Trường Đại học Sư Phạm Thành Phố Hồ Chí Minh

Khoa: Công Nghệ Thông Tin

**BÁO CÁO VỀ FIREBASE**

Môn: Cơ Sở Dữ Liệu Nâng Cao

Giảng viên bộ môn: Th. Lương Trần Hy Hiến

Sinh viên thực hiện: Nguyễn Khắc Duy 41.01.104.021

Văn Ý Linh 41.01.104.053

Thành phố Hồ Chí Minh, 14 tháng 11 năm 2018.

**LỜI CẢM ƠN**

Trong thời gian làm bài báo cáo này, nhóm em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy và bạn bè.

Nhóm em xin gửi lời cảm ơn chân thành đến Th.Lương Trần Hy Hiến, giảng viên Bộ môn Cơ Sở Dữ Liệu nâng cao, trường Đại học Sư Phạm TP.Hồ Chí Minh, người đã tận tình hướng dẫn, chỉ bảo nhóm em trong suốt quá trình làm bài báo cáo này.

Cuối cùng, nhóm em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên nhóm em trong suốt quá trình học tập và hoàn thành bài báo cáo này.

**Nhận xét, đánh giá của giảng viên**

**.....................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................**

Mục Lục

[1. noSQL Database 6](#_Toc529930378)

[1. 1. Giới thiệu 6](#_Toc529930379)

[1. 2. Ưu điểm 7](#_Toc529930380)

[1. 3. Nhược điểm 7](#_Toc529930381)

[1. 4. Phân loại 7](#_Toc529930382)

[1. 5. Tỉ lệ sử dụng noSQL 9](#_Toc529930383)

[2. Firebase 10](#_Toc529930384)

[2. 1. Firebase Realtime DB 10](#_Toc529930385)

[2. 1. 1. Ưu điểm 10](#_Toc529930386)

[2. 1. 2. Truy vấn và xử lí dữ liệu 12](#_Toc529930387)

[2. 1. 3. Làm việc với danh sách dữ liệu trên web 15](#_Toc529930388)

[2. 2. Cloud Firestore 16](#_Toc529930389)

[2. 2. 1. So sánh Firebase Realtime DB và Cloud Firestore 16](#_Toc529930390)

[3. Demo 19](#_Toc529930391)

1. noSQL Database
   1. Giới thiệu

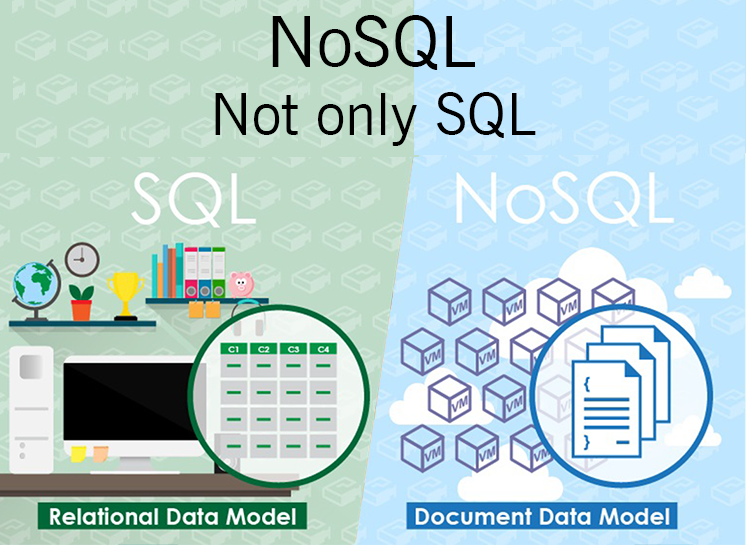
Vào năm 1998, noSQL được giới thiệu lần đầu tiên, sử dụng làm tên gọi chung cho các *lightweight open source relational database* – cơ sở dữ liệu quan hệ nguồn mở nhỏ.

2009, nhân viên của Rackspace giới thiệu lại thuật ngữ noSQL trong một hội thảo về cơ sở dữ liệu nguồn mở phân tán.

NoSQL đánh dấu bước phát triển của thế hệ cơ sở dữ liệu mới : distributed (phân tán) + non-relational (không ràng buộc).

NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value (1 đặc trưng về dữ liệu trong JSON).

NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS về tốc độ, tính năng, khả năng mở rộng, memory cache,... Vì NoSQL không hạn chế việc mở rộng dữ liệu nên tồn tại nhiều nhược điểm như: sự phục thuộc của từng bản ghi, tính nhất quán, toàn vẹn dữ liệu,....



* 1. Ưu điểm

Là nguồn mở.

Các cơ sở dữ liệu no SQL khác nhau cho những dự án khác nhau.

Phù hợp với công nghệ đám mây.

Có khả năng lưu trữ dữ liệu với lượng cực lớn.

Truy vấn dữ liệu tốc độ cao mà không đòi hỏi quá nhiều về năng lực phần cứng cũng như tài nguyên hệ thống.

Khả năng chịu lỗi cao.

* 1. Nhược điểm

Không có lược đồ hỗ trợ.

Thiếu tính nhất quán.

Lock-in ( hầu hết các hệ thống noSQL đều tương tự về khái niệm, tuy nhiên cách thực hiện lại rất khác nhau. Mỗi hệ thống sẽ có cơ chế truy vấn dữ liệu và quản lí riêng).

Vấn đề về tương thích cũng là một mối quan ngại lớn. Bởi vì mỗi cơ sở dữ liệu noSQL sẽ có các giao diện lập trình ứng dụng API riêng và những sự riêng biệt khác, nên việc tương thích với nhau là không có khả năng.

* 1. Phân loại

Có bốn loại noSQL phổ biến.

Mỗi loại đều có các thuộc tính và giới hạn riêng. Không có một giải pháp duy nhất nào tốt hơn tất cả các giải pháp khác, tuy nhiên có một số cơ sở dữ liệu tốt hơn để giải quyết các vấn đề cụ thể. Để làm rõ cơ sở dữ liệu NoSQL, hãy thảo luận các loại phổ biến nhất:

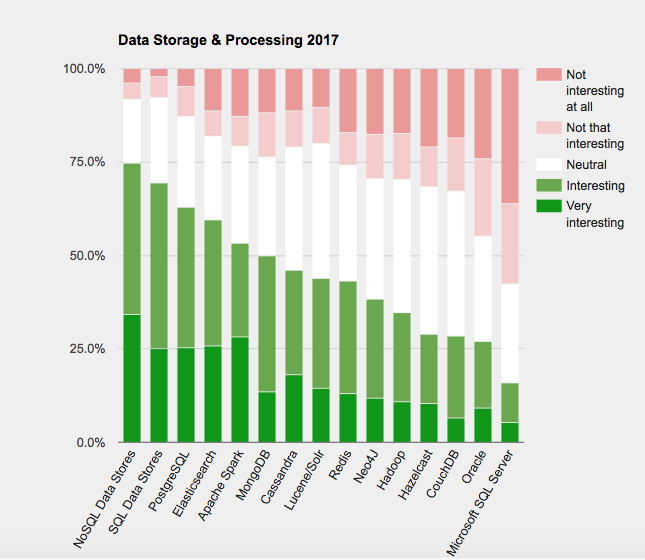
a. Key-value stores. Lưu trữ kiểu key-value là kiểu lưu trữ dữ liệu NoSQL đơn giản nhất sử dụng từ một API. Chúng ta có thể nhận được giá trị cho khóa, đặt một giá trị cho một khóa, hoặc xóa một khóa từ dữ liệu. Ví dụ, giá trị là ‘blob’ được lưu trữ thì chúng ta không cần quan tâm hoặc biết những gì ở bên trong. Từ các cặp giá trị được lưu trữ luôn luôn sử dụng truy cập thông qua khóa chính và thường có hiệu năng truy cập tốt và có thể dễ dàng thu nhỏ lại. Một vài cơ sở dữ liệu key-value phổ biến là Riak, Redis(thường dùng phía server), memcached, Berkeley DB, HamsterDB, Amazon DynamoDB(mã nguồn đóng), Project Voldemort và Couchbase. Tất cả các cơ sở dữ liệu kiểu key-value đều không giống nhau, có rất nhiều điểm khác nhau giữa các sản phẩm. Ví dụ, dữ liệu của memcached không được nhất quán trong khi ngược lại với Riak. Đấy là những điểm nổi bật quan trọng khi chọn giải pháp phù hợp để sử dụng. Cụ thể hơn là khi chúng ta cần cài đặt caching cho nội dung yêu thích của người dùng, cài đặt sử dụng memcached có nghĩa là khi các nút hỏng hết dẫn tới dữ liệu bị mất và cần phải làm mới lại từ hệ thống nguồn. Tuy nhiên, nếu chúng ta lưu trữ cùng dữ liệu đó trong Riak, chúng ta không cần lo lắng về việc mất dữ liệu nhưng cần phải xem xét việc cập nhật trạng thái của dữ liệu như thế nào. Điều này là quan trọng không chỉ cho chọn cơ sở dữ liệu key-value cho hệ thống và còn quan trọng cho việc chọn cơ sở dữ liệu key-value nào.

