ sở dữ liệu của bạn (đọc, ghi, xóa) và, ở tốc độ, băng thông và lưu trữ thấp hơn.  Cloud Firestore hỗ trợ giới hạn chi tiêu hàng ngày cho các dự án Google App Engine, để đảm bảo bạn không vượt quá chi phí mà bạn cảm thấy thoải mái. |

### Sử dụng Cloud Firestore và cơ sở dữ liệu thời gian thực

Bạn có thể sử dụng cả hai cơ sở dữ liệu trong cùng một ứng dụng hoặc dự án Firebase. Cả hai cơ sở dữ liệu NoQuery đều có thể lưu trữ cùng loại dữ liệu và các thư viện máy khách hoạt động theo cách tương tự. Hãy ghi nhớ những khác biệt được nêu ở trên nếu bạn quyết định sử dụng cả hai cơ sở dữ liệu trong ứng dụng của mình.

## Cloud Firestore Data model

Cloud Firestore là một cơ sở dữ liệu NoSQL hướng document (document-oriented). Không giống như cơ sở dữ liệu SQL, không có tables hoặc rows. Thay vào đó, bạn lưu trữ dữ liệu trong các documents được sắp xếp thành các collections collections.

Mỗi documents chứa một tập hợp các cặp khóa-giá trị. Cloud Firestore được tối ưu hóa để lưu trữ các collections lớn các documents nhỏ.

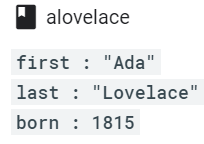
Tất cả các documents phải được lưu trữ trong collections. Các documents có thể chứa các subcollections và các đối tượng lồng nhau, cả hai đều có thể chứa các trường nguyên thủy (primitive fields) như chuỗi hoặc các đối tượng phức tạp như danh sách.

collections và documents được tạo ngầm trong Cloud Firestore. Đơn giản chỉ cần gán dữ liệu cho một documents trong một collection. Nếu collections hoặc documents không tồn tại, Cloud Firestore sẽ tạo ra nó.

### Documents

Trong Cloud Firestore, đơn vị lưu trữ là document. document là một bản ghi nhẹ chứa các trường, ánh xạ tới các giá trị. Mỗi document được xác định bằng một tên.

Một document đại diện cho người dùng alovelace có thể trông như thế này:



**Lưu ý**: Cloud Firestore hỗ trợ nhiều loại dữ liệu cho các giá trị: boolean, number, string, geo point, binary blob, và timestamp. Bạn cũng có thể sử dụng mảng hoặc các đối tượng lồng nhau (nested objects), dược gọi là maps, để cấu trúc dữ liệu trong document

Các đối tượng phức tạp, lồng nhau trong một document được gọi là maps. Ví dụ: bạn có thể cấu trúc tên người dùng từ ví dụ trên bằng map, như thế này:



Bạn có thể nhận thấy rằng các documents trông rất giống JSON. Trong thực tế, chúng về cơ bản là. Có một số khác biệt (ví dụ: documents hỗ trợ các loại dữ liệu bổ sung và bị giới hạn kích thước ở mức 1 MB), nhưng nói chung, bạn có thể coi documents là bản ghi JSON nhẹ.

### Collections

documents nằm trong các collections, đơn giản là các thùng chứa documents. Ví dụ: bạn có thể có một collection **users** để chứa nhiều người dùng khác nhau, mỗi người được đại diện bởi một document:



Cloud Firestore là schemaless (không cần định nghĩa một cấu trúc trước khi lưu data), vì vậy bạn hoàn toàn tự do đối với những trường bạn đặt trong mỗi documents và loại dữ liệu nào bạn lưu trữ trong các trường đó. Tất cả các documents trong cùng collection đều có thể chứa các trường khác nhau hoặc lưu trữ các loại dữ liệu khác nhau trong các trường đó. Tuy nhiên, nên sử dụng cùng các trường và loại dữ liệu trên nhiều document để bạn có thể truy vấn document dễ dàng hơn.

Một collection chứa các documents và không có gì khác. Nó không thể chứa trực tiếp các trường thô có giá trị và không thể chứa các collection khác. (Xem Dữ liệu phân cấp để biết giải thích về cách cấu trúc dữ liệu phức tạp hơn trong Cloud Firestore.)

Tên của các documents trong một collection là duy nhất. Bạn có thể cung cấp các khóa riêng của mình, chẳng hạn như ID người dùng hoặc bạn có thể để Cloud Firestore tự động tạo ID ngẫu nhiên cho bạn.

Bạn không cần phải "tạo" hoặc "xóa" các collection. Sau khi bạn tạo document đầu tiên trong collection, collection tồn tại. Nếu bạn xóa tất cả các documents trong một collection, nó không còn tồn tại.

### Tham chiếu

Mỗi document trong Cloud Firestore được xác định duy nhất bởi vị trí của nó trong cơ sở dữ liệu. Ví dụ trước cho thấy một document **alovelace** trong collection **users**. Để tham chiếu đến vị trí này trong code của bạn, bạn có thể tạo một tham chiếu đến nó.

PHP:

$document = $db->collection('users')->document('lovelace');

Tham chiếu là một đối tượng nhẹ chỉ trỏ đến một vị trí trong cơ sở dữ liệu của bạn. Bạn có thể tạo một tham chiếu cho dù dữ liệu có tồn tại ở đó hay không và việc tạo một tham chiếu không thực hiện bất kỳ hoạt động mạng lưới nào.

Bạn cũng có thể tạo document tham khảo cho các collections:

PHP:

$collection = $db->collection('users');

**Lưu ý**: Tham chiếu collection và Tham chiếu document là hai loại tham chiếu riêng biệt và cho phép bạn thực hiện các hoạt động khác nhau. Ví dụ: bạn có thể sử dụng tham chiếu collection để truy vấn các document trong collection và bạn có thể sử dụng tham chiếu document để đọc hoặc viết một document riêng lẻ.

Để thuận tiện, bạn cũng có thể tạo các tham chiếu bằng cách chỉ định đường dẫn đến document hoặc collection dưới dạng chuỗi, với các thành phần đường dẫn được phân tách bằng dấu gạch chéo (/). Ví dụ: để tạo một tham chiếu đến document alovelace:

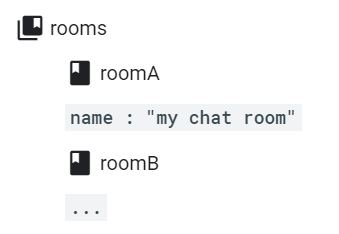
PHP:

$document = $db->document('users/lovelace');

### Dữ liệu phân cấp

Để hiểu cách cấu trúc dữ liệu phân cấp hoạt động trong Cloud Firestore, hãy xem xét một ứng dụng trò chuyện mẫu với các tin nhắn và chat rooms.

Bạn có thể tạo một collection được gọi là **rooms** để lưu trữ các **chat rooms** khác nhau:



Bây giờ bạn có chat rooms, hãy quyết định cách lưu trữ tin nhắn của bạn. Bạn có thể không muốn lưu trữ chúng trong document của chat room. Document trong Cloud Firestore phải gọn nhẹ và một chat room có thể chứa một số lượng lớn tin nhắn. Tuy nhiên, bạn có thể tạo các collection bổ sung trong document của chat room của mình dưới dạng các collections.

### Subcollections

Cách tốt nhất để lưu trữ tin nhắn trong ví dụ này là sử dụng các subcollections. Một subcollection là một collection được liên kết với một document cụ thể.

**Lưu ý**: Bạn có thể truy vấn trên các subcollections có cùng ID collection bằng cách sử dụng Truy vấn nhóm collection (Collection Group Queries).

Bạn có thể tạo một subcollection được gọi là messages cho mọi document room trong collection **rooms** của bạn:



Trong ví dụ này, bạn sẽ tạo một tham chiếu đến một message trong subcollection với code sau:

$document = $db  
    ->collection('rooms')  
    ->document('roomA')  
    ->collection('messages')  
    ->document('message1');

Lưu ý mô hình xen kẽ (alternating pattern) của các collections và documents. collections và documents của bạn phải luôn theo mô hình này. Bạn không thể tham chiếu collections trong collections hoặc documents trong documents.

Subcollections cho phép bạn cấu trúc dữ liệu theo cấp bậc, làm cho dữ liệu dễ truy cập hơn. Để nhận tất cả các messages trong roomA, bạn có thể tạo một tham chiếu collections cho các subcollection messages và tương tác với nó giống như bất kỳ tham chiếu collections nào khác.

Các documents trong Subcollections cũng có thể chứa các subcollections, cho phép bạn thêm dữ liệu lồng. Bạn có thể lồng dữ liệu sâu tới 100 cấp.

**Cảnh báo: Xóa một document không xóa các subcollections của nó!**

**Khi bạn xóa một document có các subcollections, các subcollections đó sẽ không bị xóa. Ví dụ: có thể có một document nằm ở coll / doc / subcoll / suboc mặc dù document coll / doc không còn tồn tại. Nếu bạn muốn xóa documents trong các subcollections khi xóa document gốc, bạn phải thực hiện thủ công, như thể hiện trong Xóa Collections.**

## Các kiểu dữ liệu được hỗ trợ

### Kiểu dữ liệu

Bảng sau liệt kê các loại dữ liệu được Cloud Firestore hỗ trợ. Nó cũng mô tả thứ tự sắp xếp được sử dụng khi so sánh các giá trị cùng loại:

|  |  |  |
| --- | --- | --- |
| Kiểu dữ liệu | Thứ tự sắp xếp | Ghi chú |
| Mảng | Theo giá trị phần tử | Một mảng không thể chứa giá trị mảng khác như là một trong những phần tử của nó.  Trong một mảng, các phần tử duy trì vị trí được gán cho chúng. Khi sắp xếp hai hoặc nhiều mảng, mảng được sắp xếp dựa trên các giá trị phần tử của chúng.  Ví dụ: [1, 2, 3] <[1, 2, 3, 1] <[2]. Mảng [2] có giá trị phần tử đầu tiên lớn nhất. Mảng [1, 2, 3] có các giá trị phần tử bằng ba phần tử đầu tiên của [1, 2, 3, 1] nhưng có chiều dài ngắn hơn. |
| Boolean | false < true | - |
| Bytes | Thứ tự byte | Lên đến 1.048.487 byte (1 MiB - 89 byte). Chỉ 1.500 byte đầu tiên được xem xét bởi các truy vấn. |
| Ngày và giờ | theo thời gian | Khi được lưu trữ trong Cloud Firestore, chỉ chính xác đến micro giây; bất kỳ độ chính xác bổ sung được làm tròn xuống. |
| Số thực | Số | Độ chính xác kép 64 bit, IEEE 754. |
| Điểm địa lý | Theo vĩ độ, sau đó kinh độ | - |
| Integer | Số | 64-bit |
| Map | Theo khóa, sau đó theo giá trị | Đại diện cho một đối tượng được nhúng trong một document. Khi được lập chỉ mục, bạn có thể truy vấn trên các trường con. Nếu bạn loại trừ giá trị này khỏi lập chỉ mục, thì tất cả các trường con cũng được loại trừ khỏi lập chỉ mục.  Thứ tự khóa luôn được sắp xếp. Ví dụ: nếu bạn viết {c: "foo", a: "bar", b: "qux"} thì map được sắp xếp theo khóa và được lưu dưới dạng {a: "foo", b: "bar", c: " qux "}.  Các trường của Map được sắp xếp theo khóa và được so sánh theo các cặp khóa-giá trị, trước tiên là so sánh các khóa và sau đó là các giá trị. Nếu các cặp khóa-giá trị đầu tiên bằng nhau, các cặp khóa-giá trị tiếp theo được so sánh, v.v. Nếu hai Map bắt đầu với các cặp khóa-giá trị giống nhau, thì độ dài Map sẽ được xem xét. Ví dụ: các Map sau theo thứ tự tăng dần:  {a: "aaa", b: "baz"}  {a: "foo", b: "bar"}  {a: "foo", b: "bar", c: "qux"}  {a: "foo", b: "baz"}  {b: "aaa", c: "baz"}  {c: "aaa"} |
| Null | Không | - |
| Reference | Theo các yếu tố đường dẫn (collection, document ID, collection, document ID...) | Ví dụ: projects/[PROJECT\_ID]/databases/[DATABASE\_ID]/documents/[DOCUMENT\_PATH]. |
| Text string | Thứ tự byte được mã hóa UTF-8 | Lên đến 1.048.487 byte (1 MiB - 89 byte). Chỉ 1.500 byte đầu tiên của đại diện UTF-8 được xem xét bởi các truy vấn. |

### Thứ tự của các giá trị kiểu dữ liệu

Khi truy vấn liên quan đến một trường có các giá trị của các loại hỗn hợp, Cloud Firestore sử dụng thứ tự xác định dựa trên các biểu hiện bên trong. Danh sách sau đây cho thấy thứ tự:

1. Null
2. Boolean
3. Integer và số thực, sắp xếp theo thứ tự số
4. Ngày và giờ
5. Text string
6. Byte
7. Cloud Firestore references
8. Điểm địa lý
9. Mảng
10. Map

## SDK và client libraries

Phần này mô tả SDKS và client libraries có sẵn cho API Cloud Firestore. Mặc dù bạn có thể thực hiện các cuộc gọi HTTP và RPC trực tiếp tới API Cloud Firestore, client libraries của Cloud Firestore triển khai cách làm tốt nhất cho bạn và giúp truy cập cơ sở dữ liệu của bạn dễ dàng hơn.

Cloud Firestore hỗ trợ mobile/web SDKs và server client libraries.

### Mobile và web SDKs

Cloud Firestore hỗ trợ SDK cho Android, IOS và Web. Kết hợp với Cloud Firestore security rules và Firebase Auth, SDK mobile và web hỗ trợ kiến trúc ứng dụng không có máy chủ khi khách hàng kết nối trực tiếp với cơ sở dữ liệu Cloud Firestore của bạn. Với kiến trúc không có máy chủ, bạn không cần duy trì máy chủ trung gian giữa các khách hàng và cơ sở dữ liệu Cloud Firestore của bạn.

Mobile và web SDKs cũng hỗ trợ cập nhật thời gian thực(realtime updates) và kiên trì dữ liệu ngoại tuyến(offline data persistence).

Để bắt đầu với Android, iOS hoặc Web SDK, hãy xem phần II: Bắt đầu với Cloud Firestore

### References và resources

Để biết thêm về mỗi SDK, hãy xem các resources sau:

Web:

* [API Reference Documentation](https://firebase.google.com/docs/reference/js/firebase.firestore)
* [Source Code](https://github.com/firebase/firebase-js-sdk/tree/master/packages/firestore)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-js-sdk/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/google-cloud-firestore+javascript)

IOS:

* [API Reference Documentation](https://firebase.google.com/docs/reference/swift/firebasefirestore/api/reference/Classes)
* [Source Code](https://github.com/firebase/firebase-ios-sdk/tree/master/Firestore)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-ios-sdk/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/google-cloud-firestore+ios)

Android:

* [API Reference Documentation](https://firebase.google.com/docs/reference/android/com/google/firebase/firestore/package-summary)
* [Source Code](https://github.com/firebase/firebase-android-sdk/tree/master/firebase-firestore)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-android-sdk/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/google-cloud-firestore+android)

### Server client libraries

Cloud Firestore hỗ trợ các server client libraries cho C #, Go, Java, Node.js, PHP, Python và Ruby. Sử dụng các client libraries này để thiết lập môi trường máy chủ đặc quyền.

Không giống như Mobile và Web SDKs, các client libraries tạo ra một môi trường Cloud Firestore đặc quyền với toàn quyền truy cập vào cơ sở dữ liệu của bạn. Trong môi trường này, các yêu cầu không được đánh giá theo các quy tắc bảo mật Cloud Firestore của bạn. Các máy chủ Cloud Firestore đặc quyền được bảo mật bằng cách sử dụng Quản lý truy cập và nhận dạng đám mây, xem Bảo mật cho các client libraries([Security for server client libraries](https://firebase.google.com/docs/firestore/security/iam)).

Sử dụng các server client libraries cho các tác vụ cơ sở dữ liệu quản trị hoặc nếu bạn thích kiến trúc có máy chủ trung gian giữa người dùng và cơ sở dữ liệu Cloud Firestore của bạn.

Server client libraries của Cloud Firestore có sẵn dưới dạng Firebase Admin SDKs và Google Cloud client libraries. Cả hai bộ thư viện đều cung cấp các tính năng Cloud Firestore giống nhau. Gói Firebase Admin SDKs truy cập vào Cloud Firestore và một số sản phẩm Firebase khác, như Firebase Auth và Firebase Cloud Messaging, trong một thư viện.

### Firebase Admin SDKs

[Firebase Admin SDKs](https://firebase.google.com/docs/admin/setup) gói Google Cloud client libraries cho Cloud Firestore cùng với client libraries và SDK cho một số tính năng Firebase khác. Sử dụng một trong các Admin SDKs, bạn có thể khởi tạo quyền truy cập vào Cloud Firestore và một số dịch vụ khác từ một SDK duy nhất. The Firebase Admin SDKs hỗ trợ truy cập Cloud Firestore trong Java, Python, Node.js và Go.

Để bắt đầu với Firebase Admin SDK, xem [Add the Firebase Admin SDK to Your Server](https://firebase.google.com/docs/admin/setup).

#### References và resources

Java:

* [API Reference Documentation](https://firebase.google.com/docs/reference/admin/java/reference/com/google/firebase/package-summary)
* [Source Code](https://github.com/firebase/firebase-admin-java)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-admin-java/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/firebase-admin+java)

Python:

* [API Reference Documentation](https://firebase.google.com/docs/reference/admin/python/)
* [Source Code](https://github.com/firebase/firebase-admin-python)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-admin-python/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/firebase-admin+python)

Node.js:

* [API Reference Documentation](https://firebase.google.com/docs/reference/admin/node/)
* [Source Code](https://github.com/firebase/firebase-admin-node)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-admin-node/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/node.js+firebase-admin)

Go:

* [API Reference Documentation](https://godoc.org/firebase.google.com/go)
* [Source Code](https://github.com/firebase/firebase-admin-go)
* [GitHub Issue Tracker](https://github.com/firebase/firebase-admin-go/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/firebase-admin+go)

### Google Cloud client libraries

Google Cloud client libraries hỗ trợ truy cập Cloud Firestore trong Java, Python, Node.js, Go, PHP, C # và Ruby. Để bắt đầu với một trong các Google Cloud client libraries, hãy xem [Quickstart using a Server Client Library](https://cloud.google.com/firestore/docs/quickstart-servers).

#### References và resources

PHP:

* [API Reference Documentation](https://googleapis.github.io/google-cloud-php/#/docs/cloud-firestore/latest/firestore/readme)
* [Source Code](https://github.com/googleapis/google-cloud-php/tree/master/Firestore)
* [GitHub Issue Tracker](https://github.com/googleapis/google-cloud-php/labels/api%3A%20firestore)
* [Stack Overflow](https://stackoverflow.com/questions/tagged/google-cloud-firestore+php)

### Tích hợp thư viện của bên thứ ba

Cloud Firestore cung cấp một số tích hợp với các thư viện mã nguồn mở bên cạnh mobile/web SDKs và server client libraries, xem [Library and framework integrations](https://firebase.google.com/docs/firestore/library-integrations).

## Các loại chỉ mục trong Cloud Firestore

Các chỉ mục(Index) là một yếu tố quan trọng trong việc thực hiện cơ sở dữ liệu. Giống như chỉ mục của một cuốn sách ánh xạ các chủ đề trong một cuốn sách sang số trang, một chỉ mục cơ sở dữ liệu ánh xạ các phần tử trong cơ sở dữ liệu đến vị trí của chúng trong cơ sở dữ liệu. Khi bạn gửi đến cơ sở dữ liệu một truy vấn, cơ sở dữ liệu có thể sử dụng một chỉ mục để nhanh chóng tìm kiếm vị trí của các phần tử bạn yêu cầu.

Phần này mô tả hai loại chỉ mục mà Cloud Firestore sử dụng, chỉ mục trường đơn (single-field indexes) và chỉ mục hỗn hợp (composite indexes).

### Một chỉ mục đằng sau mọi truy vấn

Nếu không có chỉ mục nào tồn tại trong một truy vấn, hầu hết các cơ sở dữ liệu sẽ thu thập thông tin qua nội dung phần tử của chúng theo từng phần tử, một quá trình chậm thậm chí sẽ chậm hơn nữa khi cơ sở dữ liệu phát triển. Cloud Firestore đảm bảo hiệu năng truy vấn cao bằng cách sử dụng các chỉ mục cho tất cả các truy vấn. Kết quả là, hiệu suất truy vấn phụ thuộc vào kích thước của tập kết quả chứ không phụ thuộc vào số lượng phần tử trong cơ sở dữ liệu.

### Quản lý chỉ mục ít hơn, phát triển ứng dụng nhiều hơn

Cloud Firestore bao gồm các tính năng giúp giảm lượng thời gian bạn cần để quản lý các chỉ mục. Các chỉ mục cần thiết cho các truy vấn cơ bản nhất sẽ tự động được tạo cho bạn. Khi bạn sử dụng và kiểm tra ứng dụng của mình, Cloud Firestore giúp bạn xác định và tạo các chỉ mục bổ sung mà ứng dụng của bạn yêu cầu.

### Các loại chỉ mục

Cloud Firestore sử dụng hai loại chỉ mục: trường đơn và hỗn hợp. Bên cạnh số lượng các trường được lập chỉ mục, các chỉ mục trường đơn và hỗn hợp khác nhau về cách bạn quản lý chúng.