b. Column-oriented databases (column-family). Cơ sở dữ liệu column-family lưu trữ dữ liệu trong nhiều cột trong mỗi dòng với key cho từng dòng. Column families là một nhóm các dữ liệu liên quan được truy cập cùng với nhau. Ví dụ, với khách hàng, chúng ta thường xuyên sử dụng thông tin cá nhân trong cùng một lúc chứ không phải hóa đơn của họ. Cassandra là một trong số cơ sở dữ liệu column-family phổ biến. Ngoài ra còn có một số cơ sở dữ liệu khác như HBase, Hypertable và Amazon DynamoDB. Cassandra có thể được miêu tả nhanh và khả năng mở rộng dễ dàng với các thao tác viết thông qua các cụm. Các cụm không có node master, vì thế bất kỳ việc đọc và ghi nào đểu có thể được xử lý bởi bất kỳ node nào trong cụm.

c. Graph databases Kiểu đồ thị này cho phép bạn lưu trữ các thực thể và quan hệ giữa các thực thể. Các đối tượng này còn được gọi là các nút, trong đó có các thuộc tính. Mỗi nút là một thể hiện của một đối tượng trong ứng dụng. Quan hệ được gọi là các cạnh, có thể có các thuộc tính. Cạnh có ý nghĩa định hướng; các nút được tổ chức bởi các mối quan hệ. Các tổ chức của đồ thị cho phép các dữ liệu được lưu trữ một lần và được giải thích theo nhiều cách khác nhau dựa trên các mối quan hệ. Thông thường, khi chúng ta lưu trữ một cấu trúc đồ thị giống như trong RDBMS, nó là một loại duy nhất của mối quan hệ. Việc tăng thêm một mối quan hệ có nghĩa là rất nhiều thay đổi sơ đồ và di chuyển dữ liệu, mà không phải là trường hợp khó khi chúng ta đang sử dụng cơ sở dữ liệu đồ thị. Trong cơ sở dữ liệu đồ thị, băng qua các thành phần tham gia hoặc các mối quan hệ là rất nhanh. Các mối quan hệ giữa các node không được tính vào thời gian truy vấn nhưng thực sự tồn tại như là một mối quan hệ. Đi qua các mối quan hệ là nhanh hơn so với tính toán cho mỗi truy vấn. Có rất nhiều cơ sở dữ liệu đồ thị có sẵn, chẳng hạn như Neo4J, Infinite Graph, OrientDB, hoặc FlockDB (đó là một trường hợp đặc biệt: một cơ sở dữ liệu đồ thị mà chỉ hỗ trợ các mối quan hệ duy nhất chuyên sâu hoặc danh sách kề, nơi mà bạn không thể đi qua nhiều hơn một mức độ sâu sắc đối với các mối quan hệ ).

d. Document Oriented databases Tài liệu là nguyên lý chính của cơ sở dữ liệu kiểu dữ liệu. Dữ liệu lưu trữ và lấy ra là các tài liệu với định dạng XML, JSON, BSON,… Tài liệu miêu tả chính nó, kế thừa từ cấu trúc dữ liệu cây. Có thể nói cơ sở dữ liệu tài liệu là 1 phần của key-value. Cơ sở dữ liệu kiểu tài liệu như MongoDB cung cấp ngôn ngữ truy vấn đa dạng và cúc trúc như là cơ sở dữ liệu như đánh index,… Một số cơ sở dữ liệu tài liệu phổ biến mà chúng ta hay gặp là MongoDB, CouchDB, Terastore, OrientDB, RavenDB.

* 1. Tỉ lệ sử dụng noSQL



1. Firebase

Firebase là một nền tảng ứng dụng di động và web với các công cụ và platform được thiết kế để giúp các lập trình viên xây dựng các ứng dụng có chất lượng cao.

Các tính năng cơ bản của Firebase

[1. Realtime Database – Cơ sở dữ liệu thời gian thực](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

[2. Firebase Authentication – Hệ thống xác thực của Firebase](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

[3. Firebase Hosting](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

[4. Cloud Messaging](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

[5. Firebase Storage](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

[6. Firebase Test Lab và Crash Reporting](https://blog.vietnamlab.vn/2017/06/30/firebase-p1-gioi-thieu-firebase/)

* 1. Firebase Realtime DB

Firebase Realtime database là một cloud hosted database hỗ trợ đa nền tảng: Android, IOS và Web. Tất cả dữ liệu được lưu trữ ở định dạng JSON và với bất kể một sự thay đổi dữ liệu nào thì có sự phản hồi ngay lập tức, hiển thị đồng bồ trên các nền tảng và các thiết bị. Bài hướng dân này xây dựng nhằm thể hiện sự phản hồi theo thời gian tực của apps một cách đơn giản khi sử dụng Firebase Realtime database.

* + 1. Ưu điểm



1. Triển khai ứng dụng cực nhanh

Với Firebase bạn có thể giảm bớt rất nhiều thời gian cho việc viết các dòng code để quản lý và đồng bộ cơ sở dữ liệu, mọi việc sẽ diễn ra hoàn toàn tự động với các API của Firebase. Không chỉ có vậy Firebase còn hỗ trợ đã nền tảng nên bạn sẽ càng đỡ mất thời gian rất nhiều khi ứng dụng bạn muốn xây dựng là ứng dụng đa nền tảng.

Không chỉ nhanh chóng trong việc xây dựng database, Google Firebase còn giúp ta đơn giản hóa quá trình đăng kí và đăng nhập vào ứng dụng bằng các sử dụng hệ thống xác thực do chính Firebase cung cấp.

Firebase Realtime Database sử dụng đồng bộ dữ liệu mối khi dữ liệu có thay đổi, mọi thiết bị được kết nối sẽ nhận được thay đổi trong vài mili giây.

2. Bảo mật

Firebase hoạt động dựa trên nền tảng cloud và thực hiện kết nối thông qua giao thức bảo mật SSL, chính vì vậy bạn sẽ bớt lo lắng rất nhiều về việc bảo mật của dữ liệu cũng như đường truyền giữa client và server. Không chỉ có vậy, việc cho phép phân quyền người dùng database bằng cú pháp javascipt cũng nâng cao hơn nhiều độ bảo mật cho ứng dụng của bạn, bởi chỉ những user mà bạn cho phép mới có thể có quyền chỉnh sửa cơ sở dữ liệu.

Firebase Realtime Database có thể truy cập từ một thiết bị mobile hoặc trình duyệt web. Nó không cần một ứng dụng server nào cả. Bảo mật và xác thực dữ liệu có thể thông qua các Rule bảo mật của Firebase Realtime Database, các rule được thực thi khi dữ liệu được đọc hoặc ghi.

Dữ liệu được lưu trữ ở local trong khi offline và thực hiện đồng bộ khi được kết nối

Khi thiết bị khôi phục kết nối, Realtime DB sẽ tự động đồng bộ hóa các thay đổi trong dữ liệu local với các bản cập nhật từ xa xảy ra khi client ở trạng thái offline, Tự động tích hợp nếu có xung đột.

Firebase Realtime Database Security Rules cung cấp một ngôn ngữ quy tắc dựa trên biểu thức linh hoạt xác định cấu trúc dữ liệu và thời gian để cho phép đọc và ghi dữ liệu.

3. Tính linh hoạt và khả năng mở rộng

Sử dụng Firebase sẽ giúp bạn dễ dàng hơn rất nhiều mỗi khi cần nâng cấp hay mở rộng dịch vụ. Ngoài ra firebase còn cho phép bạn tự xây dựng server của riêng mình để bạn có thể thuận tiện hơn trong quá trình quản lý.