#### Chỉ mục trường đơn

Một chỉ mục trường đơn lưu trữ một ánh xạ được sắp xếp của tất cả các document (document) trong một collection (collection) có chứa một trường cụ thể. Mỗi bản ghi trong một chỉ mục trường đơn ghi lại giá trị của document cho một trường cụ thể và vị trí của document trong cơ sở dữ liệu. Cloud Firestore sử dụng các chỉ mục này để thực hiện nhiều truy vấn cơ bản. Bạn quản lý các chỉ mục trường đơn bằng cách định cấu hình cài đặt lập chỉ mục tự động và miễn trừ chỉ mục của cơ sở dữ liệu của bạn.

**Lập chỉ mục tự động**

Theo mặc định, Cloud Firestore tự động duy trì các chỉ mục trường đơn cho từng trường trong document (document) và từng trường con trong Map. Cloud Firestore sử dụng các cài đặt mặc định sau cho các chỉ mục trường đơn:

* Đối với mỗi trường không phải mảng và không phải Map, Cloud Firestore xác định hai chỉ mục trường đơn trong phạm vi collection (collection), một ở chế độ tăng dần và một ở chế độ giảm dần.
* Đối với mỗi trường là Map, Cloud Firestore tạo một chỉ mục tăng dần trong phạm vi collection (collection) và một chỉ mục giảm dần cho từng trường con không phải mảng và không phải Map trong Map đó.
* Đối với mỗi trường mảng trong document (document) , Cloud Firestore tạo và duy trì chỉ mục chứa mảng trong phạm vi collection (collection).
* Các chỉ mục trường đơn với phạm vi nhóm collection (collection) không được duy trì theo mặc định.

**Miễn trừ chỉ mục trường đơn**

Bạn có thể miễn trừ một trường khỏi cài đặt lập chỉ mục tự động của bạn bằng cách tạo miễn trừ chỉ mục một trường. Miễn trừ lập chỉ mục ghi đè lên cài đặt chỉ mục tự động trên toàn cơ sở dữ liệu. Một sự miễn trừ có thể cho phép một chỉ mục một trường mà các cài đặt lập chỉ mục tự động của bạn sẽ vô hiệu hóa hoặc vô hiệu hóa một chỉ mục một trường mà lập chỉ mục tự động cho phép. Đối với các trường hợp miễn trừ hữu ích, hãy xem [indexing best practices](https://firebase.google.com/docs/firestore/query-data/index-overview#indexing_best_practices).

Nếu bạn tạo miễn trừ chỉ mục một trường cho trường là Map, các trường con của Map sẽ kế thừa các cài đặt đó. Tuy nhiên, bạn có thể xác định miễn trừ chỉ mục trường đơn cho các trường con cụ thể. Nếu bạn xóa miễn trừ cho trường con, trường con sẽ kế thừa cài đặt miễn trừ của cha mẹ, nếu chúng tồn tại hoặc cài đặt trên toàn cơ sở dữ liệu nếu không có miễn trừ cha mẹ tồn tại.

**Lưu ý:** Miễn trừ chỉ áp dụng cho cài đặt chỉ mục tự động. Một trường được miễn từ lập chỉ mục trường đơn vẫn có thể được lập chỉ mục như là một phần của chỉ mục hỗn hợp.

Để tạo và quản lý miễn trừ chỉ mục một trường, hãy xem [Managing Indexes in Cloud Firestore](https://firebase.google.com/docs/firestore/query-data/indexing#exemptions).

#### Chỉ mục hỗn hợp

Một chỉ mục hỗn hợp lưu trữ một ánh xạ được sắp xếp của tất cả các document trong một collection có chứa nhiều trường cụ thể thay vì chỉ một.

**Lưu ý**: Bạn có thể có tối đa một trường mảng trên mỗi chỉ mục tổng hợp.

Cloud Firestore sử dụng các chỉ mục hỗn hợp để hỗ trợ các truy vấn chưa được hỗ trợ bởi các chỉ mục trường đơn.

Cloud Firestore không tự động tạo các chỉ mục hỗn hợp giống như các chỉ mục trường đơn vì số lượng lớn các kết hợp trường có thể có. Thay vào đó, Cloud Firestore giúp bạn xác định và tạo các chỉ mục hỗn hợp cần thiết khi bạn xây dựng ứng dụng của mình.

Nếu bạn thử truy vấn ở trên mà không tạo chỉ mục cần thiết trước, Cloud Firestore sẽ trả về thông báo lỗi có chứa một liên kết bạn có thể theo dõi để tạo chỉ mục bị thiếu. Điều này xảy ra bất cứ khi nào bạn thử truy vấn không được hỗ trợ lập chỉ mục. Bạn cũng có thể xác định và quản lý các chỉ mục tổng hợp theo cách thủ công bằng cách sử dụng console hoặc bằng cách sử dụng Firebase CLI. Để biết thêm về cách tạo và quản lý chỉ mục tổng hợp, xem [Managing Indexes](https://firebase.google.com/docs/firestore/query-data/indexing).

#### Chế độ chỉ mục và phạm vi truy vấn

Bạn định cấu hình các chỉ mục một trường và hỗn hợp khác nhau, nhưng cả hai đều yêu cầu bạn xác định cấu hình các chế độ chỉ mục và phạm vi truy vấn cho các chỉ mục của bạn.

**chế độ chỉ mục**

Khi bạn xác định một chỉ mục, bạn chọn một chế độ chỉ mục cho từng trường được lập chỉ mục. Chế độ chỉ mục của mỗi trường hỗ trợ các mệnh đề truy vấn cụ thể trên trường đó. Bạn có thể chọn từ các chế độ chỉ mục sau:

|  |  |
| --- | --- |
| **chế độ chỉ mục** | **Mô tả** |
| **Ascending ↑** | Hỗ trợ mệnh đề truy vấn <, <=, ==,> = và > trên trường và hỗ trợ sắp xếp kết quả theo thứ tự tăng dần dựa trên giá trị trường này. |
| **Descending↓** | Hỗ trợ mệnh đề truy vấn <, <=, ==,> = và > trên trường và hỗ trợ sắp xếp kết quả theo thứ tự giảm dần dựa trên giá trị trường này. |
| **Array‑contains** | Hỗ trợ mệnh đề truy vấn [array\_contains](https://firebase.google.com/docs/firestore/query-data/queries#array_contains)  trên trường. |

**phạm vi truy vấn**

Mỗi chỉ mục nằm trong phạm vi của một collection (collection) hoặc một nhóm collection (collection group). Điều này được gọi là phạm vi truy vấn của chỉ mục:

**Phạm vi của collection**

Cloud Firestore tạo các chỉ mục với phạm vi collection theo mặc định. Các chỉ mục này hỗ trợ các truy vấn trả về kết quả từ một collection.

**Phạm vi của nhóm các collection**

Một nhóm collection bao gồm tất cả các collection có cùng ID collection. Để chạy truy vấn nhóm collection trả về kết quả được lọc hoặc đã sắp xếp từ nhóm collection, bạn phải tạo một chỉ mục tương ứng với phạm vi nhóm collection.

### Ví dụ lập chỉ mục

Bằng cách tự động tạo các chỉ mục trường đơn cho bạn, Cloud Firestore cho phép ứng dụng của bạn nhanh chóng hỗ trợ các truy vấn cơ sở dữ liệu cơ bản nhất. Các chỉ mục trường đơn cho phép bạn thực hiện các truy vấn đơn giản dựa trên các giá trị trường và các bộ so sánh <, <=, ==,> = và >. Đối với các trường mảng, chúng cho phép bạn thực hiện các truy vấn array\_contains.

Để minh họa, Xem xét các ví dụ sau từ quan điểm của việc tạo chỉ mục. Đoạn code sau tạo một vài document city trong collection cities và đặt các trường name, state, country, capital, population và tags cho mỗi document:

Web:

var citiesRef = db.collection("cities");  
  
citiesRef.doc("SF").set({  
    name: "San Francisco", state: "CA", country: "USA",  
    capital: false, population: 860000,  
    regions: ["west\_coast", "norcal"] });  
citiesRef.doc("LA").set({  
    name: "Los Angeles", state: "CA", country: "USA",  
    capital: false, population: 3900000,  
    regions: ["west\_coast", "socal"] });  
citiesRef.doc("DC").set({  
    name: "Washington, D.C.", state: null, country: "USA",  
    capital: true, population: 680000,  
    regions: ["east\_coast"] });  
citiesRef.doc("TOK").set({  
    name: "Tokyo", state: null, country: "Japan",  
    capital: true, population: 9000000,  
    regions: ["kanto", "honshu"] });  
citiesRef.doc("BJ").set({  
    name: "Beijing", state: null, country: "China",  
    capital: true, population: 21500000,  
    regions: ["jingjinji", "hebei"] });

Giả sử cài đặt lập chỉ mục tự động mặc định, Cloud Firestore cập nhật một chỉ mục trường đơn tăng dần cho mỗi trường không phải mảng, một chỉ mục trường đơn giảm dần cho mỗi trường không phải mảng và một chỉ mục trường array-contains cho trường mảng. Mỗi hàng trong bảng sau đây biểu thị một mục trong chỉ mục một trường:

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| cities | ↑ name | Collection |
| cities | ↑ state | Collection |
| cities | ↑ country | Collection |
| cities | ↑ capital | Collection |
| cities | ↑ population | Collection |
| cities | ↓ name | Collection |
| cities | ↓ state | Collection |
| cities | ↓ country | Collection |
| cities | ↓ capital | Collection |
| cities | ↓ population | Collection |
| cities | array-contains regions | Collection |

#### Các truy vấn được hỗ trợ bởi các chỉ mục trường đơn

Sử dụng các chỉ mục trường đơn được tạo tự động này, bạn có thể chạy các truy vấn đơn giản như sau:

Web:

citiesRef.where("state", "==", "CA")  
citiesRef.where("population", "<", 100000)  
citiesRef.where("name", ">=", "San Francisco")

Chỉ mục Array\_contains cho phép bạn truy vấn trường mảng regions:

Web:

citiesRef.where("regions", "array-contains", "west\_coast")

Bạn cũng có thể tạo các truy vấn ghép dựa trên các đẳng thức (==):

Web:

citiesRef.where("state", "==", "CO").where("name", "==", "Denver")  
citiesRef.where("country", "==", "USA").where("capital", "==", false).where("state", "==", "CA").where("population", "==", 860000)

Nếu bạn cần chạy truy vấn hỗn hợp sử dụng so sánh phạm vi (<, <=,> hoặc> =) hoặc nếu bạn cần sắp xếp theo một trường khác, bạn phải tạo một chỉ mục tổng hợp cho truy vấn đó.

#### Các truy vấn được hỗ trợ bởi các chỉ mục tổng hợp

Cloud Firestore sử dụng các chỉ mục tổng hợp để hỗ trợ các truy vấn hỗn hợp chưa được hỗ trợ bởi các chỉ mục trường đơn. Ví dụ: bạn sẽ cần một chỉ mục tổng hợp cho các truy vấn sau:

Web:

citiesRef.where("country", "==", "USA").orderBy("population", "asc")  
citiesRef.where("country", "==", "USA").where("population", "<", 3800000)  
citiesRef.where("country", "==", "USA").where("population", ">", 690000)

Các truy vấn này yêu cầu chỉ mục tổng hợp dưới đây. Lưu ý rằng vì truy vấn sử dụng một đẳng thức cho trường country , chế độ chỉ mục của trường có thể giảm dần hoặc tăng dần. Theo mặc định, các truy vấn bất đẳng thức sẽ áp dụng thứ tự sắp xếp tăng dần dựa trên trường trong mệnh đề bất đẳng thức.

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| cities | ↑ (or ↓) country, ↑ population | Collection |

Nếu bạn muốn chạy các truy vấn tương tự nhưng với thứ tự sắp xếp giảm dần, thì bạn cần một chỉ mục tổng hợp bổ sung theo hướng giảm dần cho population:

Web:

citiesRef.where("country", "==", "USA").orderBy("population", "desc")  
  
citiesRef.where("country", "==", "USA")  
         .where("population", "<", 3800000)  
         .orderBy("population", "desc")  
  
citiesRef.where("country", "==", "USA")  
         .where("population", ">", 690000)  
         .orderBy("population", "desc")

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| cities | ↑ country, ↑ population | Collection |
| cities | ↑ country, ↓ population | Collection |

Bạn cũng cần tạo một chỉ mục tổng hợp nếu bạn muốn kết hợp truy vấn Array\_contains với các mệnh đề bổ sung.

Web:

citiesRef.where("regions", "array\_contains", "east\_coast")  
         .where("capital", "==", true)

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| cities | **array-contains** tags, ↑ (or ↓) capital | Collection |

#### Các truy vấn được hỗ trợ bởi các chỉ mục nhóm collection

Để thể hiện một chỉ mục với phạm vi nhóm collection, hãy tưởng tượng bạn thêm một collection phụ (sub-collection) landmarks vào một số document của city:

Web:

var citiesRef = db.collection("cities");  
  
citiesRef.doc("SF").collection("landmarks").doc().set({  
    name: "Golden Gate Bridge",  
    category : "bridge" });  
citiesRef.doc("SF").collection("landmarks").doc().set({  
    name: "Golden Gate Park",  
    category : "park" });  
  
citiesRef.doc("DC").collection("landmarks").doc().set({  
    name: "National Gallery of Art",  
    category : "museum" });  
citiesRef.doc("DC").collection("landmarks").doc().set({  
    name: "National Mall",  
    category : "park" });

Sử dụng chỉ mục trường đơn sau đây với phạm vi collection, bạn có thể truy vấn park landmarks trong một city:

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| landmarks | ↑ (or ↓) category | Collection |

Web:

citiesRef.doc("SF").collection("landmarks").where("category", "==", "park")

Bây giờ, hãy tưởng tượng rằng bạn quan tâm đến các park landmarks trên tất cả các city. Để chạy truy vấn này trên nhóm collection bao gồm tất cả các collection landmarks, bạn phải bật chỉ mục trường đơn landmarks với phạm vi nhóm collection:

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| landmarks | ↑ (or ↓) category | Nhóm collection |

Với chỉ mục này được bật, bạn có thể truy vấn nhóm collection landmarks:

Web:

var landmarksGroupRef = db.collectionGroup("landmarks");  
  
landmarksGroupRef.where("category", "==", "park")

Để chạy truy vấn nhóm collection trả về kết quả được lọc hoặc đã sắp xếp, bạn phải bật một chỉ mục tổng hợp hoặc trường đơn tương ứng với phạm vi nhóm collection. Tuy nhiên, các truy vấn nhóm collection không lọc hoặc sắp xếp kết quả, không yêu cầu bất kỳ định nghĩa chỉ mục bổ sung nào.

Ví dụ: bạn có thể chạy truy vấn nhóm collection sau mà không cần bật chỉ mục bổ sung:

Web:

db.collectionGroup("landmarks").get()

### Lập chỉ mục và chi phí

Các chỉ mục đóng góp vào chi phí lưu trữ của ứng dụng của bạn. Để biết thêm về cách tính kích thước lưu trữ cho các chỉ mục, hãy xem [Index entry size](https://firebase.google.com/docs/firestore/storage-size#index-entry-size).

#### Lợi dụng hợp nhất lập chỉ mục

Mặc dù Cloud Firestore sử dụng một chỉ mục cho mọi truy vấn, nhưng nó không nhất thiết yêu cầu một chỉ mục cho mỗi truy vấn. Đối với các truy vấn có nhiều mệnh đề bằng (==) và, tùy ý, một mệnh đề orderBy, Cloud Firestore có thể sử dụng lại các chỉ mục hiện có. Cloud Firestore có thể hợp nhất các chỉ mục cho các bộ lọc bằng đơn giản để xây dựng các chỉ mục tổng hợp cần thiết cho các truy vấn bằng lớn hơn.

Bạn có thể giảm chi phí lập chỉ mục bằng cách xác định các tình huống mà bạn có thể tận dụng lợi thế của việc hợp nhất chỉ mục. Ví dụ: hãy tưởng tượng collection restaurants cho ứng dụng xếp hạng nhà hàng:



Bây giờ, hãy tưởng tượng ứng dụng này sử dụng các truy vấn như những câu dưới đây. Lưu ý rằng ứng dụng sử dụng kết hợp các mệnh đề bằng cho category, city và editors\_pick trong khi luôn sắp xếp theo tăng dần star\_rating:

Web:

db.collection("restaurants").where("category", "==", "burgers")  
                            .orderBy("star\_rating")  
  
db.collection("restaurants").where("city", "==", "San Francisco")  
                            .orderBy("star\_rating")  
  
db.collection("restaurants").where("category", "==", "burgers")  
                            .where("city", "==", "San Francisco")  
                            .orderBy("star\_rating")  
  
db.collection("restaurants").where("category", "==", "burgers")  
                            .where("city", "==" "San Francisco")  
                            .where("editors\_pick", "==", true )  
                            .orderBy("star\_rating")

Bạn có thể tạo một chỉ mục cho mỗi truy vấn:

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| restaurants | ↑ category, ↑star\_rating | collection |
| restaurants | ↑city, ↑star\_rating | collection |
| restaurants | ↑category, ↑city, ↑star\_rating | collection |
| restaurants | ↑category, ↑city, ↑editors\_pick,↑star\_rating | collection |

Một giải pháp tốt hơn, bạn có thể giảm số lượng chỉ mục bằng cách tận dụng khả năng của Cloud Firestore để hợp nhất các chỉ mục cho các mệnh đề bằng:

|  |  |  |
| --- | --- | --- |
| Collection | Trường được lập chỉ mục | Phạm vi truy vấn |
| restaurants | ↑ category, ↑star\_rating | collection |
| restaurants | ↑city, ↑star\_rating | collection |
| restaurants | ↑editors\_pick, ↑star\_rating | collection |

Bộ chỉ mục này không chỉ nhỏ hơn mà còn hỗ trợ một truy vấn bổ sung:

Web:

db.collection("restaurants").where("editors\_pick", "==", true)  
                            .orderBy("star\_rating")

### Giới hạn chỉ mục

Các giới hạn sau áp dụng cho các chỉ mục. Đối với tất cả hạn ngạch và giới hạn, xem [Quotas and Limits](https://firebase.google.com/docs/firestore/quotas).

|  |  |
| --- | --- |
| **Giới hạn** | **Chi tiết** |
| Số lượng chỉ mục tổng hợp tối đa cho cơ sở dữ liệu | 200 |
| Số lượng miễn trừ chỉ mục trường đơn tối đa cho cơ sở dữ liệu | 200 |
| Số lượng mục nhập chỉ mục tối đa cho mỗi document | 40.000  Số lượng mục nhập chỉ mục là tổng của các mục sau đây cho một document:   * Số lượng mục chỉ mục trường đơn * Số lượng mục chỉ mục tổng hợp |
| Kích thước tối đa của một mục nhập chỉ mục | 7,5 KiB  Để xem cách Cloud Firestore tính toán kích thước mục nhập chỉ mục, hãy xem [index entry size](https://firebase.google.com/docs/firestore/storage-size#index-entry-size). |
| Tổng kích thước tối đa của các mục nhập chỉ mục của document | 8 MiB  Tổng kích thước là tổng của các tổng sau đây cho một document:   * Tổng kích thước của các mục chỉ mục trường đơn của document * Tổng kích thước của các mục chỉ mục tổng hợp của document |
| Kích thước tối đa của giá trị trường được lập chỉ mục | 1500 byte  Giá trị trường trên 1500 byte bị cắt ngắn. Các truy vấn liên quan đến các giá trị trường bị cắt bớt có thể trả về kết quả không nhất quán. |

### Mẫu tốt nhất cho lập chỉ mục

Đối với hầu hết các ứng dụng, bạn có thể dựa vào lập chỉ mục tự động và các liên kết thông báo lỗi để quản lý các chỉ mục của bạn. Tuy nhiên, bạn có thể muốn thêm các trường hợp miễn trừ trường đơn trong các trường hợp sau:

|  |  |
| --- | --- |
| **Trường hợp** | **Mô tả** |
| Trường chuỗi lớn | Nếu bạn có trường chuỗi mà thường giữ các giá trị chuỗi dài mà bạn không sử dụng để truy vấn, bạn có thể cắt giảm chi phí lưu trữ bằng cách miễn cho trường khỏi lập chỉ mục. |
| Tốc độ ghi cao vào collection chứa các document có giá trị tuần tự | Nếu bạn lập chỉ mục một trường tăng hoặc giảm tuần tự giữa các document trong collection, như timestamp, thì tốc độ ghi tối đa vào collection là 500 ghi mỗi giây. Nếu bạn không truy vấn dựa trên trường có các giá trị tuần tự, bạn có thể miễn trừ trường lập chỉ mục để vượt qua giới hạn này.  Ví dụ, trong trường hợp sử dụng IoT có tốc độ ghi cao, một collection chứa các document có trường timestamp có thể đạt tới giới hạn 500 ghi mỗi giây. |
| Trường có mảng lớn hoặc Map | Trường có mảng lớn hoặc Map có thể đạt đến giới hạn 20.000 mục nhập chỉ mục trên mỗi document. Nếu bạn không truy vấn dựa trên một trường có mảng lớn hoặc Map, bạn nên miễn cho nó lập chỉ mục. |

## Vị trí cơ sở dữ liệu của Cloud Firestore

Trước khi sử dụng Cloud Firestore, bạn phải chọn một vị trí cho cơ sở dữ liệu của mình. Để giảm độ trễ và tăng tính khả dụng, hãy lưu trữ dữ liệu của bạn gần với người dùng và dịch vụ cần dữ liệu đó. Cài đặt vị trí này là [default Google Cloud Platform (GCP) resource location](https://firebase.google.com/docs/firestore/locations#default-cloud-location) trong dự án của bạn.

Lưu ý rằng default GCP resource location của bạn có thể đã được đặt trước đó, trong khi tạo dự án hoặc khi thiết lập một dịch vụ khác yêu cầu cài đặt vị trí (xem bên dưới).

**Quan trọng: default GCP resource location của bạn được sử dụng cho các dịch vụ GCP trong dự án yêu cầu cài đặt vị trí, cụ thể là default Cloud Storage bucket và App Engine app của bạn (được yêu cầu nếu bạn sử dụng Cloud Scheduler).**

### Default GCP resource location

Một số dịch vụ khả dụng cho ứng dụng của bạn yêu cầu cài đặt vị trí, được gọi là default Google Cloud Platform (GCP) resource location. Vị trí này là nơi dữ liệu của bạn được lưu trữ cho các dịch vụ GCP yêu cầu cài đặt vị trí.

Cảnh báo: Đặt vị trí cho một trong các dịch vụ sau cũng đặt vị trí cho các dịch vụ khác. Sau khi bạn đặt default GCP resource location của dự án, bạn không thể thay đổi nó.

Các sản phẩm sau đây có chung default GCP resource location:

* **Cloud Firestore**

Lưu ý rằng default GCP resource location của bạn không áp dụng cho Firebase Realtime Database.

* **Cloud Storage**

Lưu ý rằng default GCP resource location của bạn chỉ áp dụng cho nhóm Cloud Storage mặc định của bạn. Nếu bạn đang sử dụng gói Blaze, bạn có thể tạo nhiều nhóm, mỗi nhóm có vị trí riêng.

* **Google App Engine (GAE) app**

Nếu bạn thiết lập App Engine app, vị trí của ứng dụng sẽ chia sẻ default GCP resource location của bạn. Lưu ý rằng nếu bạn sử dụng Cloud Scheduler (ví dụ: để chạy scheduled functions)), bạn bắt buộc phải App Engine app trong dự án của mình.

### Các loại vị trí

Bạn có thể lưu trữ dữ liệu Cloud Firestore của mình ở một vị trí đa vùng hoặc một vùng.

Lưu ý rằng nếu bạn đã có App Engine app với vị trí là us-central hoặc europe-west, cơ sở dữ liệu Cloud Firestore của bạn sẽ được coi là đa khu vực.

#### Vị trí đa vùng

Một vị trí đa khu vực là một khu vực địa lý chung, chẳng hạn như Hoa Kỳ. Dữ liệu ở một vị trí đa vùng được nhân rộng ở nhiều vùng. Trong một khu vực, dữ liệu được nhân rộng giữa các khu vực.

Chọn một vị trí đa vùng để tối đa hóa tính khả dụng và độ bền của cơ sở dữ liệu của bạn. Các vị trí đa vùng có thể chịu được mất toàn bộ khu vực và duy trì tính khả dụng mà không mất dữ liệu.

Firebase hỗ trợ các GCP resource locations đa vùng sau:

|  |  |  |
| --- | --- | --- |
| **Tên đa vùng** | **Mô tả đa vùng** | **Khu vực cấu thành** |
| eur3 hoặc europe-west | Châu Âu | europe-west1, europe-west4 |
| nam5 hoặc us-central | Hoa Kỳ | us-central1, us-central2 (Oklahoma — khu vực GCP tư nhân) |

#### Vị trí khu vực

Vị trí khu vực là một vị trí địa lý cụ thể, chẳng hạn như Nam Carolina. Dữ liệu ở một vị trí khu vực được nhân rộng ở nhiều khu vực trong một vùng. Tất cả các vị tri1 trong khu vực được tách ra từ các vị trí khác trong khu vực ít nhất 100 dặm.

Chọn một vị trí khu vực với chi phí thấp hơn, cho độ trễ ghi thấp hơn nếu ứng dụng của bạn nhạy cảm với độ trễ hoặc cho [co-location with other GCP resources](https://cloud.google.com/about/locations/#products-available-by-region).

Firebase hỗ trợ các vị trí tài nguyên GCP khu vực sau:

|  |  |
| --- | --- |
| **Tên khu vực** | **Mô tả khu vực** |
| **Bắc Mỹ** | |
| us-west2 | Los Angeles |
| northamerica-northeast1 | Montréal |
| us-east1 | South Carolina |
| us-east4 | Northern Virginia |
| **Nam Mỹ** | |
| southamerica-east1 | São Paulo |
| **Châu Âu** | |
| europe-west2 | London |
| europe-west3 | Frankfurt |
| europe-west6 | Zürich |
| **Châu Á** | |
| asia-south1 | Mumbai |
| asia-east2 | Hong Kong |
| asia-northeast1 | Tokyo |
| asia-northeast2 | Osaka |
| **Châu Úc** | |
| australia-southeast1 | Sydney |

### Vị trí SLA

Loại vị trí Cloud Firestore của bạn xác định tỷ lệ phần trăm thời gian hoạt động của Service Level Agreement (SLA):

|  |  |
| --- | --- |
| Dịch vụ được bảo hiểm | Tỷ lệ phần trăm thời gian hoạt động hàng tháng |
| Cloud Firestore Đa vùng | >= 99.999% |
| Cloud Firestore Khu vực | >= 99.99% |

### Chi phí vị trí

Vị trí Cloud Firestore của bạn xác định chi phí cho các hoạt động của cơ sở dữ liệu.

Để được giải thích toàn diện về giá cả theo vùng và từng loại vùng, xem [Understand Cloud Firestore billing](https://firebase.google.com/docs/firestore/pricing).

### Xem cài đặt của vị trí dự án của bạn

Trong Firebase console, đi đến [Project settings](https://console.firebase.google.com/project/_/settings/general/) của bạn.

# Thêm và quản lý dữ liệu:

## Cấu trúc của dữ liệu

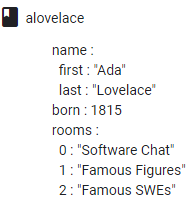
Khi bạn xây dựng dữ liệu của mình trong Cloud Firestore, bạn có một vài tùy chọn khác nhau:

* Các document
* Nhiều collection
* Collection trong document

Hãy xem xét các ưu điểm của từng tùy chọn vì chúng liên quan đến trường hợp sử dụng của bạn. Một vài cấu trúc ví dụ cho dữ liệu phân cấp được nêu trong hướng dẫn này.

### Dữ liệu lồng nhau trong document

Bạn có thể lồng các đối tượng phức tạp như mảng hoặc bản đồ trong document.

**Ưu điểm**: Nếu bạn có danh sách dữ liệu đơn giản, cố định mà bạn muốn giữ trong document của mình, việc này dễ dàng thiết lập và sắp xếp hợp lý cấu trúc dữ liệu của bạn.

**Hạn chế**: Đây không phải là khả năng mở rộng như các tùy chọn khác, đặc biệt là nếu dữ liệu của bạn mở rộng theo thời gian. Với danh sách lớn hơn hoặc đang phát triển, document cũng phát triển, điều này có thể dẫn đến thời gian truy xuất document chậm hơn.

**Trường hợp sử dụng có thể là gì?** Ví dụ, trong một ứng dụng trò chuyện, bạn có thể lưu trữ 3 phòng trò chuyện được truy cập gần đây nhất của người dùng dưới dạng danh sách lồng nhau trong hồ sơ của họ.

### Các collection con (Subcollections)

Bạn có thể tạo các collection trong các document khi bạn có dữ liệu có thể mở rộng theo thời gian.

**Ưu điểm**: Khi danh sách của bạn tăng lên, kích thước của document gốc không thay đổi. Bạn cũng có được khả năng truy vấn đầy đủ trên các collection con.

**Hạn chế**: Bạn không thể dễ dàng xóa các collection con.

**Trường hợp sử dụng có thể là gì?** Ví dụ, trong cùng một ứng dụng trò chuyện, bạn có thể tạo collection người dùng hoặc tin nhắn trong document phòng trò chuyện.

### Collection cấp gốc (root-level)

Tạo các collection ở cấp gốc của cơ sở dữ liệu của bạn để sắp xếp các tập dữ liệu khác nhau.

**Ưu điểm:** Collection cấp gốc cung cấp tính linh hoạt và khả năng mở rộng cao nhất, cùng với truy vấn mạnh mẽ trong mỗi collection.

**Hạn chế:** Nhận dữ liệu được phân cấp tự nhiên có thể ngày càng phức tạp khi cơ sở dữ liệu của bạn phát triển.

**Trường hợp sử dụng có thể là gì?** Ví dụ, trong cùng một ứng dụng trò chuyện, bạn có thể tạo một collection cho người dùng và một collection khác cho phòng và tin nhắn.

## Thêm dữ liệu vào Cloud Firestore

Có một số cách để ghi dữ liệu lên Cloud Firestore:

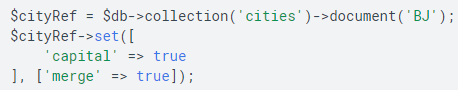
* Đặt dữ liệu của document trong collection, chỉ định rõ ràng định danh document.
* Thêm một document mới vào một collection. Trong trường hợp này, Cloud Firestore tự động tạo định danh document.
* Tạo một document trống với mã định danh được tạo tự động và gán dữ liệu cho nó sau.

### Tạo document

Để tạo hoặc ghi đè lên một document, sử dụng phương thức set() – minh họa bằng code PHP:



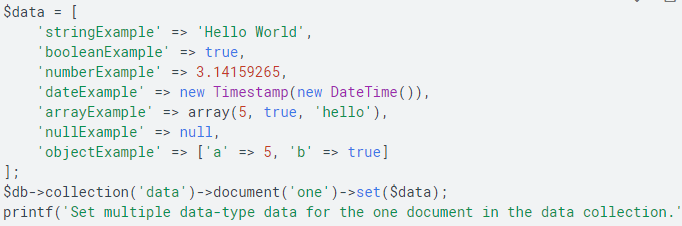
Nếu document không tồn tại, nó sẽ được tạo. Nếu document tồn tại, nội dung của nó sẽ được ghi đè bằng dữ liệu mới được cung cấp, trừ khi bạn chỉ định rằng dữ liệu sẽ được hợp nhất vào document hiện có, như sau:



Nếu bạn không chắc liệu document có tồn tại hay không, hãy chuyển tùy chọn hợp nhất dữ liệu mới với bất kỳ document hiện có nào để tránh ghi đè lên toàn bộ document.

Các kiểu dữ liệu

Cloud Firestore cho phép bạn viết nhiều loại dữ liệu bên trong một document, bao gồm các chuỗi, booleans, số, ngày, null và các mảng và đối tượng lồng nhau. Cloud Firestore luôn lưu trữ các số là double, bất kể bạn sử dụng loại số nào trong mã của mình.



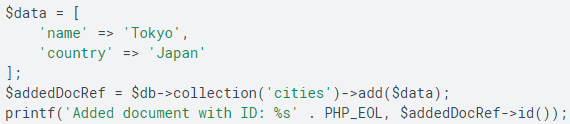
Đối tượng tùy chỉnh

Việc sử dụng các đối tượng Java map để thể hiện các document của bạn thường không thuận tiện lắm, vì vậy Cloud Firestore cũng hỗ trợ viết các đối tượng Java của riêng bạn bằng các lớp tùy chỉnh. Cloud Firestore sẽ chuyển đổi nội bộ các đối tượng thành các kiểu dữ liệu được hỗ trợ.

**Quan trọng***: Mỗi lớp tùy chỉnh phải có một hàm tạo công khai không có đối số. Ngoài ra, lớp phải bao gồm một getter công khai cho mỗi tài sản*.

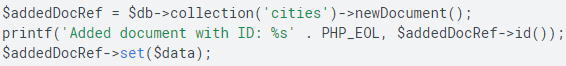
Thêm một document

Khi bạn sử dụng set() để tạo document, bạn phải chỉ định ID cho document cần tạo. Ví dụ: 

Nhưng đôi khi không có ID có ý nghĩa cho document và sẽ thuận tiện hơn khi để Cloud Firestore tự động tạo ID cho bạn. Bạn có thể làm điều này bằng cách gọi add():

**Quan trọng**: *Không giống như "ID đẩy" trong Cơ sở dữ liệu thời gian thực Firebase, ID được tạo tự động của Cloud Firestore không cung cấp bất kỳ thứ tự tự động nào. Nếu bạn muốn có thể đặt hàng document của mình trước ngày tạo, bạn nên lưu dấu thời gian dưới dạng một trường trong document.*

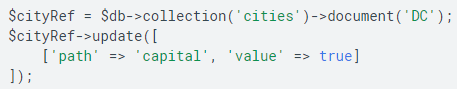
Trong một số trường hợp, có thể hữu ích khi tạo tham chiếu document với ID được tạo tự động, sau đó sử dụng tham chiếu sau. Đối với trường hợp sử dụng này, bạn có thể gọi doc():



Đằng sau hậu trường, .add(...) và .doc().Set(...) hoàn toàn tương đương, vì vậy bạn có thể sử dụng bất kỳ cách nào thuận tiện hơn.

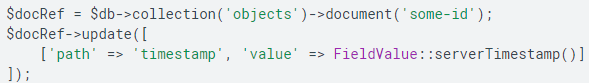
### Cập nhật một document

Để cập nhật một số trường của document mà không ghi đè lên toàn bộ document, hãy sử dụng phương thức update ():



Mốc thời gian máy chủ

Bạn có thể đặt trường trong document của mình thành dấu thời gian của máy chủ để theo dõi khi máy chủ nhận được bản cập nhật.



Cập nhật các trường trong các đối tượng lồng nhau

Nếu document của bạn chứa các đối tượng lồng nhau, bạn có thể sử dụng "ký hiệu dấu chấm" để tham chiếu các trường lồng nhau trong document khi bạn gọi update ():



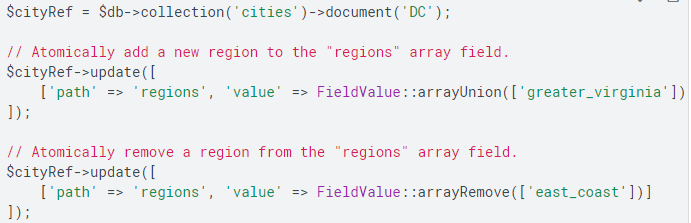
**Quan trọng***: Khi cố gắng cập nhật các trường lồng nhau trong document, điều quan trọng là phải nhận thức được sự khác biệt về ngữ nghĩa tinh tế giữa các tùy chọn có sẵn.*

Ký hiệu dấu chấm cho phép bạn cập nhật một trường lồng đơn mà không ghi đè lên trường lồng nhau khác. Nếu bạn cập nhật trường lồng nhau mà không có ký hiệu chấm, bạn sẽ ghi đè lên toàn bộ trường bản đồ, ví dụ:



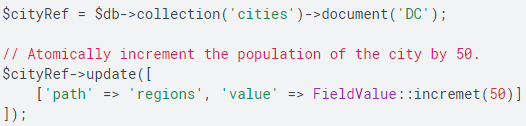
Cập nhật các phần tử trong một mảng

Nếu document của bạn chứa một trường mảng, bạn có thể sử dụng ArrayUnion () và ArrayRemove () để thêm và xóa các phần tử. ArrayUnion () thêm các phần tử vào một mảng nhưng chỉ các phần tử chưa có. mảngRemove () loại bỏ tất cả các phiên bản của từng phần tử đã cho.



Tăng một giá trị số

Bạn có thể tăng hoặc giảm giá trị trường số như trong ví dụ sau. Một hoạt động gia tăng làm tăng hoặc giảm giá trị hiện tại của một trường theo số tiền đã cho. Nếu trường không tồn tại hoặc nếu giá trị trường hiện tại không phải là giá trị số, thì thao tác sẽ đặt trường thành giá trị đã cho.



Các hoạt động gia tăng rất hữu ích để thực hiện các bộ đếm, nhưng hãy nhớ rằng bạn có thể cập nhật một document duy nhất một lần mỗi giây.

## Giao dịch và tập ghi

Cloud Firestore hỗ trợ các hoạt động nguyên tử để đọc và ghi dữ liệu. Trong một tập hợp các hoạt động nguyên tử, tất cả các hoạt động đều thành công hoặc không có hoạt động nào được áp dụng. Có hai loại hoạt động nguyên tử trong Cloud Firestore:

* **Các giao dịch (Transactions):** giao dịch là một tập hợp các hoạt động đọc và ghi trên một hoặc nhiều document.
* **Các tập ghi (batched writes):** một tập ghi là một tập hợp các thao tác ghi trên một hoặc nhiều document.

Mỗi giao dịch hoặc đợt ghi có thể ghi tối đa 500 document. Để biết thêm các giới hạn liên quan đến việc viết, hãy xem Hạn ngạch và Giới hạn.

### Cập nhật dữ liệu với các giao dịch

Sử dụng các thư viện máy khách Cloud Firestore, bạn có thể nhóm nhiều hoạt động thành một giao dịch. Giao dịch rất hữu ích khi bạn muốn cập nhật giá trị của một trường dựa trên giá trị hiện tại của nó hoặc giá trị của một số trường khác.

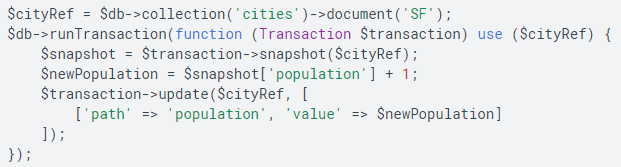
Một giao dịch bao gồm bất kỳ số lượng các hoạt động get () theo sau bởi bất kỳ số lượng các hoạt động ghi như set(), update() hoặc delete(). Trong trường hợp chỉnh sửa đồng thời, Cloud Firestore sẽ chạy lại toàn bộ giao dịch. Ví dụ: nếu một giao dịch đọc document và một khách hàng khác sửa đổi bất kỳ document nào trong số đó, Cloud Firestore sẽ thử lại giao dịch. Tính năng này đảm bảo rằng giao dịch chạy trên dữ liệu cập nhật và nhất quán.

Giao dịch không bao giờ áp dụng một phần ghi. Tất cả các văn bản thực hiện vào cuối một giao dịch thành công.

Khi sử dụng giao dịch, lưu ý rằng:

* Hoạt động đọc phải có trước hoạt động viết.
* Một chức năng gọi một giao dịch (chức năng giao dịch) có thể chạy nhiều lần nếu một chỉnh sửa đồng thời ảnh hưởng đến một document mà giao dịch đọc.
* Chức năng giao dịch không nên trực tiếp sửa đổi trạng thái ứng dụng.
* Giao dịch sẽ thất bại khi khách hàng ngoại tuyến.

Ví dụ sau đây cho thấy cách tạo và chạy giao dịch:



Truyền thông tin ra khỏi giao dịch

Không sửa đổi trạng thái ứng dụng bên trong các chức năng giao dịch của bạn. Làm như vậy sẽ giới thiệu vấn đề đồng thời, bởi vì chức năng giao dịch có thể chạy nhiều lần và không được đảm bảo để chạy trên thread UI. Thay vào đó, chuyển thông tin bạn cần ra khỏi chức năng giao dịch của bạn. Ví dụ sau được xây dựng dựa trên ví dụ trước để chỉ ra cách truyền thông tin ra khỏi giao dịch:



Giao dịch thất bại

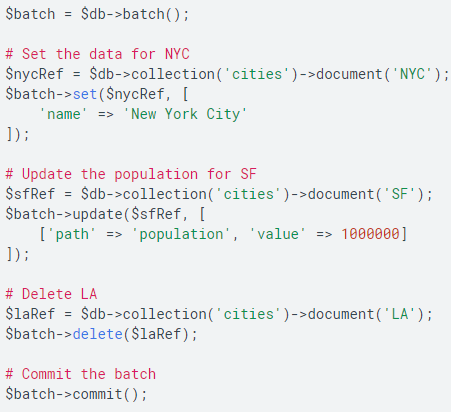
Một giao dịch có thể thất bại vì những lý do sau:

* Giao dịch chứa các hoạt động đọc sau khi viết hoạt động. Hoạt động đọc phải luôn luôn đến trước bất kỳ hoạt động ghi.
* Giao dịch đọc một document đã được sửa đổi bên ngoài giao dịch. Trong trường hợp này, giao dịch sẽ tự động chạy lại. Giao dịch được thử lại một số lần hữu hạn.
* Giao dịch vượt quá kích thước yêu cầu tối đa là 10 MiB.

Kích thước giao dịch phụ thuộc vào kích thước của document và mục nhập chỉ mục được sửa đổi bởi giao dịch. Đối với thao tác xóa, phần này bao gồm kích thước của document đích và kích thước của các mục chỉ mục bị xóa để đáp ứng với thao tác.

Một giao dịch thất bại trả về một lỗi và không ghi bất cứ điều gì vào cơ sở dữ liệu. Bạn không cần phải khôi phục giao dịch; Cloud Firestore thực hiện điều này tự động.

### Các tập ghi

Nếu bạn không cần phải đọc bất kỳ document nào trong bộ thao tác của mình, bạn có thể thực thi nhiều thao tác ghi dưới dạng một lô có chứa bất kỳ kết hợp nào của các hoạt động set(), update()hoặc delete(). Một loạt các bài viết hoàn thành nguyên tử và có thể ghi vào nhiều document. Ví dụ sau đây cho thấy cách xây dựng và cam kết một đợt ghi:

Một bài viết theo đợt có thể chứa tới 500 thao tác. Mỗi thao tác trong lô được tính riêng cho việc sử dụng Cloud Firestore của bạn. Trong một thao tác ghi, các trường biến đổi như serverTimestamp, arrayUnion, và increment mỗi số như một thao tác bổ sung.

Giống như các giao dịch, viết theo đợt là nguyên tử. Không giống như các giao dịch, việc ghi theo lô không cần phải đảm bảo rằng các document đã đọc vẫn chưa được sửa đổi dẫn đến ít trường hợp thất bại hơn. Họ không phải là đối tượng thử lại hoặc thất bại từ quá nhiều lần thử lại. Batched write thực thi ngay cả khi thiết bị của người dùng ngoại tuyến.