Việc Firebase sử dụng NoSQL, giúp cho database của bạn sẽ không bị bó buộc trong các bảng và các trường mà bạn có thể tùy ý xây dựng database theo cấu trúc của riêng bạn.

4. Sự ổn định

Firebase hoạt động dựa trên nền tảng cloud đến từ Google vì vậy hầu như bạn không bao giờ phải lo lắng về việc sập server, tấn công mạng như DDOS, tốc độ kết nối lúc nhanh lúc chậm, … nữa, bởi đơn giản là Firebase hoạt động trên hệ thống server của Google. Hơn nữa nhờ hoạt động trên nền tảng Cloud nên việc nâng cấp, bảo trì server cũng diễn ra rất đơn giản mà không cần phải dừng server để nâng cấp như truyền thống.

5. Giá thành

Google Firebase có rất nhiều gói dịch vụ với các mức dung lượng lưu trữ cũng như băng thông khác nhau với mức giá dao động từ Free đến $1500 đủ để đáp ứng được nhu cầu của tất cả các đối tượng. Chính vì vậy bạn có thể lựa chọn gói dịch vụ phù hợp nhất với nhu cầu của mình. Điều này giúp bạn tới ưu hóa được vốn đầu tư và vận hành của mình tùy theo số lượng người sử dụng. Ngoài ra bạn còn không mất chi phí để bảo trì, nâng cấp, khắc phục các sự cố bởi vì những điều này đã có Firebase lo.

* + 1. Truy vấn và xử lí dữ liệu



1.Initialize the Realtime DB JavaScript SDK

  // Set the configuration for your app  
  // TODO: Replace with your project's config object  
  var config = {  
    apiKey: "***apiKey***",  
    authDomain: "***projectId***.firebaseapp.com",  
    databaseURL: "https://***databaseName***.firebaseio.com",  
    storageBucket: "***bucket***.appspot.com"  
  };  
  firebase.initializeApp(config);  
  
  // Get a reference to the database service  
  var database = firebase.database();

2.Structure DB

Tất cả các dữ liệu của Firebase Realtime DB đều được lưu trữ dưới dạng đối tượng JSON.

Có thể xem cơ sở dữ liệu được tổ chức dưới dạng cây JSON, mỗi một dữ liệu thâm vào cây, sẽ trở thành một nút trong cấu trúc với khóa có liên quan.

Phương án làm phẳng cấu trúc dữ liệu – *Flatten data structures*

{  
  // Chats contains only meta info about each conversation  
  // stored under the chats's unique ID  
  "chats": {  
    "one": {  
      "title": "Historical Tech Pioneers",  
      "lastMessage": "ghopper: Relay malfunction found. Cause: moth.",  
      "timestamp": 1459361875666  
    },  
    "two": { ... },  
    "three": { ... }  
  },  
  
  // Conversation members are easily accessible  
  // and stored by chat conversation ID  
  "members": {  
    // we'll talk about indices like this below  
    "one": {  
      "ghopper": true,  
      "alovelace": true,  
      "eclarke": true  
    },  
    "two": { ... },  
    "three": { ... }  
  },  
  
  // Messages are separate from data we may want to iterate quickly  
  // but still easily paginated and queried, and organized by chat  
  // conversation ID  
  "messages": {  
    "one": {  
      "m1": {  
        "name": "eclarke",  
        "message": "The relay seems to be malfunctioning.",  
        "timestamp": 1459361875337  
      },  
      "m2": { ... },  
      "m3": { ... }  
    },  
    "two": { ... },  
    "three": { ... }  
  }  
}

3.Thao tác trên cơ sở dữ liệu

Để đọc hoặc ghi dữ liệu từ cơ sở dữ liệu, cần phải có lời gọi đến firebase.database.Reference;

// Get a reference to the database service  
var database = firebase.database();

Ghi dữ liệu cơ bản : sử dụng *set()* để lưu dữ liệu vào một tham chiếu được chỉ định, thay thế bất kì dữ liệu đang tồn tại trên đường dẫn đó. Sử dụng *set()* để ghi đè dữ liệu lên bất kì vị trí chỉ định.

function writeUserData(userId, name, email, imageUrl) {  
  firebase.database().ref('users/' + userId).set({  
    username: name,  
    email: email,  
    profile\_picture : imageUrl  
  });  
}