Lưu ý*: Để nhập dữ liệu hàng loạt, hãy sử dụng thư viện máy khách với ghi riêng lẻ. Batched write thực hiện tốt hơn ghi tuần tự nhưng không tốt hơn ghi song song. Bạn nên sử dụng thư viện máy chủ cho các hoạt động dữ liệu hàng loạt và không phải là SDK di động / web.*

### Xác thực dữ liệu cho các hoạt động vi mô

Đối với thư viện ứng dụng khách trên thiết bị di động / web, bạn có thể xác thực dữ liệu bằng Quy tắc bảo mật của Cloud Firestore. Bạn có thể đảm bảo rằng các document liên quan luôn được cập nhật nguyên tử và luôn là một phần của giao dịch hoặc viết theo đợt. Sử dụng hàm quy tắc bảo mật  [getAfter()](https://firebase.google.com/docs/reference/rules/rules.firestore#.getAfter)để truy cập và xác thực trạng thái của document sau khi một bộ hoạt động hoàn thành nhưng trước khi Cloud Firestore thực hiện các hoạt động.

Ví dụ, hãy tưởng tượng rằng cơ sở dữ liệu cho ví dụ về cities cũng chứa collection  countries. Mỗi document   country sử dụng trường last\_updated để theo dõi lần cuối cùng bất kỳ thành phố nào liên quan đến quốc gia đó được cập nhật. Các quy tắc bảo mật sau đây yêu cầu rằng một bản cập nhật cho document city cũng phải tự động cập nhật trường last\_updated của  country liên quan:

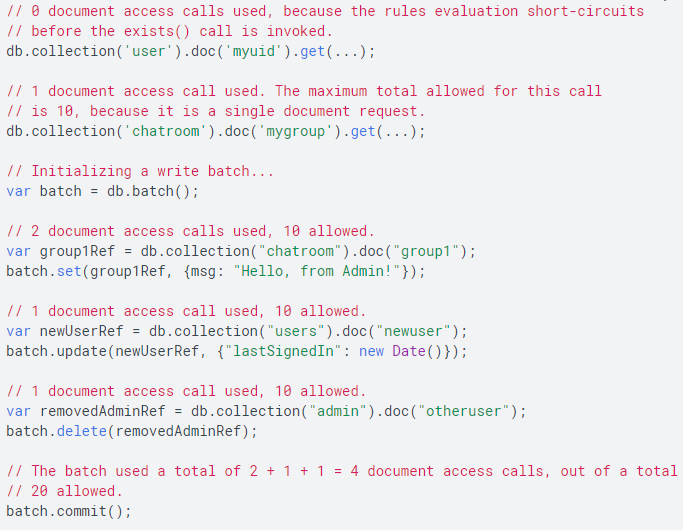


Giới hạn quy tắc bảo mật

Trong các quy tắc bảo mật cho các giao dịch hoặc ghi theo đợt, có giới hạn 20 lệnh truy cập document cho toàn bộ hoạt động nguyên tử ngoài giới hạn 10 cuộc gọi thông thường cho mỗi thao tác document đơn lẻ trong lô.

Ví dụ: xem xét các quy tắc sau cho ứng dụng trò chuyện:



Đoạn mã dưới đây minh họa số lượng cuộc gọi truy cập document được sử dụng cho một vài mẫu truy cập dữ liệu:

## Xóa dữ liệu khỏi Cloud Firestore

Các ví dụ sau minh họa cách xóa document, trường và collection.

### Xóa document

Để xóa document, sử dụng phương thức delete():



**Cảnh báo**: *Xóa một document không xóa các tập hợp con của nó!*

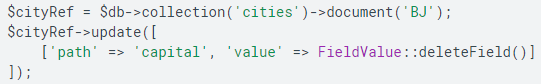
Khi bạn xóa một document, Cloud Firestore không tự động xóa các document trong các collection con của nó. Bạn vẫn có thể truy cập các document collection phụ bằng cách tham khảo. Ví dụ: bạn có thể truy cập document tại /mycoll/mydoc/mysubcoll/mysubdoc ngay cả khi bạn xóa document tổ tiên tại /mycoll/mydoc.

Document tổ tiên không tồn tại xuất hiện trong bảng điều khiển, nhưng chúng không xuất hiện trong kết quả truy vấn và ảnh chụp nhanh.

Nếu bạn muốn xóa một document và tất cả các document trong các collection của nó, bạn phải thực hiện thủ công. Để biết thêm thông tin, xem Xóa Collection.

### Xóa các trường

Để xóa các trường cụ thể khỏi document, hãy sử dụng phương thức FieldValue.delete() khi bạn cập nhật document:

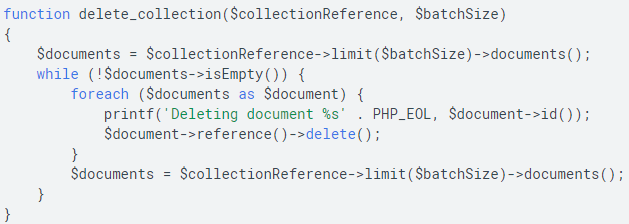


### Xóa collection

Để xóa toàn bộ collection hoặc collection con trong Cloud Firestore, hãy truy xuất tất cả các document trong collection hoặc collection con và xóa chúng. Nếu bạn có collection lớn hơn, bạn có thể muốn xóa các document theo lô nhỏ hơn để tránh lỗi hết bộ nhớ. Lặp lại quy trình cho đến khi bạn xóa toàn bộ collection hoặc collection con.

Xóa một collection đòi hỏi phải phối hợp một số lượng yêu cầu xóa riêng lẻ không giới hạn. Nếu bạn cần xóa toàn bộ collection, chỉ làm như vậy từ môi trường máy chủ đáng tin cậy. Mặc dù có thể xóa collection khỏi ứng dụng khách trên thiết bị di động / web, nhưng làm như vậy có ý nghĩa tiêu cực về bảo mật và hiệu suất.

Các đoạn dưới đây có phần đơn giản hóa và không xử lý lỗi, bảo mật, xóa các bộ lọc con hoặc tối đa hóa hiệu suất. Để tìm hiểu thêm về một cách tiếp cận được đề xuất để xóa các collection trong sản xuất, hãy xem Xóa Collection và Collection.



### Xóa dữ liệu với Firebase CLI

Bạn cũng có thể sử dụng Firebase CLI để xóa document và collection. Sử dụng lệnh sau để xóa dữ liệu:



## Quản lý Cloud Firestore với Bảng điều khiển Firebase

Bạn có thể thực hiện các hành động sau trên Cloud Firestore khi sử dụng bảng điều khiển Firebase:

* Xem, thêm, chỉnh sửa và xóa dữ liệu.
* Tạo và cập nhật Quy tắc bảo mật của Cloud Firestore.
* Quản lý chỉ mục.
* Giám sát việc sử dụng.

### Xem dữ liệu

Bạn có thể xem tất cả dữ liệu Cloud Firestore của mình trong bảng điều khiển Firebase. Từ tab Cloud Firestore Data, nhấp vào document hoặc collection để mở dữ liệu được lồng trong mục đó.

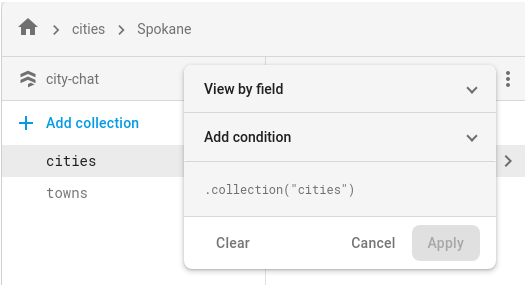
Mở một đường dẫn cụ thể

Để mở document hoặc collection tại một đường dẫn cụ thể, hãy sử dụng nút **Edit path** 



Lọc document trong collection

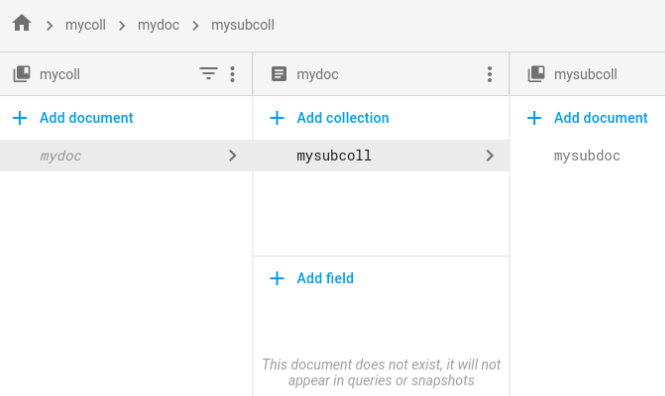
Để lọc các document được liệt kê trong collection, hãy sử dụng nút  **Filter list** :



Document tổ tiên không tồn tại

Một document có thể tồn tại ngay cả khi một hoặc nhiều tổ tiên của nó không tồn tại. Ví dụ: document tại /mycoll/mydoc/mysubcoll/mysubdoc có thể tồn tại ngay cả khi document tổ tiên /mycoll/mydoc không có. Trình xem dữ liệu của Cloud Firestore hiển thị document tổ tiên không tồn tại như sau:

* Trong danh sách document của collection, ID document của document tổ tiên không tồn tại được in nghiêng.
* Trong bảng thông tin của document tổ tiên không tồn tại, trình xem dữ liệu chỉ ra rằng document không tồn tại.



**Cảnh báo**: *Mặc dù các document tổ tiên không tồn tại xuất hiện trong bảng điều khiển, chúng không xuất hiện trong các truy vấn và ảnh chụp nhanh. Bạn phải tạo document để đưa nó vào kết quả truy vấn.*

### Quản lý dữ liệu

Trong Cloud Firestore, bạn lưu trữ dữ liệu trong document và sắp xếp document của bạn thành các collection. Trước khi bạn bắt đầu thêm dữ liệu, hãy tìm hiểu thêm về mô hình dữ liệu Cloud Firestore.

Bạn có thể thêm, chỉnh sửa và xóa document và collection khỏi bảng điều khiển Firebase. Để quản lý dữ liệu của bạn, hãy mở tab Dữ liệu trong phần **Database** :

Lưu ý: *Đọc, viết và xóa các thao tác được thực hiện trong bảng điều khiển theo hướng sử dụng Cloud Firestore của bạn.*

Thêm dữ liệu

1. Nhấp vào **Add collection**, sau đó nhập tên collection của bạn và nhấp vào **Next**.
2. Nhập ID document cụ thể hoặc nhấp vào **Auto ID**, sau đó thêm các trường cho dữ liệu trong document của bạn.
3. Nhấp vào để **Save.** Collection và document mới của bạn xuất hiện trong trình xem dữ liệu.
4. Để thêm nhiều document vào collection, bấm  **Add document**.

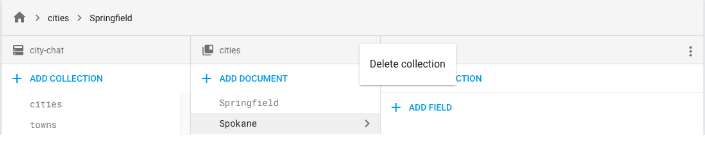
Chỉnh sửa dữ liệu

1. Bấm vào một collection để xem các document của nó, sau đó bấm vào một document để xem các trường và các collection của nó.
2. Bấm vào một trường để chỉnh sửa giá trị của nó. Để thêm các trường hoặc collection con vào document đã chọn, bấm **Add field** hoặc **Add collection**

Xóa dữ liệu

Để xóa một collection:

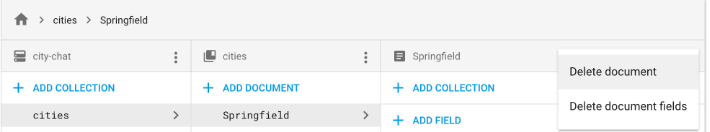
1. Chọn collection bạn muốn xóa.
2. Nhấp vào biểu tượng menu ở đầu cột document, sau đó nhấp vào **Delete collection**.



Để xóa một document hoặc tất cả các trường của nó:

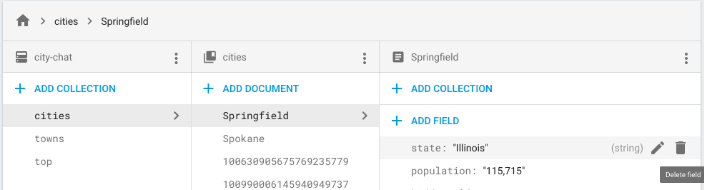
1. Chọn document bạn muốn xóa.
2. Nhấp vào biểu tượng menu ở đầu cột chi tiết document. Chọn **Delete document**  hoặc **Delete document fields**.

Xóa một document sẽ xóa tất cả các dữ liệu lồng nhau trong document đó, bao gồm mọi collection con. Tuy nhiên, xóa các trường của document không xóa các tập hợp con của nó.



Để xóa một trường cụ thể trong document:

1. Chọn document để xem các lĩnh vực của nó.
2. Nhấp vào biểu tượng xóa bên cạnh trường bạn muốn xóa.



### Quản lý quy tắc bảo mật đám mây

Để thêm, chỉnh sửa và xóa Quy tắc bảo mật của Cloud Firestore khỏi bảng điều khiển Firebase, hãy chuyển đến tab Quy tắc trong phần Cơ sở dữ liệu. Tìm hiểu thêm về thiết lập và tùy chỉnh các quy tắc.

### Quản lý chỉ mục

Để tạo các chỉ mục mới cho các truy vấn của bạn và quản lý các chỉ mục hiện có từ bảng điều khiển Firebase, hãy chuyển đến tab Chỉ mục trong phần Cơ sở dữ liệu. Tìm hiểu thêm về quản lý chỉ mục.

### Giám sát việc sử dụng

Để theo dõi việc sử dụng Cloud Firestore của bạn, hãy mở tab Sử dụng Cloud Firestore trong Bảng điều khiển Firebase. Sử dụng bảng điều khiển để đánh giá mức độ sử dụng của bạn trong các khoảng thời gian khác nhau.

## Xuất và nhập dữ liệu

Bạn có thể sử dụng dịch vụ xuất và nhập được quản lý của Cloud Firestore để khôi phục sau khi xóa dữ liệu do vô tình và để xuất dữ liệu để xử lý ngoại tuyến. Bạn có thể xuất tất cả các document hoặc chỉ các collection cụ thể. Tương tự như vậy, bạn có thể nhập tất cả dữ liệu từ xuất hoặc chỉ các collection cụ thể. Dữ liệu được xuất từ ​​một cơ sở dữ liệu Cloud Firestore có thể được nhập vào cơ sở dữ liệu Cloud Firestore khác. Bạn cũng có thể tải các bản xuất của Cloud Firestore vào BigQuery.

Trang này mô tả cách xuất và nhập document Cloud Firestore bằng dịch vụ xuất và nhập được quản lý và Cloud Storage. Dịch vụ xuất và nhập được quản lý bởi Cloud Firestore có sẵn thông qua công cụ dòng lệnh [gcloud](https://cloud.google.com/sdk/gcloud/) và API Cloud Firestore (REST, RPC).

### Trước khi bắt đầu

Trước khi bạn có thể sử dụng dịch vụ xuất và nhập được quản lý, bạn phải hoàn thành các tác vụ sau:

1. Cho phép thanh toán cho dự án Google Cloud Platform của bạn. Chỉ các dự án GCP có bật hóa đơn mới có thể sử dụng chức năng xuất và nhập.

Lưu ý: *Các dự án Firebase phải nằm trong kế hoạch Blaze để sử dụng dịch vụ xuất nhập khẩu được quản lý. Việc bật thanh toán cho Google Cloud Platform sẽ tự động nâng cấp dự án Firebase của bạn lên gói Blaze.*

1. Tạo nhóm lưu trữ đám mây cho dự án của bạn ở một vị trí gần vị trí cơ sở dữ liệu Cloud Firestore của bạn. Bạn không thể sử dụng nhóm Yêu cầu thanh toán cho các hoạt động xuất và nhập.
2. Đảm bảo tài khoản của bạn có các quyền cần thiết cho Cloud Firestore và Cloud Storage. **Nếu bạn là chủ dự án, tài khoản của bạn có các quyền cần thiết**. Mặt khác, các vai trò sau đây cấp các quyền cần thiết cho hoạt động xuất và nhập và để truy cập vào Cloud Storage:

* Vai trò của Cloud Firestore: Owner, Cloud Datastore Owner, Owner hoặc Storage Admin Cloud Datastore Import Export Admin

Lưu ý: *Các vai trò của Kho dữ liệu đám mây này cũng cấp quyền trong Cloud Firestore*.

* Vai trò lưu trữ đám mây: Owner hoặc Storage Admin

Thiết lập gcloud cho dự án của bạn

Thiết lập công cụ dòng lệnh gcloud và kết nối với dự án của bạn theo một trong các cách sau:

* Truy cập gcloud từ bảng điều khiển Google Cloud Platform bằng Cloud Shell.

Đảm bảo gcloud được cấu hình cho dự án chính xác:



* Cài đặt và khởi tạo Google Cloud SDK.

### Xuất dữ liệu

Một thao tác xuất sao chép các document trong cơ sở dữ liệu của bạn vào một tập hợp các tệp trong nhóm Lưu trữ đám mây. Lưu ý rằng xuất không phải là ảnh chụp nhanh cơ sở dữ liệu chính xác được thực hiện tại thời điểm bắt đầu xuất. Xuất khẩu có thể bao gồm các thay đổi được thực hiện trong khi hoạt động đang chạy.

Lưu ý*: Bạn phải xuất các collection cụ thể nếu bạn dự định:*

* *Chỉ nhập các collection cụ thể*
* *Tải dữ liệu Cloud Firestore vào BigQuery*

Xuất tất cả document

Sử dụng lệnh  [firestore export](https://cloud.google.com/sdk/gcloud/reference/firestore/export) để xuất tất cả các document trong cơ sở dữ liệu của bạn, thay thế [BUCKET\_NAME] bằng tên của nhóm Lưu trữ đám mây của bạn. Thêm cờ

--async để ngăn công cụ gcloud chờ hoạt động hoàn tất.



Bạn có thể tổ chức xuất khẩu của mình bằng cách thêm tiền tố tệp sau tên nhóm, ví dụ: [BUCKET\_NAME] /my-exports-folder/export-name. Nếu bạn không cung cấp tiền tố tệp, dịch vụ xuất được quản lý sẽ tạo một tiền tố dựa trên dấu thời gian hiện tại.

Khi bạn bắt đầu thao tác xuất, đóng thiết bị đầu cuối không hủy hoạt động, hãy xem hủy thao tác.

Xuất collection cụ thể

Để xuất các nhóm collection cụ thể, hãy sử dụng cờ  [--collection-ids](https://cloud.google.com/sdk/gcloud/reference/firestore/export#FLAGS). Hoạt động chỉ xuất các nhóm collection với ID collection đã cho. Nhóm collection bao gồm tất cả các collection và collection con (tại bất kỳ đường dẫn nào) với ID collection được chỉ định.

gcloud firestore export gs://[BUCKET\_NAME] --collection-ids=[COLLECTION\_ID\_1],[COLLECTION\_ID\_2]

### Nhập dữ liệu

Khi bạn đã xuất các tệp trong Cloud Storage, bạn có thể nhập document trong các tệp đó trở lại dự án của bạn hoặc sang dự án khác. Lưu ý các điểm sau về hoạt động nhập khẩu:

* Khi bạn nhập dữ liệu, các chỉ mục cần thiết được cập nhật bằng định nghĩa chỉ mục hiện tại của cơ sở dữ liệu của bạn. Xuất khẩu không chứa định nghĩa chỉ mục.
* Nhập khẩu không chỉ định ID document mới. Nhập khẩu sử dụng ID thu được tại thời điểm xuất khẩu. Vì một document đang được nhập, ID của nó được dành riêng để ngăn va chạm ID. Nếu một document có cùng ID đã tồn tại, quá trình nhập sẽ ghi đè lên document hiện có.
* Nếu một document trong cơ sở dữ liệu của bạn không bị ảnh hưởng bởi quá trình nhập, nó sẽ vẫn còn trong cơ sở dữ liệu của bạn sau khi nhập.

Nhập tất cả document từ xuất

Sử dụng lệnh  [firestore import](https://cloud.google.com/sdk/gcloud/reference/firestore/import) để nhập document từ thao tác xuất trước đó.



trong đó [BUCKET\_NAME] và [EXPORT\_PREFIX] trỏ đến vị trí tệp xuất của bạn. Ví dụ:



Nhập collection cụ thể

Để nhập các nhóm collection cụ thể từ một tập hợp các tệp xuất, hãy sử dụng

[--collection-ids](https://cloud.google.com/sdk/gcloud/reference/firestore/export#FLAGS). Hoạt động chỉ nhập các nhóm collection với ID collection đã cho. Nhóm collection bao gồm tất cả các collection và collection con (tại bất kỳ đường dẫn nào) với ID collection được chỉ định.

Chỉ xuất khẩu các nhóm collection cụ thể mới hỗ trợ nhập các nhóm collection cụ thể. Bạn không thể nhập các collection cụ thể từ xuất tất cả các document.

gcloud firestore import gs://[BUCKET\_NAME]/[EXPORT\_PREFIX]/ --collection-ids=[COLLECTION\_ID\_1],[COLLECTION\_ID\_2]

### Quản lý hoạt động xuất nhập dữ liệu

Sau khi bạn bắt đầu thao tác xuất hoặc nhập, Cloud Firestore gán cho thao tác một tên duy nhất. Bạn có thể sử dụng tên hoạt động để xóa, hủy hoặc kiểm tra trạng thái hoạt động.

Tên hoạt động được tiền tố với các dự án projects/[PROJECT\_ID]/databases/(default)/operations/, ví dụ:



Tuy nhiên, bạn có thể bỏ tiền tố khi chỉ định tên thao tác cho describe, cancel, và delete.

Liệt kê tất cả các hoạt động xuất nhập

Sử dụng lệnh [operations list](https://cloud.google.com/sdk/gcloud/reference/firestore/operations/list)  để xem tất cả các hoạt động xuất và nhập đã hoàn thành gần đây:

gcloud firestore operations list

Kiểm tra trạng thái hoạt động

Sử dụng lệnh  [operations describe](https://cloud.google.com/sdk/gcloud/reference/firestore/operations/describe) để hiển thị trạng thái của thao tác xuất hoặc nhập.

gcloud firestore operations describe [OPERATION\_NAME]

**Ước tính thời gian hoàn thành**

Một yêu cầu cho trạng thái của một hoạt động chạy dài trả về các số liệu workEstimated và workCompleted. Mỗi số liệu này được trả về cả số byte và số lượng thực thể:

* workEstimated hiển thị tổng số byte và document mà một thao tác sẽ xử lý.
* workCompleted hiển thị số byte và document được xử lý cho đến nay. Sau khi hoàn thành thao tác, giá trị hiển thị tổng số byte và document đã được xử lý thực sự, có thể lớn hơn giá trị của workEstimated.

Phân chia workCompleted bởi workEstimated cho một ước tính tiến độ sơ bộ. Ước tính này có thể không chính xác, bởi vì nó phụ thuộc vào việc thu thập số liệu thống kê bị trì hoãn.

Hủy bỏ một hoạt động

Sử dụng lệnh [operations cancel](https://cloud.google.com/sdk/gcloud/reference/firestore/operations/cancel) để dừng hoạt động đang diễn ra:

gcloud firestore operations cancel [OPERATION\_NAME]

Hủy bỏ một hoạt động đang chạy không hoàn tác các hoạt động. Một hoạt động xuất bị hủy sẽ để lại các document đã được xuất trong Cloud Storage và một hoạt động nhập bị hủy sẽ để lại các bản cập nhật đã được thực hiện cho cơ sở dữ liệu của bạn. Bạn không thể nhập một xuất khẩu hoàn thành một phần.

Xóa một thao tác

Sử dụng lệnh [operations delete](https://cloud.google.com/sdk/gcloud/reference/firestore/operations/delete)  để xóa một hoạt động khỏi đầu ra operations list. Lệnh này sẽ không xóa các tệp xuất khỏi Cloud Storage.

gcloud firestore operations delete [OPERATION\_NAME]

### Thanh toán và giá cả cho các hoạt động xuất nhập

Bạn được yêu cầu kích hoạt thanh toán cho dự án Google Cloud Platform trước khi bạn sử dụng dịch vụ xuất và nhập được quản lý. Các hoạt động xuất khẩu và nhập khẩu được tính cho việc đọc và ghi document theo tỷ lệ được liệt kê trong giá của Cloud Firestore.

Chi phí cho hoạt động xuất nhập khẩu không được tính vào giới hạn chi tiêu của bạn. Hoạt động xuất hoặc nhập sẽ không kích hoạt cảnh báo ngân sách Google Cloud Platform cho đến khi hoàn thành. Tương tự, đọc và ghi được thực hiện trong một hoạt động xuất hoặc nhập được áp dụng cho hạn ngạch hàng ngày của bạn sau khi hoàn thành thao tác.

### Xuất sang BigQuery

Bạn có thể tải dữ liệu từ bản xuất của Cloud Firestore vào BigQuery, nhưng chỉ khi bạn đã chỉ định bộ lọc collection-ids

## Di chuyển dữ liệu giữa các dự án

Trang này mô tả cách sử dụng các tính năng nhập và xuất được quản lý để di chuyển dữ liệu Cloud Firestore từ dự án này sang dự án khác. Điều này có thể hữu ích để thiết lập môi trường phát triển hoặc là một phần của việc di chuyển vĩnh viễn một ứng dụng sang dự án khác. Ví dụ trên trang này trình bày cách xuất dữ liệu từ dự án nguồn và sau đó nhập dữ liệu đó vào dự án đích. Di chuyển dữ liệu giữa các dự án bao gồm các bước sau:

1. Tạo nhóm lưu trữ đám mây để giữ dữ liệu từ dự án nguồn của bạn.
2. Xuất dữ liệu từ dự án nguồn của bạn vào thùng.
3. Cho phép dự án đích của bạn để đọc từ xô.
4. Nhập dữ liệu từ nhóm vào dự án đích của bạn.

### Trước khi bắt đầu

Trước khi bạn có thể sử dụng dịch vụ xuất và nhập được quản lý, bạn phải hoàn thành các tác vụ sau:

1. Cho phép thanh toán cho cả dự án nguồn và dự án đích của bạn. Chỉ các dự án GCP có bật hóa đơn mới có thể sử dụng chức năng xuất và nhập.

Lưu ý*: Các dự án Firebase phải nằm trong kế hoạch Blaze để sử dụng dịch vụ xuất nhập khẩu được quản lý. Việc bật thanh toán cho Google Cloud Platform sẽ tự động nâng cấp dự án Firebase của bạn lên gói Blaze.*

1. Đảm bảo tài khoản của bạn có các quyền Cloud IAM cần thiết trong dự án nguồn và dự án đích của bạn. **Nếu bạn là chủ dự án cho cả hai dự án, tài khoản của bạn có các quyền cần thiết**. Mặt khác, các vai trò Cloud IAM sau đây cấp các quyền cần thiết cho các hoạt động xuất và nhập của Cloud Firestore:

Owner, Cloud Datastore Owner, or Cloud Datastore Import Export Admin

Lưu ý: *Các vai trò này cấp quyền cho cả Cloud Datastore và Cloud Firestore.*

Chủ dự án có thể cấp một trong những vai trò này bằng cách thực hiện theo các bước trong quyền truy cập Grant.

1. Thiết lập công cụ dòng lệnh gcloud và kết nối với dự án của bạn theo một trong các cách sau:

* Truy cập gcloud từ Bảng điều khiển nền tảng đám mây của Google bằng Cloud Shell.

Đảm bảo gcloud được cấu hình cho dự án chính xác:



* Cài đặt và khởi tạo Google Cloud SDK.

### Xuất dữ liệu từ dự án nguồn

Xuất dữ liệu của bạn bằng cách tạo nhóm Lưu trữ đám mây cho các tệp xuất Cloud Firestore của bạn và bắt đầu thao tác xuất.

Tạo nhóm lưu trữ đám mây

Tạo nhóm lưu trữ đám mây ở cùng vị trí với cơ sở dữ liệu Cloud Firestore của bạn. Để xem vị trí cơ sở dữ liệu của bạn, xem cài đặt vị trí dự án của bạn. Bạn không thể sử dụng nhóm Yêu cầu thanh toán cho các hoạt động xuất và nhập.

**Nếu nhóm Lưu trữ đám mây của bạn không nằm trong dự án nguồn của bạn,** bạn phải cấp quyền truy cập tài khoản dịch vụ mặc định của dự án nguồn vào nhóm. Mỗi dự án GCP có một tài khoản dịch vụ mặc định được tạo tự động với tên *PROJECT\_ID*@appspot.gserviceaccount.com. Các hoạt động xuất của Cloud Firestore sử dụng tài khoản dịch vụ mặc định này để ủy quyền cho các hoạt động của nhóm Cloud Storage. Để cấp quyền truy cập tài khoản dịch vụ mặc định vào nhóm nguồn của bạn, hãy cấp cho nó vai trò [Storage Admin](https://cloud.google.com/storage/docs/access-control/iam-roles).

Bạn có thể cấp vai trò này với công cụ gsutil  có sẵn trong Cloud Shell: 

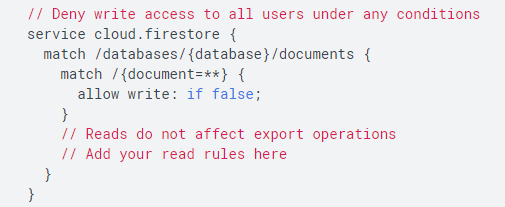
Bạn cũng có thể cấp vai trò này trong Bảng điều khiển Google Cloud Platform.

Vô hiệu hóa thao tác ghi (tùy chọn)

Nếu ứng dụng của bạn tiếp tục ghi vào cơ sở dữ liệu của bạn trong khi bạn thực hiện thao tác xuất, bạn có thể không nắm bắt được tất cả những gì ghi trong tệp xuất của bạn. Để xuất dữ liệu từ trạng thái nhất quán, hãy vô hiệu hóa ghi vào cơ sở dữ liệu của bạn bằng cách cập nhật các quy tắc bảo mật của bạn và tạm dừng mọi hoạt động của SDK quản trị.

1. Cập nhật các quy tắc bảo mật

Trong tab Cloud Firestore Rules của bảng điều khiển, hãy cập nhật các quy tắc bảo mật dự án nguồn của bạn để từ chối tất cả các ghi. Ví dụ:



1. Dừng viết từ SDK quản trị

Các quy tắc bảo mật không ngừng ghi đến từ các môi trường máy chủ đặc quyền được tạo bằng SDK quản trị Firebase hoặc Thư viện máy khách Google Cloud Server. Đảm bảo tạm dừng các hoạt động ghi từ máy chủ quản trị của bạn bằng cách tắt hoặc cập nhật máy chủ của bạn.

Bắt đầu một hoạt động xuất khẩu

Sử dụng lệnh gcloud firestore export để xuất dữ liệu từ dự án nguồn của bạn. Bạn có thể xuất tất cả dữ liệu của bạn hoặc chỉ các collection cụ thể. Thay thế [SOURCE\_BUCKET] bằng tên của nhóm Lưu trữ đám mây của bạn:

**Xuất tất cả dữ liệu**



**Xuất collection cụ thể**

gcloud firestore export gs://*[SOURCE\_BUCKET]* --collection-ids=*[COLLECTION\_ID\_1]*,*[COLLECTION\_ID\_2]* --async

Hãy lưu ý đầu raURIPrefix của hoạt động xuất khẩu của bạn vì bạn sẽ sử dụng điều này sau này. Theo mặc định, Cloud Firestore thêm một sửa chữa trước vào các tệp xuất của bạn dựa trên dấu thời gian:



Khi hoạt động xuất chạy, bạn có thể sử dụng lệnh danh sách hoạt động của Firestore để xem tiến trình hoạt động của mình:



### Nhập dữ liệu vào dự án đích

Tiếp theo, cấp quyền truy cập dự án đích vào các tệp dữ liệu Cloud Firestore của bạn và bắt đầu thao tác nhập.

Cấp quyền truy cập dự án đích vào các tệp dữ liệu của bạn

Trước khi bạn có thể bắt đầu thao tác nhập, bạn phải đảm bảo dự án đích của bạn có thể truy cập các tệp dữ liệu Cloud Firestore của bạn.

**Di chuyển tệp dữ liệu vào nhóm cục bộ**

Nếu vị trí nhóm nguồn của bạn khác với vị trí Cloud Firestore của dự án đích của bạn, bạn phải di chuyển các tệp dữ liệu của mình sang nhóm Lưu trữ đám mây ở cùng vị trí với dự án đích của bạn.

Di chuyển tệp dữ liệu của bạn sang nhóm lưu trữ đám mây khác bằng cách thực hiện theo các bước trong Di chuyển và đổi tên nhóm. Đối với tất cả các bước sau, hãy sử dụng nhóm mới này dưới dạng [SOURCE\_BucksET].

**Cấp quyền truy cập tài khoản dịch vụ dự án vào thùng nguồn của bạn**

Nếu nhóm nguồn của bạn không nằm trong dự án đích của bạn, thì bạn phải cấp quyền truy cập tài khoản dịch vụ mặc định của dự án đích vào nhóm nguồn của bạn. Tài khoản dịch vụ mặc định được đặt tên [DESTINATION\_PROJECT\_ID] @ appspot.gserviceaccount.com. Để cấp quyền truy cập tài khoản dịch vụ mặc định vào nhóm nguồn của bạn, hãy cấp cho nó vai trò Quản trị viên lưu trữ.

Bạn có thể cấp vai trò này với công cụ gsutil có sẵn trong Cloud Shell:

gsutil iam ch serviceAccount:*[DESTINATION\_PROJECT\_ID]*@appspot.gserviceaccount.com:admin \

gs://*[SOURCE\_BUCKET]*

Bạn cũng có thể cấp vai trò này trong bảng điều khiển GCP.

### Bắt đầu một hoạt động nhập khẩu

Trước khi bắt đầu thao tác nhập, hãy đảm bảo gcloud được cấu hình cho dự án chính xác:

gcloud config set project *[DESTINATION\_PROJECT\_ID]*

Sử dụng lệnh nhập gcloud Firestore để nhập dữ liệu trong nhóm nguồn của bạn vào dự án đích của bạn:

gcloud firestore import gs://*[SOURCE\_BUCKET]*/*[EXPORT\_PREFIX]* –async

Trong đó [XUẤT PREFIX] khớp với tiền tố trong đầu ra hoạt động xuất của bạn UriPrefix. Ví dụ:

gcloud firestore import gs://*[SOURCE\_BUCKET]*/2019-03-05T20:58:23\_56418 –async

Khi hoạt động xuất chạy, bạn có thể sử dụng lệnh danh sách hoạt động của Firestore để xem tiến trình hoạt động của mình:

gcloud firestore operations list

# Đọc dữ liệu:

## Lấy dữ liệu 1 lần

Có 2 cách để lấy dữ liệu từ Firestore, chúng đều có thể được dùng cho document, collection của các document hoặc kết quả truy vấn:

* Gọi method để lấy dữ liệu
* Dựng listener để thu thập các sự kiện thay đổi dữ liệu (không hỗ trợ trong PHP)

### Lấy 1 document

Để lấy nội dung của 1 document ta dùng phương thức snapshot() từ một reference đến một document trong collection cần lấy dữ liệu.

|  |
| --- |
| $docRef = $db->collection('cities')->document('SanFrancisco');  $snapshot = $docRef->snapshot();  **if** ($snapshot->exists())  {  **print\_r**($snapshot->data());  } **else**  {  **printf**('Document %s does not exist!', $snapshot->id());  } |

Nếu không có document nào được tham chiếu tại biến **docRef** thì **document** thu được sẽ rỗng, và phương thức **exists()** sẽ trả về giá trị **false**.

### Lấy nhiều document từ 1 collection

Ta có thể lấy nội dung của nhiều document trong một collection bằng cách thực hiện truy vấn các document bên trong collection đó.

|  |
| --- |
| $citiesRef = $db->collection('cities');  $query = $citiesRef->where('capital', '=', **true**);  $documents = $query->documents();  **foreach** ($documents **as** $document) {  **if** ($document->exists()) {  **print\_r**($document->data());  **printf**(**PHP\_EOL**);  } **else** {  **printf**('Document %s does not exist!', $snapshot->id());  }  } |

Để lấy tất cả document trong một collection ta không dùng phương thức where():

|  |
| --- |
| $citiesRef = $db->collection('cities');  $documents = $citiesRef->documents();  //... |

### Liệt kê các subcollection của 1 document

Phương thức collections() sẽ chứa các subcollection của một tham chiếu tới một document.

Đối với các thư viện mobile/web client ta không thể lấy một list của các collection mà chỉ có thể lấy tên của các collection cho việc quản lý trong các môi trường hệ thống được tin tưởng. Nếu cần chức năng này trong các thư viện mobile/web client, ta cần phải tái cấu trúc dữ liệu để tên của các subcollection có thể dự đoán được.

|  |
| --- |
| $cityRef = $db->collection('cities')->document('SF');  $collections = $cityRef->collections();  **foreach** ($collections **as** $collection) {  **printf**('Found subcollection with id: %s' . **PHP\_EOL**, $collection->id());  } |

## Truy vấn dữ liệu

### Truy vấn đơn giản

Phương thức where() nhận 3 tham số gồm 1 field, 1 toán tử so sánh và 1 giá trị để so sánh. Các toán tử bao gồm <, <=, ==, >, >= hay array-contains, trong đó toán tử array-contains dùng để truy vấn các phần tử mảng trong value.

|  |
| --- |
| $stateQuery = $citiesRef->where('state', '=', 'CA');  $populationQuery = $citiesRef->where('population', '>', 1000000);  $nameQuery = $citiesRef->where('name', '>=', 'San Francisco');  $containsQuery = $citiesRef->where('regions', 'array-contains', 'west\_coast'); |

### Truy vấn phức

Nhiều phương thức where() có thể được dùng nối tiếp nhau để để giả lập toán tử AND, hoặc nhiều lần truy vấn khác nhau lên một reference để thực hiện các truy vấn rõ ràng hơn. Trong một câu truy vấn phức chỉ được phép chứa tối đa 1 vế array-contains và chỉ được phép so sánh trên 1 field.

|  |
| --- |
| $chainedQuery = $citiesRef  ->where('state', '=', 'CA')  ->where('name', '=', 'San Francisco');  $chainedQuery = $citiesRef  ->where('state', '=', 'CA')  ->where('population', '<', 1000000); |

### Một số hạn chế truy vấn

* Không thể thực hiện so sánh trên nhiều field khác nhau như đã nêu trên.
* Toán tử OR, ta cần phải thực hiện truy vấn trên mỗi điều kiện OR khác nhau và gộp các kết quả thu được.
* Toán tử !=, ta cần phải thực hiện truy vấn với toán tử < và >, sau đó gộp các kết quả thu được.

### Phân loại và giới hạn dữ liệu

Theo mặc định kết quả truy vấn trả về sẽ được sắp xếp theo thứ tự tăng dần của document ID. Ta có thể phân loại dữ liệu với phương thức orderBy() và giới hạn dữ liệu với phương thức limit().

|  |
| --- |
| //Phân loại theo tên 3 kết quả đầu tiên  $query = $citiesRef->orderBy('name', 'DESC')->limit(3);  //Phân loại giảm dần và lấy 3 kết quả cuối  $query = $citiesRef->orderBy('state')->orderBy('population', 'DESC');  //Phân loại nhiểu lần để lấy kết quả theo tiểu bang và dân số giảm dần  $query = $citiesRef->orderBy('state')->orderBy('population', 'DESC'); |

Ta có thể kết hợp 3 phương thức where(), orderBy() và limit(), tuy nhiên, nếu thực hiện truy vấn so sánh trên một field thì chỉ có thể phân loại dữ liệu theo chính field đó.

|  |
| --- |
| $query = $citiesRef  ->where('population', '>', 2500000)  ->orderBy('population')  ->limit(2); |

## Phân trang dữ liệu với cursor

Với cursor ta có thể chia dữ liệu truy vấn ra thành các nhóm riêng dựa vào các tham số trong câu lệnh truy vấn. Cursor định nghĩa điểm khởi đầu và kết thúc của câu lệnh truy vấn, giúp ta:

* Trả về một tập hợp con của dữ liệu
* Phân trang truy vấn

Tuy nhiên để thiết lập một điểm truy vấn xác định, ta nên sử dụng phương thức where().

### Định nghĩa một cursor

Để lập một điểm khởi đầu cho lệnh truy vấn, ta có thể dùng phương thức startAt() hoặc startAfter(), trong đó startAt() bao gồm điểm khởi đầu và startAfter() không bao gồm điểm khởi đầu.

Để đặt điểm dừng ta có thể dùng phương thức endAt() hoặc endBefore().

|  |
| --- |
| $query = $citiesRef  ->orderBy('population')  ->startAt([1000000]);    $query = $citiesRef  ->orderBy('population')  ->endAt([2000000]); |

### Sử dụng document snapshot để định nghĩa cursor

Ta có thể dùng document snapshot để làm điểm truy vấn cho một cursor. Giá trị trong snapshop sẽ là giá trị trong cursor.

|  |
| --- |
| //Trả về các thành phố có dân số lớn hơn hoặc bằng San Francisco.  $citiesRef = $db->collection('cities');  $docRef = $citiesRef->document('SF');  $snapshot = $docRef->snapshot();  $query = $citiesRef  ->orderBy('population')  ->startAt($snapshot); |

### Phân trang truy vấn

Để phân trang truy vấn ta kết hợp cursor với phương thức limit().

|  |
| --- |
| $citiesRef = $db->collection('cities');  $firstQuery = $citiesRef->orderBy('population')->limit(3);  // Lấy document cuối cùng từ kết quả  $documents = $firstQuery->documents();  $lastPopulation = 0;  **foreach** ($documents **as** $document) {  $lastPopulation = $document['population'];  }  // Xây dựng truy vấn mới từ điểm trên  $nextQuery = $citiesRef->orderBy('population')->startAfter([$lastPopulation]);  $snapshot = $nextQuery->documents();  // Câu truy vấn này có thể trả về kết quả không mong muốn nếu các thành phố có số dân giống nhau |

### **Thiết lập nhiều điều kiện cho cursor**

Để tăng độ chi tiết cho điểm truy vấn của cursor, ta có thể thêm nhiều điều kiện bên trong vế khai báo cursor. Việc này sẽ giúp cho các câu truy vấn gồm các field thuộc điều kiện đầu trong vế cursor trả về các kết quả giống nhau. Việc đặt nhiều điều kiện giúp thể hiện rõ hơn điểm khởi đầu và kết thúc truy vấn, giảm tính mơ hồ của câu truy vấn.

|  |
| --- |
| /\* Dữ liệu mẫu:  Cities  Name | States  Springfield Massachusetts  Springfield Missouri  Springfield Wisconsin  \*/  // Will return all Springfields  $query1 = $db  ->collection('cities')  ->orderBy('name')  ->orderBy('state')  ->startAt(['Springfield']);  // Will return "Springfield, Missouri" and "Springfield, Wisconsin"  $query2 = $db  ->collection('cities')  ->orderBy('name')  ->orderBy('state')  ->startAt(['Springfield', 'Missouri']); |

## Truy cập dữ liệu khi ngoại tuyến

Chỉ hỗ trợ trong các ứng dụng Android, iOS và web.

## Quản lý chỉ mục (index)

Cloud Firestore đảm bảo chất lượng truy vấn bằng việc yêu cầu chỉ mục cho mỗi truy vấn. Các chỉ mục cho các truy vấn đơn giản thường được tạo tự động. Trong quá trình xây dựng ứng dụng, các lỗi phát sinh sẽ giúp người phát triển thêm các chỉ mục cần thiết cho ứng dụng. Các chỉ mục có thể được quản lý thông qua Firebase console hay Firebase CLI.

### **Thời gian xây dựng chỉ mục**

Tạo chỉ mục sẽ tốn vài phút, tùy theo lượng dữ liệu cần được cập nhật. Càng nhiều document tương xứng với các field chỉ mục sẽ tốn càng nhiều thời gian.

### **Lỗi xây dựng chỉ mục**

Trong quá trình dựng chỉ mục hay xảy ra lỗi, phần lớn là do index limit vì operation được thi hành đã có thể đạt tới số lượng chỉ mục tối đa cho mỗi document.

# Bảo mật và xác thực dữ liệu:

## Các Quy tắc bảo mật của Cloud firestore

- Với Quy tắc bảo mật của Cloud firestore (Cloud firestore Security rules - CFSR), bạn có thể tập trung xây dựng về mảng trải nghiệm người dùng mà không cần phải bận tâm đến quản lý cơ sở hạ tầng hay viết code ủy quyền hay xác thực server-side

- Những quy tắc bảo mật này cung cấp khả năng kiểm soát truy cập và định dạng dữ liệu theo một cách đơn giản nhưng chi tiết. Để tạo nên một cơ sở người dùng và cơ sở vai trò truy cập hệ thống thứ sẽ giúp cho dữ liệu người dùng của bạn luôn an toàn, bạn cần phải sử dụng Firebase Authentication với CFSR

### Quy tắc bảo mật version 2 (Security rules version 2)

* Kể từ tháng 5 2019, phiên bản thứ 2 của CFSR hiện tại đã có thể sử dụng. Theo GG thì chúng ta cần phải sử dụng phiên bản này nếu mục đích chúng ta dùng firestore để thu thập các nhóm truy vấn. Để sử dụng phiên bản này, chúng ta cần đặt rules\_version = '2'; ở dòng đầu tiên của Quy tắc bảo mật của bạn



### Viết Rules:

* Tất cả CFSR Gồm các câu lệnh 'match', nó dùng để định nghĩa document trong database của bạn và 'allow', nó dùng để điều khiển truy cập vào trong document:

Ex:



* Mỗi yêu cầu gửi đến database từ một Cloud Firestore mobile/web client thì phải thông qua security rules của bạn trước khi nó tiến hành đọc hay viết bất kì dữ liệu nào. Nếu việc truy cập bị từ chối bởi bất kì phần nào của document, thì toàn bộ yêu cầu sẽ bị hủy
* Dưới đây là một vài các thiết lập Rules cơ bản.

Yêu cầu đăng nhập:



Từ chối toàn bộ:



Cho phép toàn bộ:



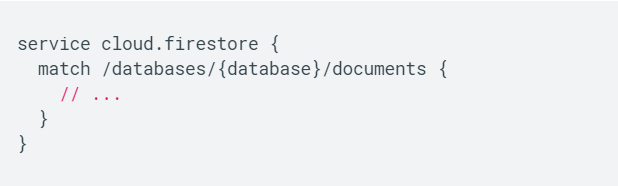
* {document=\*\*} được sử dụng ở ví dụ trên phù hợp với bất kỳ document nào ở trong database.

## Cấu trúc của Cloud Firestore Security Rules

Cloud Firestore Security Rules (CFSR) cho phép bạn điều khiển truy cập vào document và thu thập trên cơ sở dữ liệu của mình.

### Khai báo service và database

* CFSR luôn luôn bắt đầu theo đoạn khai báo như sau:



'service cloud.firestore' là đoạn khai báo cho việc bắt đầu cung cấp các quy tắc cho Cloud Firestore, ngăn ngừa sự xung đột giữa CFSR và các quy tắc khác cho những cho các sản phẩm khác giống như Cloud Storage.

'match /databases/{databases}/documents' là đoạn khai báo tiếp theo nhầm chỉ ra rằng các quy tắc phải phù hợp với bất kỳ database nào trong projects.

### Quy tắc Read/write cơ bản

* Các quy tắc cơ bản chứa một 'match' nhầm chỉ đến đường dẫn của 1 document và kèm theo phương thức 'allow' nhằm chi tiết những dữ liệu được nêu lên là cho phép đọc:



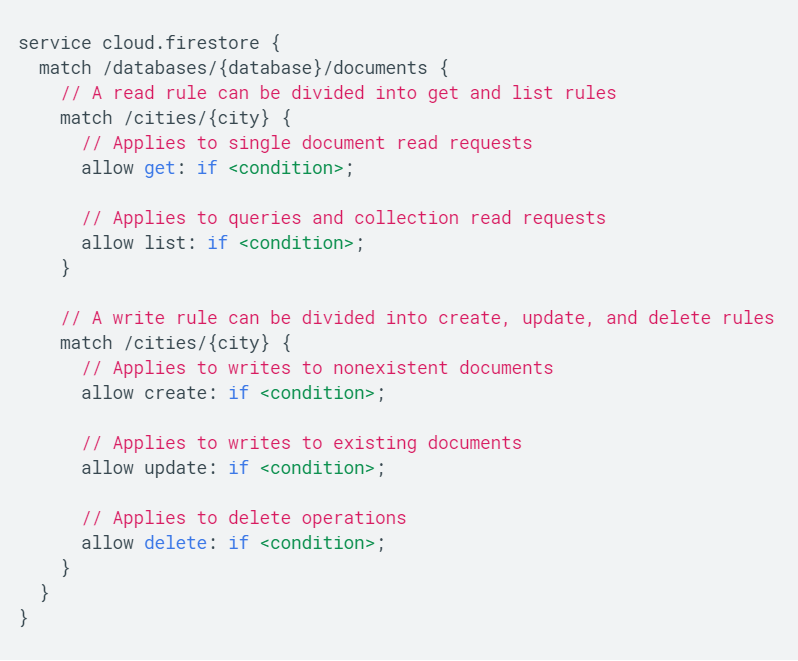
Tất cả các 'match' nên chỉ đến một document, không phải một bộ document. Một 'match' có thể chỉ đến một document cụ thể, như 'match /cities/SF' hoặc sử dụng những ký tự đại diện để chỉ đến bất cứ document nào trên đường dẫn cụ thể, ví dụ: 'match /cities/{city}'

Ở ví dụ trên, 'match' sửa dụng ký tự đại diện '{city}' để chỉ đến bất kỳ document nào nằm trong bộ 'cities', như là '/cities/SF' hay '/cities/NYC'. Và khi 'allow' nằm trong 'match' được thực thi, nó sẽ giải quyết toàn bộ các document nằm trong đoạn giá trị '{city}', giống như là 'SF' hay 'NYC'.

### Toán tử chia nhỏ (Granular operations)

* Trong một số tình huống, việc chia nhỏ 'read' và 'write' thành nhiều toán hạng con sẽ rất hữu dụng. Lấy ví dụ, phần mềm của bạn có thể sẽ muốn thực nhiều điều kiện khác nhau trong việc tạo document hơn là việc xóa nó. Hay là bạn có thể chỉ muốn cho phép đọc các document đơn và nhưng từ chối các truy vấn lớn. Một quy tắc về 'read' có thể được chia nhỏ thành 'get' và 'list', trong khi một 'write' có thể chia thành 'create', 'update', và 'delete’.

Ví dụ:



### Dữ liệu Phân cấp (hierarchical data)

* Dữ liệu trong Cloud Firestore được tổ chức bên trong các bộ của document, và mỗi document có thể mở rộng ra hệ thống phân cấp thông qua các bộ con. Đây là một thứ rất quan trọng để hiểu được như thế nào các Quy tắc Bảo mật tương tác được với dữ liệu phân cấp.
* Xem xét ngữ cảnh sau, nơi mỗi document nằm trong 1 bộ 'cities' chứa đựng 1 'landmarks' bộ con. Quy tắc bảo mật chỉ áp dụng với đường dẫn phù hợp, vì thế các điều khiển truy vấn đưa ra trên bộ 'cities' thì không được chỉ xuống cho bộ con 'landmarks'. Chính vì thế, cần phải viết một quy tắc rõ ràng để có thể điều khiển truy vấn vào bộ con.



* Khi lồng 'match' bên trong một 'match' khác, thì những thứ liên quan ở bên trong đều tương ứng với bên ngoài, do đó, một ví dụ như sau được xem là tương đương nhau:



#### Kí tự đại diện Đệ quy (Recursive wildcards)

* Nếu bạn muốn truy vấn đến một hệ thống phân cấp sâu tùy ý, cần phải sử dụng cú pháp "kí tự đại diện đệ quy", '{name=\*\*}.

Cho ví dụ:



* Khi sử dụng cú pháp trên, giá trị của kí tự đại diện sẽ chứa đựng toàn bộ đường dẫn phù hợp, ngay cả khi document được đặt ở vị trí sâu bên trong các bộ con lồng vào nhau. Lấy ví dụ, các quy tắc được liệt kê trên sẽ khớp với vị trí của một tập tin nằm ở '/cities/SF/landmarks/coit\_tower', và giá trí của 'document' lúc này sẽ là 'SF/landmarks/coit\_tower'.
* Chú ý rằng, hành vi, cách thức hoạt động của 'ký tự đại diện đệ quy' này phụ thuộc nhiều đến phiên bản của CFSR

### Chồng lấp match

* Có thể có nhiều 'match' đến một document. Trong trường hợp đó, khi mà một dòng 'allow' được một đường dẫn yêu cầu, việc truy cập đến sẽ được cho phép nếu trong trường hợp nó là 'true':



* Ở ví dụ này, tất cả việc đọc, viết đến bộ 'cities' sẽ được cho phép bởi vì theo quy tắc thứ 2 nó là true, mặc cho quy tắc ban đầu là false.

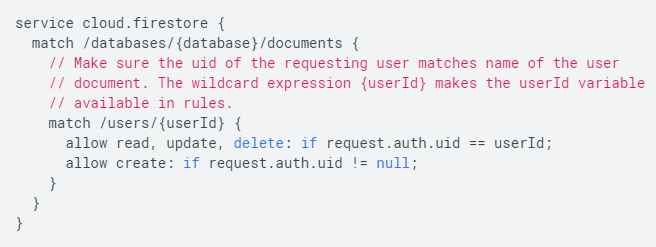
## Viết điều kiện cho Quy tắc bảo mật Cloud Firestore

### Tính xác thực

Một trong những mẫu quy tắc bảo mật phổ biến nhất là kiểm soát truy cập dựa trên trạng thái xác thực của người dùng. Lấy ví dụ, chương trình của bạn chỉ muốn để cho người dùng đã đăng nhập mới có thể viết dữ liệu:



Một mô hình phổ biến khác là đảm bảo người dùng chỉ có thể đọc và ghi dữ liệu của riêng họ:



Nếu ứng dụng của bạn sử dụng Xác thực Firebase, request.auth sẽ chứa thông tin xác thực cho khách hàng yêu cầu dữ liệu.

### Xác thực dữ liệu (Data validation)

Một số ứng dụng lưu trữ thông tin kiểm soát truy cập dưới dạng các trường trên document trong cơ sở dữ liệu. CFSR có thể tự động cho phép hoặc từ chối truy vấn dựa trên dữ liệu từ document:



Biến resource tham chiếu đến document được yêu cầu, và resource.data là một map của tất cả các trường và giá trị được lưu trữ trong document.

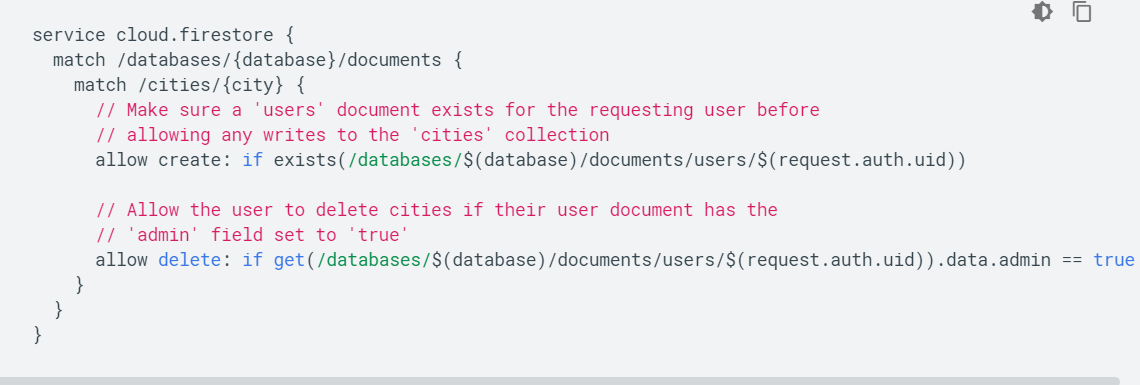
Trong khi đang ghi dữ liệu, bạn có thể muốn so sánh dữ liệu chuẩn bị đưa vào với dữ liệu đã tồn tại. Trong trường hợp đó, nếu đoạn dữ liệu đang chờ xử lý ấy được cho phép bởi Quy tắc đã thiết lập của bạn, thì request.resource chứa đựng trạng thái của document trong tương lai. Cho toán tử update thứ chỉ chỉ sửa đổi một tập hợp con của các trường trong document, biến request.resource sẽ chứa trạng thái document đang chờ xử lý sau thao tác. Bạn có thể kiểm tra các giá trị trường trong request.resource để ngăn cập nhật dữ liệu không mong muốn hoặc không nhất quán:



### Truy cập các document khác

Sử dụng các hàm get() và exists(), các quy tắc bảo mật của bạn có thể đánh giá các yêu cầu đến đối với các document khác trong cơ sở dữ liệu. Cả hai hàm get() và exists() đều cần các đường dẫn đến document được chỉ định rõ ràng. Khi sử dụng các biến để xây dựng đường dẫn cho get() và exists(), bạn cần chỉ ra rõ ràng các biến bằng cách sử dụng cú pháp $(biến).

Trong ví dụ dưới đây, biến database được bắt bởi câu lệnh match /databases/{database}/documents và được sử dụng để tạo đường dẫn:



Để ghi, bạn có thể sử dụng hàm getAfter() để truy cập trạng thái của document sau khi transaction hoặc một đợt ghi hoàn thành nhưng phải thực hiện trước một transaction hoặc một batch commits. Giống như get(), hàm getAfter() cần có một đường dẫn document được chỉ định đầy đủ. Bạn có thể sử dụng getAfter() để xác định các tập hợp ghi phải diễn ra cùng nhau dưới dạng một transaction hoặc một batch.

#### Giới hạn truy xuất

Có giới hạn đối với các lời gọi truy xuất vào document theo đánh giá của bộ quy tắc:

* 10 cho các yêu cầu document đơn và yêu cầu truy vấn.
* 20 cho nhiều document để đọc, giao dịch và viết theo đợt. Giới hạn 10 trước đó cũng áp dụng cho từng thao tác.

Vượt quá giới hạn dẫn đến lỗi từ chối cấp phép. Một số lời gọi truy xuất document có thể được lưu trong bộ nhớ cache và các lời gọi được lưu trong bộ nhớ cache không được tính vào các giới hạn.

#### Gọi truy vấn và chi phí

Sử dụng các hàm này sẽ thực thi thao tác đọc trong cơ sở dữ liệu của bạn, điều đó có nghĩa là bạn sẽ phải trẻ phí để đọc document ngay cả khi quy tắc của bạn từ chối yêu cầu.

### Chức năng tùy chỉnh (Custom functions)

Khi các quy tắc bảo mật của bạn trở nên phức tạp hơn, bạn có thể muốn bọc các bộ điều kiện trong các chức năng mà bạn có thể sử dụng lại trên quy tắc của mình. Quy tắc bảo mật hỗ trợ các chức năng tùy chỉnh. Cú pháp cho các hàm tùy chỉnh hơi giống với JavaScript, nhưng các hàm quy tắc bảo mật được viết bằng ngôn ngữ dành riêng cho tên miền có một số hạn chế quan trọng:

* Các hàm chỉ có thể chứa một câu lệnh return. Chúng không thể chứa bất kỳ logic bổ sung. Ví dụ, chúng không thể tạo các biến trung gian, thực thi các vòng lặp hoặc gọi những thứ ngoài phạm vi.
* Các hàm có thể tự động truy cập các hàm và biến từ phạm vi mà chúng được xác định. Ví dụ, một hàm được xác định trong phạm vi dịch vụ cloud.firestore có quyền truy cập vào biến tài nguyên và các hàm dựng sẵn như get() và exists().
* Các hàm có thể gọi các hàm khác nhưng có thể không tái diễn. Tổng độ sâu stack của lời gọi được giới hạn là 10.
* Một hàm được định nghĩa với từ khóa hàm và không có hoặc nhiều đối số.



Sử dụng các chức năng trong các quy tắc bảo mật của bạn làm cho chúng dễ bảo trì hơn cũng như làm tăng độ phức tạo Quy tắc bảo mật của bạn.

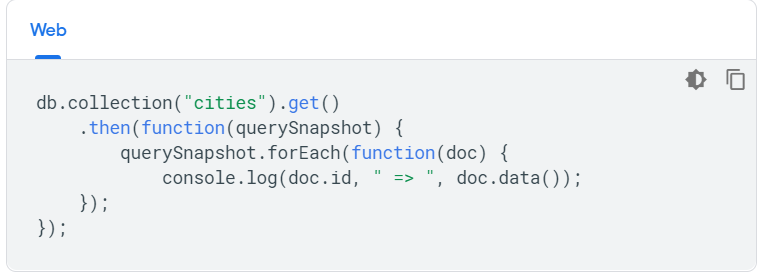
### Quy tắc không phải là bộ lọc (Rules are not filters)

Khi bạn bảo mật dữ liệu của mình và bắt đầu viết truy vấn, hãy nhớ rằng các quy tắc bảo mật không phải là bộ lọc. Bạn không thể viết truy vấn cho tất cả các document trong bộ document và mong muốn Cloud Firestore chỉ trả lại các document mà khách hàng hiện tại có quyền truy cập.

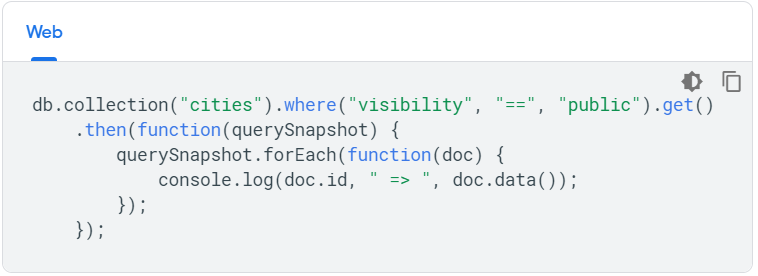
Lấy ví dụ sau:



-> Bị từ chối: Quy tắc này từ chối truy vấn sau vì tập kết quả có thể bao gồm các document không hiển thị công khai:



-> Được phép: Quy tắc này cho phép truy vấn sau vì mệnh đề where("visibility", "==", "public") đảm bảo rằng tập kết quả thỏa mãn điều kiện của quy tắc:



Các quy tắc bảo mật của Cloud Firestore xem xét từng truy vấn so với kết quả tiềm năng của nó và sẽ thất bại nếu nó có thể trả về một document mà khách hàng không có quyền đọc. Các truy vấn phải tuân theo các ràng buộc được đặt bởi các quy tắc bảo mật của bạn.

## Chỉnh sửa các quy tắc không an toàn (Fix insecure rules)

Sử dụng hướng dẫn này để hiểu các lỗ hổng phổ biến trong cấu hình Quy tắc bảo mật của Cloud Firestore, xem xét và bảo mật tốt hơn các quy tắc của riêng bạn và kiểm tra các thay đổi của bạn trước khi triển khai chúng.

Nếu bạn nhận được cảnh báo rằng cơ sở dữ liệu Cloud Firestore của bạn không được bảo mật đúng cách, bạn có thể giải quyết các lỗ hổng bằng cách sửa đổi và kiểm tra Quy tắc bảo mật Cloud Firestore của bạn.

### Hiểu quy tắc bảo mật Cloud Firestore của bạn

Quy tắc bảo mật của Cloud Firestore bảo vệ dữ liệu của bạn khỏi những người dùng nguy hiểm. Các quy tắc mặc định cho mọi phiên bản Cloud Firestore được tạo trong bảng điều khiển Firebase là từ chối quyền truy cập từ tất cả người dùng. Để phát triển ứng dụng của bạn và truy cập cơ sở dữ liệu của bạn, bạn sẽ cần sửa đổi các quy tắc đó và có thể xem xét cấp quyền truy cập chung cho tất cả người dùng trong môi trường phát triển. Tuy nhiên, trước khi triển khai ứng dụng của bạn vào môi trường sản xuất, hãy dành thời gian để định cấu hình đúng quy tắc của bạn và bảo mật dữ liệu của bạn.

### Các tình huống phổ biến với các quy tắc không an toàn

Quy tắc bảo mật của Cloud Firestore mà bạn có thể đã thiết lập theo mặc định hoặc khi bạn bắt đầu phát triển ứng dụng của mình với Cloud Firestore nên được xem xét và cập nhật trước khi bạn triển khai ứng dụng của mình. Đảm bảo bạn bảo mật đúng dữ liệu của người dùng bằng cách tránh những cạm bẫy phổ biến sau đây.

#### Mở truy cập

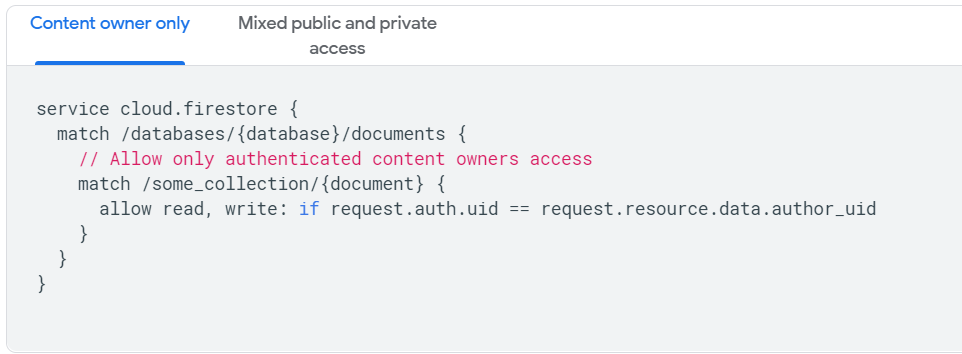
Khi bạn thiết lập Cloud Firestore, bạn có thể đã đặt quy tắc của mình để cho phép truy cập mở trong quá trình phát triển. Bạn có thể nghĩ bạn là người duy nhất sử dụng ứng dụng của mình, nhưng nếu bạn đã triển khai nó, nó sẽ trở nên khả dụng trên internet. Nếu bạn không xác thực người dùng và định cấu hình quy tắc bảo mật, thì bất kỳ ai đoán ID dự án của bạn đều có thể đánh cắp, sửa đổi hoặc xóa dữ liệu.

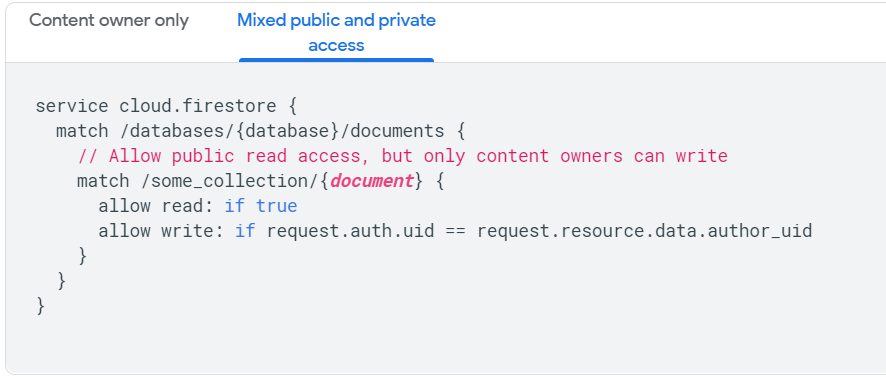
->**Không kiến nghị:** truy cập cho phép đọc và ghi dữ liệu cho toàn bộ users



->**Hướng giải quyết:** Thiết lập quy tắc để giảm thiểu việc truy cập để đọc và ghi

Xây dựng các quy tắc có ý nghĩa cho hệ thống phân cấp dữ liệu của bạn. Một trong những giải pháp phổ biến cho sự không an toàn này là bảo mật dựa trên người dùng với Tính xác thực của Firebase.





#### Truy cập cho bất kỳ người dùng đã xác thực

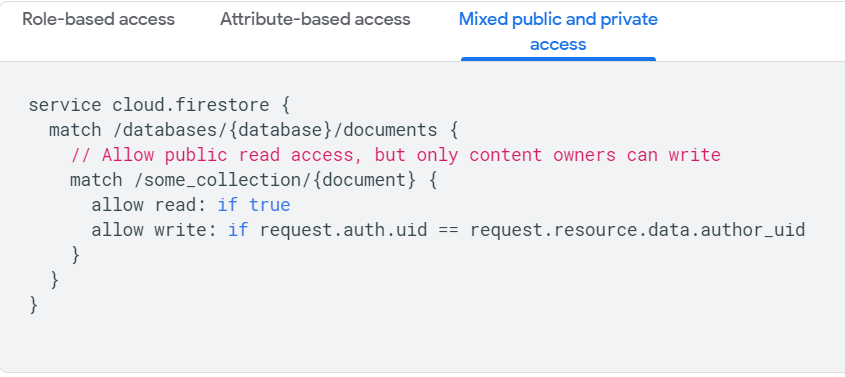
Đôi khi, Quy tắc bảo mật của Cloud Firestore kiểm tra xem người dùng đã đăng nhập chưa, nhưng không hạn chế thêm quyền truy cập dựa trên xác thực đó. Nếu một trong các quy tắc của bạn bao gồm auth! = Null, hãy xác nhận rằng bạn muốn bất kỳ người dùng đã đăng nhập nào có quyền truy cập vào dữ liệu.

->**Không kiến nghị:** Việc bất kỳ người dùng đã đăng nhập nào cũng có thể đọc, ghi dữ liệu vào trong database của bạn



->**Hướng giải quyết:** thu hẹp lại quyền truy cập bằng việc sử dụng một số các điều kiện bảo mật

Khi bạn đang kiểm tra xác thực, bạn cũng có thể muốn sử dụng một trong các thuộc tính xác thực để tiếp tục hạn chế quyền truy cập đối với người dùng cụ thể cho các bộ dữ liệu cụ thể.



#### Truy cập đóng

Trong khi bạn đang phát triển ứng dụng của mình, một cách tiếp cận phổ biến khác là giữ dữ liệu của bạn bị khóa. Thông thường, điều này có nghĩa là bạn đã đóng quyền truy cập đọc và viết cho tất cả người dùng, như sau:



SDK quản trị Firebase và các chức năng đám mây vẫn có thể truy cập cơ sở dữ liệu của bạn. Sử dụng các quy tắc này khi bạn có ý định sử dụng Cloud Firestore làm phụ trợ chỉ dành cho máy chủ kết hợp với SDK quản trị Firebase. Mặc dù an toàn, bạn nên kiểm tra xem ứng dụng của bạn trên máy khách có thể truy xuất dữ liệu đúng cách hay không.

### Kiểm tra các quy tắc bảo mật Cloud Firestore của bạn

Để kiểm tra hành vi của ứng dụng và xác minh cấu hình Quy tắc bảo mật Cloud Firestore của bạn, hãy sử dụng trình giả lập Cloud Firestore.

## Kiểm tra các quy tắc bảo mật Cloud Firestore của bạn

Khi bạn đang xây dựng ứng dụng của mình, bạn có thể muốn khóa quyền truy cập vào cơ sở dữ liệu Cloud Firestore của mình. Tuy nhiên, trước khi bạn khởi chạy, bạn sẽ cần nhiều quy tắc bảo mật Cloud Firestore hơn. Với trình giả lập Cloud Firestore, bạn có thể viết các bài kiểm tra đơn vị kiểm tra hành vi của Quy tắc bảo mật Cloud Firestore của bạn.

### Hiểu về quy tắc bảo mật Cloud Firestore

Thực hiện Quy tắc bảo mật Firebase và Cloud Firestore để xác thực không cần máy chủ, ủy quyền và xác thực dữ liệu khi bạn sử dụng thư viện máy khách và thiết bị di động.

Quy tắc bảo mật của Cloud Firestore bao gồm hai phần:

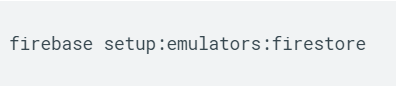
* Một tuyên bố phù hợp xác định các document trong cơ sở dữ liệu của bạn.
* Một biểu thức cho phép kiểm soát truy cập vào các document đó.

Xác thực Firebase xác minh thông tin đăng nhập của người dùng và cung cấp nền tảng cho các hệ thống truy cập dựa trên người dùng và dựa trên vai trò.

Mọi yêu cầu cơ sở dữ liệu từ thư viện ứng dụng khách trên thiết bị di động / web của Cloud Firestore được đánh giá theo các quy tắc bảo mật của bạn trước khi đọc hoặc ghi bất kỳ dữ liệu nào. Nếu các quy tắc từ chối truy cập vào bất kỳ đường dẫn document được chỉ định nào, toàn bộ yêu cầu sẽ thất bại.

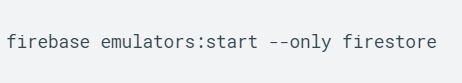
### Install trình giả lập

Để cài đặt trình giả lập Cloud Firestore, hãy sử dụng Firebase CLI và chạy lệnh bên dưới:



### Chạy trình giả lập

Để bắt đầu sử dụng trình giả lập, bạn cần sử dụng theo cú pháp sau. Trình giả lập sẽ chạy cho đến khi bạn hủy tiến trình:



Trong bất cứ trường hợp nào bạn muốn bắt đầu trình giả lập, chạy một bộ kiểm tra, và sau đó là kết thúc trình giả lập sau khi đoạn kiểm tra được thực hiện. Bạn có thể làm điều đó dễ dàng thông qua việc sử dụng cú pháp *emulators:exec*:



Khi bắt đầu, trình giả lập sẽ cố gắng chạy trên cổng cơ bản (port 8080). Bạn có thể thay đổi cổng cơ bản đó thông qua việc chỉnh sửa “emulators” thứ năm trong tệp firebase.json:



### Trước khi chạy trình giả lập

Trước khi bạn bắt đầu sử dụng trình giả lập, luôn nhớ các điều sau:

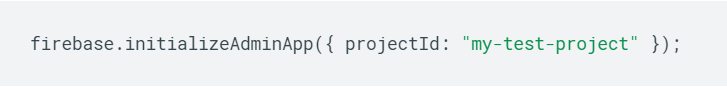
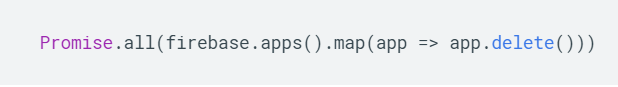
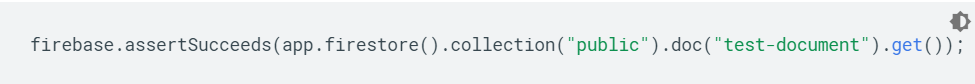
* Trình giả lập ban đầu sẽ tải các quy tắc được chỉ định trong trường Firestore.rules của tệp firebase.json của bạn. Nó hy vọng tên của một tệp cục bộ chứa Quy tắc bảo mật Cloud Firestore của bạn và áp dụng các quy tắc đó cho tất cả các dự án. Nếu bạn không cung cấp đường dẫn tệp cục bộ hoặc sử dụng phương thức loadFirestoreRules như được mô tả dưới đây, trình giả lập coi tất cả các dự án là có các quy tắc mở.
* Trong khi nhiều SDK làm việc với các giả lập, chỉ có @firebase/testing module của Node.js hỗ trợ giả mạo auth trong Quy định an toàn, làm các kiểm thử đơn vị dễ dàng hơn nhiều. Ngoài ra, mô-đun hỗ trợ một số tính năng dành riêng cho trình giả lập như xóa tất cả dữ liệu, như được liệt kê bên dưới.
* Các giả lập cũng sẽ chấp nhận sản xuất thẻ firebase Auth cung cấp thông qua Client SDK và đánh giá các quy định cho phù hợp, nó cho phép kết nối ứng dụng của bạn trực tiếp đến giả lập trong các kiểm tra tích hợp lẫn thủ công.

### Chạy thử nghiệm cục bộ

Sử dụng @firebase/testing module để tương tác trình giả lập chạy cục bộ. Nếu bạn nhận được timeout hoặc lỗi ECONNREFUSED, hãy kiểm tra thật kỹ xem trình giả lập có đang chạy không.

Chúng tôi cực kỳ khuyến khích việc sử dụng phiên bản gần đây của Node.js để bạn có thể sử dụng async/await. Hầu như tất cả các hành vi bạn có thể muốn kiểm tra đều liên quan đến các hàm không đồng bộ và mô-đun thử nghiệm được thiết kế để hoạt động với mã dựa trên Promise.

Module này đưa nêu ra với các phương thức sau:

* **initializeTestApp({ projectId: <name>, auth: <auth> }) => FirebaseApp**
  + Phương thức này sẽ trả về một app Firebase được khởi tạo tương ứng với ID project và biến auth được chỉ định trong thiết lập. Sử dụng thứ này để tạo ra một app được xác thực như một người dùng cụ thể để sử dụng trong các kiểm tra.
  + 
* **initializeAdminApp({ projectId: <name> }) => FirebaseApp**
  + Phương pháp này trả về một ứng dụng Firebase quản trị khởi tạo. Ứng dụng này bỏ qua các quy tắc bảo mật khi thực hiện đọc và ghi. Sử dụng điều này để tạo một ứng dụng được xác thực là quản trị viên để đặt trạng thái cho các kiểm thử.
  + 
* **apps() => [FirebaseApp]**
  + Phương pháp này trả về tất cả các ứng dụng quản trị và kiểm tra khởi tạo hiện tại. Sử dụng công cụ này để dọn dẹp các ứng dụng giữa hoặc sau khi kiểm tra.
  + 
* **loadFirestoreRules({ projectId: <name>, rules: <rules> }) => Promise**
  + Phương pháp này gửi các quy tắc đến một cơ sở dữ liệu đang chạy cục bộ. Nó nhận một đối tượng thứ được chỉ định trong quy tắc như một chuỗi. Sử dụng phương pháp này để đặt quy tắc cơ sở dữ liệu của bạn.
  + 
* **assertFails(pr: Promise) => Promise**
  + Phương thức này sẽ trả về một promise thứ bị từ chối nếu đầu vào thành công hoặc thành công nếu đầu vào bị từ chối. Sử dụng thứ này để khẳng định nếu một database đọc và ghi bị thất bại.
  + 
* **assertSucceeds(pr: Promise) => Promise**
  + Phương thức này sẽ trả về một promise thứ sẽ thành công nếu đầu vào là thành công hoặc bị từ chối nếu đầu vào bị từ chối. Sử dụng thứ này để khẳng định nếu một database đọc và ghi thành công.
  + 
* **clearFirestoreData({ projectId: <name> }) => Promise**
  + Phương pháp này xóa tất cả dữ liệu liên quan đến một project cụ thể trong trường hợp Firestore đang chạy cục bộ. Sử dụng phương pháp này để dọn dẹp sau khi thử nghiệm.
  + 

### Tạo báo cáo thử nghiệm

Sau khi chạy một vài bộ thử nghiệm, bạn có thể truy cập các báo cáo thống kê thử nghiệm cho biết cách từng quy tắc bảo mật của bạn được đánh giá.

Để nhận báo cáo, hãy truy vấn điểm cuối được hiển thị trên trình giả lập trong khi nó đang chạy. Đối với phiên bản thân thiện với trình duyệt, hãy sử dụng URL sau:

[***http://localhost:8080/emulator/v1/projects/<project\_id>:ruleCoverage.html***](http://localhost:8080/emulator/v1/projects/%3cproject_id%3e:ruleCoverage.html)

Điều này phá vỡ các quy tắc của bạn thành các biểu thức và biểu thức con mà bạn có thể di chuột để biết thêm thông tin, bao gồm số lượng đánh giá và giá trị được trả về. Đối với phiên bản JSON thô của dữ liệu này, hãy bao gồm URL sau trong truy vấn của bạn:

***http://localhost:8080/emulator/v1/projects/<project\_id>:ruleCoverage***

### Điểm khác nhau giữa trình giả lập và sản phẩm thật:

1. Bạn không nhất thiết phải tạo một project Cloud Firestore. Trình giả lập tự động tạo bất kỳ instance nào được truy cập.
2. Trình giả lập Cloud Firestore không hoạt động với luồng Xác thực Firebase thông thường. Thay vào đó, trong SDK thử nghiệm Firebase, chúng tôi đã cung cấp phương thức initizeTestApp() trong module thử nghiệm, trong đó có một trường xác thực. Firebase xử lý tạo ra bằng phương pháp này sẽ cư xử như thể nó đã xác thực thành công như bất cứ điều gì thực thể mà bạn cung cấp. Nếu bạn chuyển qua null, nó sẽ hoạt động như một người dùng không được xác thực (ví dụ: quy tắc null sẽ không thành công).

### Khắc phục sự cố đã biết:

Khi bạn sử dụng trình giả lập Cloud Firestore, bạn có thể gặp phải các vấn đề đã biết sau đây. Thực hiện theo hướng dẫn bên dưới để khắc phục mọi hành vi bất thường mà bạn gặp phải. Các ghi chú này được viết với SDK thử nghiệm Firebase, nhưng các cách tiếp cận chung có thể áp dụng cho bất kỳ SDK Firebase nào.

#### \*Hành vi kiểm tra không nhất quán:

Nếu các kiểm thử của bạn thỉnh thoảng vượt qua và thất bại, ngay cả khi không có bất kỳ thay đổi nào đối với các chúng, bạn có thể cần phải xác minh rằng chúng được sắp xếp đúng. Hầu hết các tương tác với trình giả lập là không đồng bộ, vì vậy hãy kiểm tra kỹ xem tất cả mã async có được sắp xếp đúng hay không. Bạn có thể sửa lỗi trình tự bằng cách xâu chuỗi các promise hoặc sử dụng await một cách tự do.

Cụ thể, xem lại các hoạt động không đồng bộ sau:

* Đặt quy tắc bảo mật, ví dụ như **firebase.loadFirestoreRules**.
* Đọc và ghi dữ liệu, ví dụ, với **db.collection("user").doc("alice").Get()**.
* Các xác nhận hoạt động, bao gồm **firebase.assertSucceeds** và **firebase.assertFails.**

#### \*Các kiểm thử chỉ qua mỗi lần đầu tiên khi bạn tải trình giả lập:

Trình giả lập là bộ máy. Nó lưu trữ tất cả dữ liệu được ghi vào bộ nhớ, vì vậy mọi dữ liệu sẽ bị mất bất cứ khi nào trình giả lập tắt. Nếu bạn đang chạy nhiều thử nghiệm với cùng một id project, mỗi thử nghiệm có thể tạo ra dữ liệu có thể ảnh hưởng đến các thử nghiệm tiếp theo. Bạn có thể sử dụng bất kỳ phương pháp nào sau đây để bỏ qua hành vi này:

* Một ID project cho một thử nghiệm.
* Cơ cấu lại các bài kiểm thử của bạn để chúng không tương tác với dữ liệu được viết trước đó (ví dụ: sử dụng một collection khác nhau cho mỗi bài kiểm tra).
* Xóa tất cả các dữ liệu được viết trong một kiểm thử.

#### \* Thiết lập thử nghiệm rất phức tạp:

Bạn có thể muốn kiểm tra các tình huống mà Quy tắc bảo mật Cloud Firestore của bạn không thực sự cho phép. Ví dụ: việc kiểm tra xem người dùng chưa được xác thực có thể chỉnh sửa dữ liệu hay không, vì bạn không thể chỉnh sửa dữ liệu với tư cách là người dùng chưa được xác thực.

Nếu quy tắc của bạn khiến việc thiết lập thử nghiệm trở nên phức tạp, hãy thử sử dụng ứng dụng khách được ủy quyền quản trị để bỏ qua các quy tắc. Bạn có thể làm điều này với firebase.initializeAdminApp. Đọc và viết từ các khách hàng được ủy quyền của quản trị viên bỏ qua các quy tắc và không kích hoạt các lỗi PERMISSION\_DENIED.

## Truy vấn dữ liệu một cách an toàn

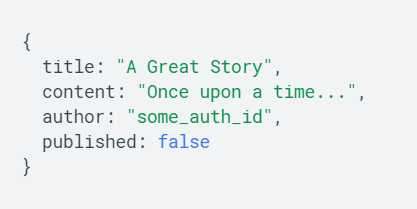
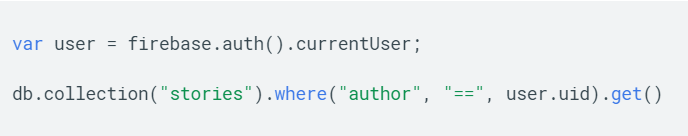
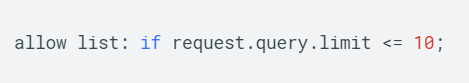
Trang này dựa trên các khái niệm trong Cấu trúc quy tắc bảo mật và điều kiện viết cho quy tắc bảo mật để giải thích cách quy tắc bảo mật của Cloud Firestore tương tác với các truy vấn. Cần xem xét kỹ hơn về cách các quy tắc bảo mật ảnh hưởng đến các truy vấn bạn có thể viết và mô tả cách đảm bảo các truy vấn của bạn sử dụng các ràng buộc giống như các quy tắc bảo mật của bạn. Trang này cũng mô tả cách viết quy tắc bảo mật để cho phép hoặc từ chối truy vấn dựa trên các thuộc tính truy vấn như limit và orderBy.

### \* Quy tắc không phải là bộ lọc

Khi viết truy vấn để truy xuất document, hãy nhớ rằng các quy tắc bảo mật không phải là bộ lọc, tất cả đều phụ thuộc vào truy vấn. Để tiết kiệm thời gian và tài nguyên của bạn, Cloud Firestore đánh giá một truy vấn dựa trên tập kết quả tiềm năng của nó thay vì các giá trị trường thực tế cho tất cả các document của bạn. Nếu một truy vấn có khả năng trả về các document mà khách hàng không có quyền đọc, toàn bộ yêu cầu không thành công.

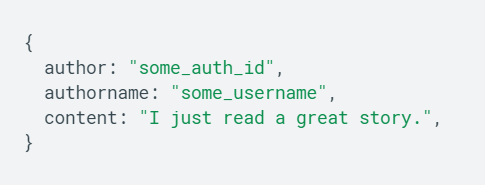
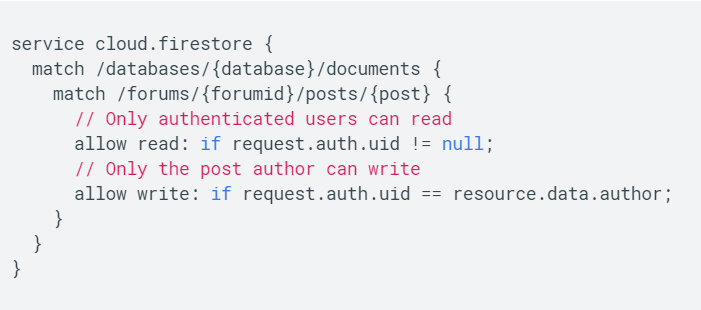
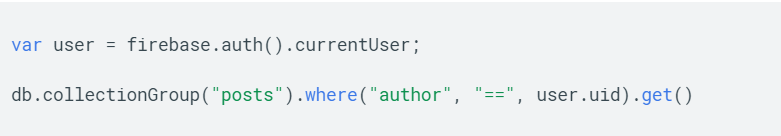
### \* Truy vấn và quy tắc bảo mật

Như các ví dụ dưới đây chứng minh, bạn phải viết các truy vấn của mình để phù hợp với các ràng buộc của các quy tắc bảo mật của bạn.

* Bảo mật và truy vấn document dựa trên **auth.uid**
  + Ví dụ sau minh họa cách viết truy vấn để truy xuất document được bảo vệ theo quy tắc bảo mật. Hãy xem xét một cơ sở dữ liệu có chứa một tập hợp các document story:
  + **/stories/{storyid}**
  + 
  + Ngoài trường title và content, mỗi document lưu trữ trường author và published được sử dụng cho việc kiểm soát truy cập. Các ví dụ này giả định ứng dụng sử dụng Xác thực Firebase để đặt trường author thành UID của người dùng đã tạo document. Xác thực Firebase cũng điền vào biến request.auth trong các quy tắc bảo mật.
  + Quy tắc bảo mật sau sử dụng các biến request.auth và resource.data để hạn chế quyền truy cập đọc và viết cho mỗi story đối với tác giả của nó:
  + 
  + Giả định rằng ứng dụng của bạn bao gồm một trang hiển thị cho người dùng danh sách các story mà họ là tác giả. Bạn có thể mong đợi rằng mình có thể sử dụng truy vấn sau đây để điền trang này. Tuy nhiên, truy vấn này sẽ thất bại, vì nó không bao gồm các ràng buộc giống như các quy tắc bảo mật của bạn:
  + -> **Không hợp lệ**: Ràng buộc truy vấn không khớp với các ràng buộc quy tắc bảo mật
  + 
  + Truy vấn không thành công ngay cả khi người dùng hiện tại thực sự là tác giả của mọi document story. Lý do cho hành vi này là khi Cloud Firestore áp dụng các quy tắc bảo mật của bạn, nó đánh giá truy vấn dựa trên tập kết quả tiềm năng của nó, chứ không phải dựa trên các thuộc tính thực tế của document trong cơ sở dữ liệu của bạn. Nếu một truy vấn có khả năng bao gồm các document vi phạm quy tắc bảo mật của bạn, truy vấn sẽ thất bại.
  + Ngược lại, truy vấn sau thành công, bởi vì nó bao gồm cùng một ràng buộc trên trường tác giả như các quy tắc bảo mật:
  + -> **Hợp lệ**: Ràng buộc truy vấn khớp với các ràng buộc quy tắc bảo mật
  + 
* Bảo mật và truy vấn document dựa trên một trường:
  + Để thể hiện rõ hơn sự tương tác giữa các truy vấn và quy tắc, các quy tắc bảo mật bên dưới mở rộng quyền truy cập cho phép toàn bất cứ người dùng nào cũng có thể đọc document story trong tập stories khi mà trường **published** được đặt thành true:
  + 
  + Truy vấn cho các trang published phải bao gồm các ràng buộc giống như các quy tắc bảo mật:
  + 
  + Ràng buộc truy vấn .where("published", "==", true) đảm bảo rằng resource.data.published là đúng với mọi kết quả. Do đó, truy vấn này đáp ứng các quy tắc bảo mật và được phép đọc dữ liệu.
* Đánh giá các ràng buộc về truy vấn:
  + Các quy tắc bảo mật của bạn cũng có thể chấp nhận hoặc từ chối các truy vấn dựa trên các ràng buộc của chúng. Biến request.query chứa các thuộc tính giới hạn, offset và orderBy của truy vấn:
  + 
  + Các quy tắc sau đây trình bày cách như thế nào để viết các quy tắc bảo mật thứ đánh giá các ràng buộc được đặt trên các truy vấn. Ví dụ này mở rộng quy tắc stories trước đó với các thay đổi sau:
    - Bộ quy tắc tách quy tắc read thành quy tắc get và list.
    - Quy tắc get giới hạn truy xuất các document đơn lẻ thành document công khai hoặc document mà người dùng đã ủy quyền.
    - Quy tắc list áp dụng các hạn chế tương tự như get nhưng cho các truy vấn. Nó cũng kiểm tra giới hạn truy vấn, sau đó từ chối mọi truy vấn không có giới hạn hoặc với giới hạn lớn hơn 10.
    - Bộ quy tắc xác định hàm AuthorOrPublished () để tránh trùng lặp mã.



### Các truy vấn nhóm bộ kết hợp và quy tắc bảo mật (Collection group queries and security rules)

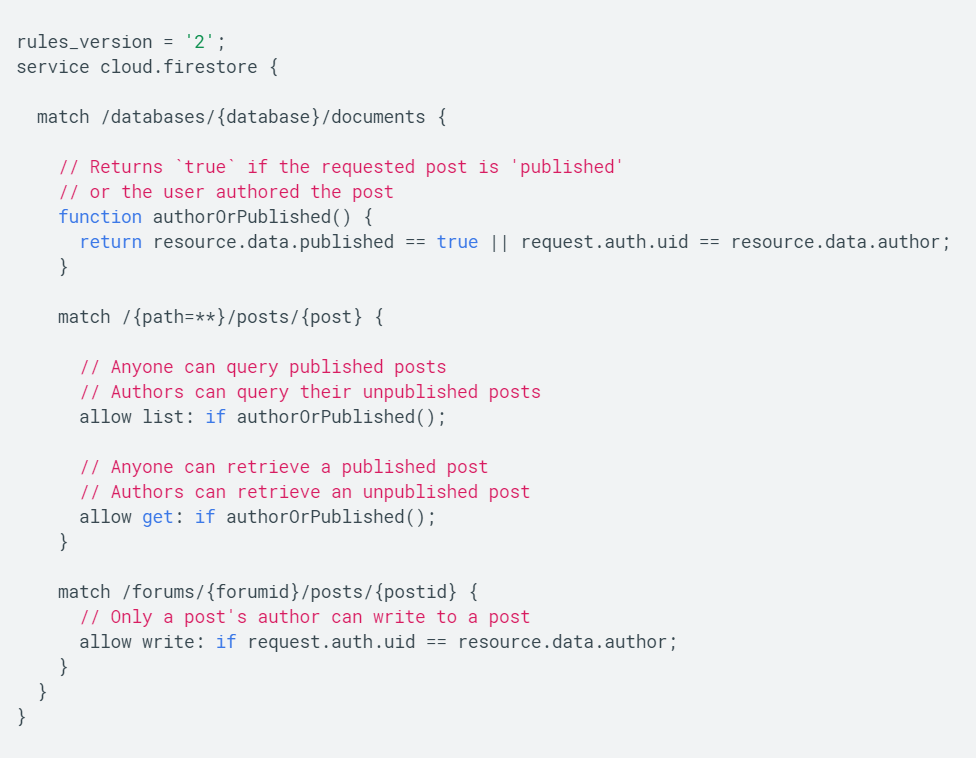
* Theo mặc định, các truy vấn được đặt trong một collection duy nhất và chúng chỉ lấy kết quả từ collection đó. Với các truy vấn nhóm collection, bạn có thể truy xuất kết quả từ một nhóm collection bao gồm tất cả các collection có cùng ID. Phần này mô tả cách bảo mật các truy vấn nhóm collection của bạn bằng các quy tắc bảo mật.
* Bảo mật và truy vấn document dựa trên các nhóm collection:
  + Trong quy tắc bảo mật của bạn, bạn phải cho phép rõ ràng cho phép truy vấn nhóm collection bằng cách viết quy tắc cho nhóm collection:
    - Đảm bảo rules\_version = '2'; là dòng đầu tiên của quy tắc của bạn. Các truy vấn nhóm collection yêu cầu ký tự đệ quy mới {name = \*\*} của quy tắc bảo mật phiên bản 2.
    - Viết quy tắc cho nhóm collection của bạn bằng cách sử dụng **match /{path=\*\*}/[COLLECTION\_ID]/{doc}.**
  + Ví dụ, hãy xem xét một diễn đàn được tổ chức thành các document forum có chứa các posts con:
  + **/forums/{forumid}/posts/{postid}**
  + 
  + Trong ứng dụng này, chúng ta làm cho các bài đăng có thể chỉnh sửa bởi chủ sở hữu của họ và người dùng được xác thực có thể đọc được:
  + 
  + Bất kỳ ai đã đăng nhập cũng có thể truy xuất các bài đăng của bất kỳ diễn đàn nào:
  + 
  + Nhưng nếu bạn muốn hiển thị cho người dùng hiện tại bài đăng của họ trên tất cả các diễn đàn thì sao? Bạn có thể sử dụng truy vấn nhóm collection để truy xuất kết quả từ tất cả các collection bài đăng:
  + 
  + Trong quy tắc bảo mật của bạn, bạn phải cho phép truy vấn này bằng cách viết quy tắc đọc hoặc liệt kê cho nhóm collection posts:
  + 
  + Tuy nhiên, lưu ý rằng các quy tắc này sẽ áp dụng cho tất cả các collection có ID posts, bất kể phân cấp. Ví dụ:



* Các truy vấn nhóm collection an toàn dựa trên một trường:
  + Giống như các truy vấn bộ đơn, các truy vấn nhóm collection cũng phải đáp ứng các ràng buộc được đặt theo quy tắc bảo mật của bạn. Ví dụ: chúng ta có thể thêm một trường **published** vào mỗi bài đăng trên diễn đàn như chúng ta đã làm trong ví dụ **stories** ở trên:
  + **/forums/{forumid}/posts/{postid}**



* + Sau đó chúng ta có thể viết các quy tắc cho nhóm collection **posts** dựa trên trạng thái **published** và **author**:



* + Với web, IOS, android client, có thể theo những quy tắc sau:
    - Bất cứ ai cũng có thể lấy các bài đăng được xuất bản trong một diễn đàn:



* + - Bất cứ ai cũng có thể truy xuất các bài đăng được xuất bản của tác giả trên tất cả các diễn đàn:

**db.collectionGroup("posts").where("author", "==", "some\_auth\_id").where('published', '==', true).get()**

* + - Tác giả có thể truy xuất tất cả các bài đăng được xuất bản và chưa được công bố của họ trên tất cả các diễn đàn:



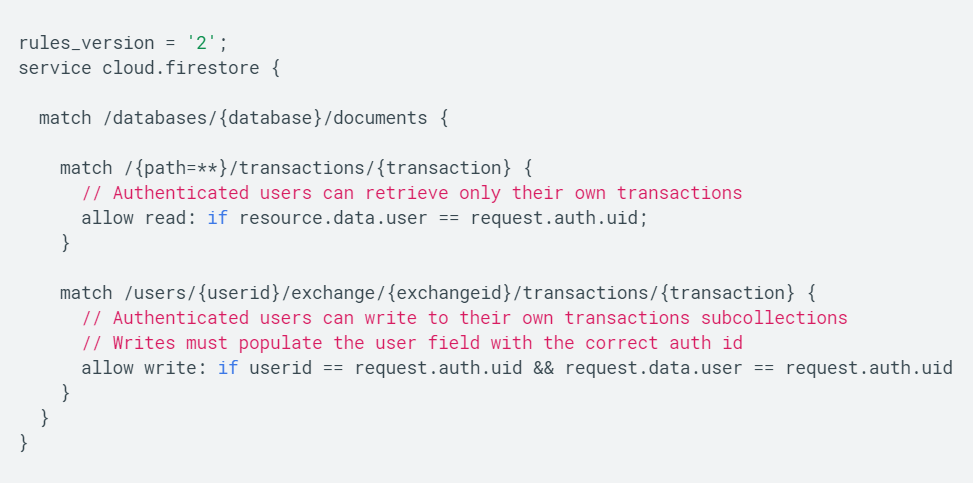
* Bảo mật và truy vấn document dựa trên nhóm collection và đường dẫn document:
  + Trong một số trường hợp, bạn có thể muốn hạn chế truy vấn nhóm collection dựa trên đường dẫn document. Để tạo các hạn chế này, bạn có thể sử dụng các kỹ thuật tương tự để bảo mật và truy vấn document dựa trên một trường.
  + Hãy xem xét một ứng dụng theo dõi từng giao dịch của người dùng giữa một số trao đổi chứng khoán và tiền điện tử:

**/users/{userid}/exchange/{exchangeid}/transactions/{transaction}**

* + 
  + Chú ý trường **user**. Mặc dù chúng ta biết người dùng nào sở hữu document **transaction** từ đường dẫn của document, chúng ta sao chép thông tin này trong mỗi document **transaction** vì nó cho phép chúng tôi thực hiện hai điều:
    - Viết các truy vấn nhóm collection được giới hạn trong các document bao gồm một **/users/{userid}** cụ thể trong đường dẫn document của chúng. Ví dụ:



* + - Thực thi hạn chế này đối với tất cả các truy vấn trên nhóm thu thập **transaction** để một người dùng không thể truy xuất document **transaction** của người dùng khác.
  + Chúng ta thực thi hạn chế này trong các quy tắc bảo mật của mình và bao gồm xác thực dữ liệu cho trường người dùng:



# Demo

Trong phần demo cho Firebase/Firestore, nhóm sử dụng một đề tài về việc xây dựng một ứng dụng dành cho việc hỗ trợ quản lý, đăng ký khám bệnh cho bệnh nhân.

**Mô tả**  
Mô tả đề tài như sau:

“*Ứng dụng hỗ trợ đăng ký khám bệnh, thanh toán viện phí cho bệnh nhân*

*Ứng dụng này cho phép người quản lý (nhân viên ở quầy viện phí) tiến hành xem danh sách bệnh nhân hiện tại, danh sách bác sĩ hiện tại. Thực hiện đăng ký khám bệnh cho bệnh nhân với một bác sĩ nhất định có trong bệnh viện. Sau khi khám bệnh xong bệnh nhân sẽ cần phải thanh toán tiền khám bệnh theo giá tiền quy định cho mỗi lần khám (giá tiền là 70 ngàn đồng), nếu bệnh nhân nợ tiền khám gấp 2 lần số tiền quy định sẽ không được phép đăng ký khám mới mà cần phải trả viện phí mới được phép đăng ký. Nếu là bệnh nhân mới thì ứng dụng sẽ hỗ trợ đăng ký bệnh nhân (trả về một mã bệnh nhân với độ dài 5 ký tự), bệnh nhân cần phải khai báo tên, tuổi, giới tính theo yêu cầu. Có thể kiểm tra bệnh nhân, số tiền viện phí cần thanh toán.*”

**Các nút chức năng**

- Xem danh sách bệnh nhân -> tìm kiếm bệnh nhân

- Thêm bệnh nhân

- Đăng ký khám bệnh cho bệnh nhân

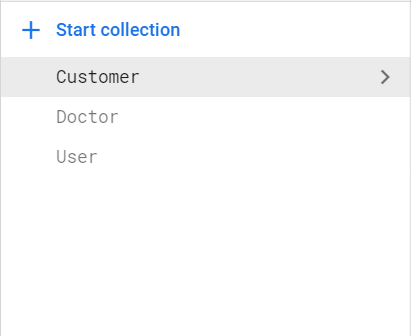
- Thanh toán khám bệnh cho bệnh nhân

- Hủy khám bệnh / Kết thúc khám bệnh cho bệnh nhân

**Cơ sở dữ liệu thiết kế trên Firestore:**

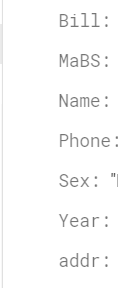
Trên firestore, việc thiết kế cơ sở dữ liệu khác với việc thiết kế trong các chương trình SQL, nó không cần thiết ta phải tạo table, nhưng phải tạo một thứ gần giống với table là Bộ lưu trữ, hay còn gọi là Collection. Trong một Collection sẽ có thể tạo một Document, tập dữ liệu lưu trữ, trong document có thể tạo tiếp tục Bộ lưu trữ con, Subcollection hoặc tạo các trường thuộc tính cho nó.

Vì đây là thiết kế một phần demo cho đề tài cũng như nội dung nghiên cứu của nhóm, csdl được thiết kế như sau:

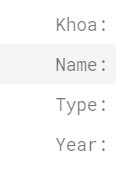


Với 3 Collection:

Customer sẽ có các trường như sau:



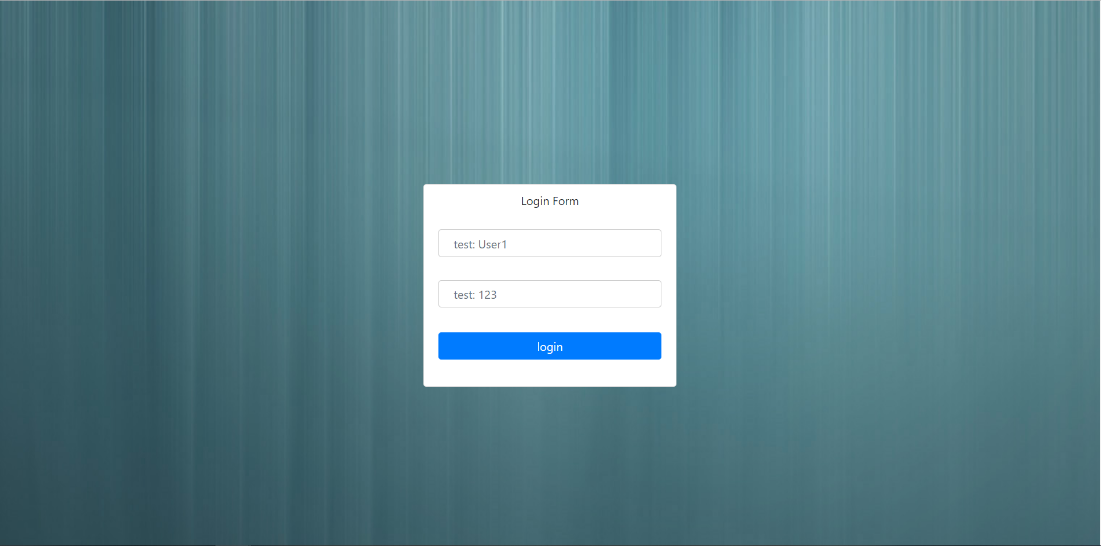
Doctor sẽ có các trường như sau:



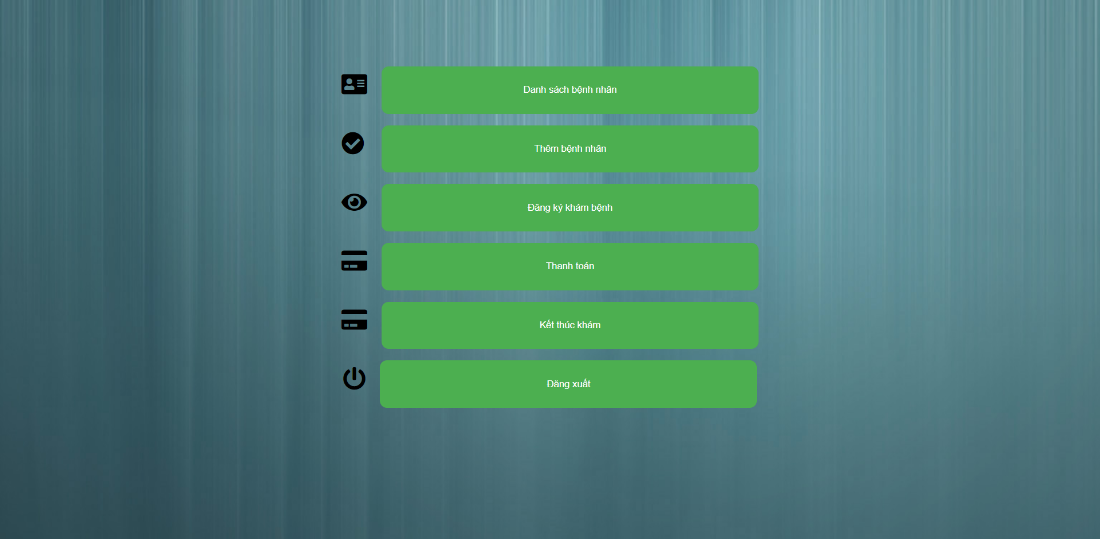
User sẽ có các trường như sau:



**Màn hình đăng nhập:**



**Màn hình chính:**



Các nút chức năng khi thực thi sẽ được mô tả kĩ hơn trong phần demo thực thi

# Tài liệu tham khảo

<https://en.wikipedia.org/wiki/Firebase>

<https://firebase.google.com/>

<https://firebase.google.com/docs/firestore>

<https://viblo.asia/p/cung-tim-hieu-ve-firebase-cloud-firestore-ByEZk32YZQ0>