Đọc dữ liệu tại vị trí : sử dụng phương thức *on()* hoặc *once()* của *firebase.database.Reference*

var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');  
starCountRef.on('value', function(snapshot) {  
  updateStarCount(postElement, snapshot.val());  
});

Đọc dữ liệu một lần : trong một số trường hợp, muốn lấy nhanh dữ liệu một lần và không có sử dụng lại thì có thể sử dụng phương thức *once()*

var userId = firebase.auth().currentUser.uid;  
return firebase.database().ref('/users/' + userId).once('value').then(function(snapshot) {  
  var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';  
  // ...  
});

Cập nhật các trường cụ thể: sử dụng phương thức *update()* để ghi vào các nút con cụ thể mà không ghi đè lên các nút khác.

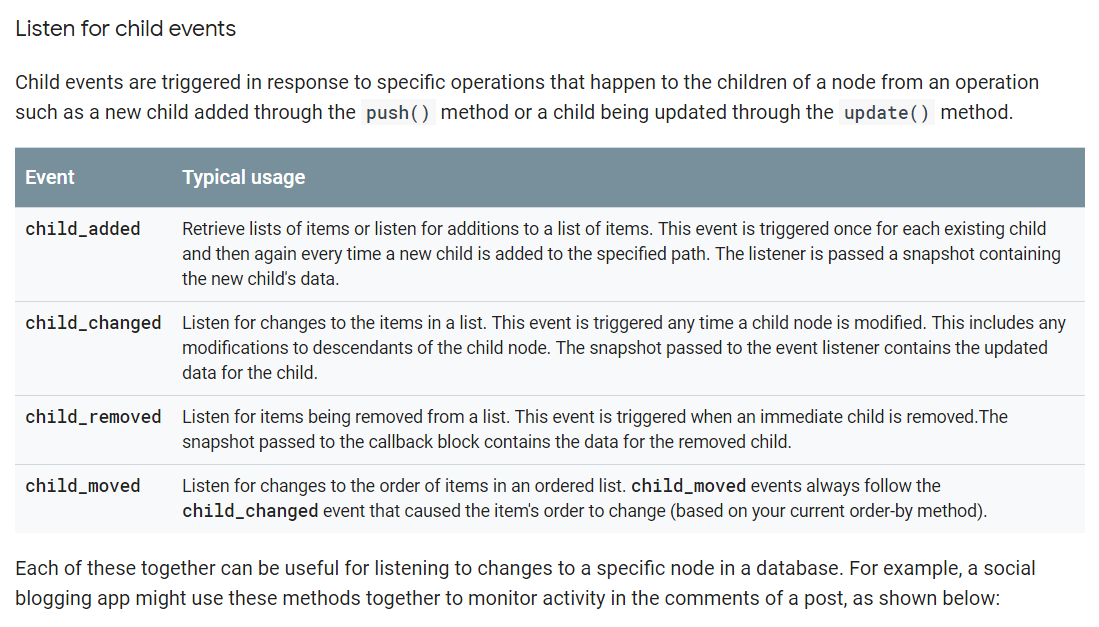
function writeNewPost(uid, username, picture, title, body) {  
  // A post entry.  
  var postData = {  
    author: username,  
    uid: uid,  
    body: body,  
    title: title,  
    starCount: 0,  
    authorPic: picture  
  };  
  
  // Get a key for a new Post.  
  var newPostKey = firebase.database().ref().child('posts').push().key;  
  
  // Write the new post's data simultaneously in the posts list and the user's post list.  
  var updates = {};  
  updates['/posts/' + newPostKey] = postData;  
  updates['/user-posts/' + uid + '/' + newPostKey] = postData;  
  
  return firebase.database().ref().update(updates);  
}

Xóa dữ liệu: Các đơn giản nhất để xóa dữ liệu là gọi *remove()* trên tham chiếu đến vị trí của dữ liệu đó.

* + 1. Làm việc với danh sách dữ liệu trên web

Sử dụng phương thức *push ()* để nối dữ liệu vào danh sách trong các ứng dụng đa người dùng. Phương thức *push ()* tạo ra một khóa duy nhất mỗi lần một con mới được thêm vào tham chiếu Firebase được chỉ định. Bằng cách sử dụng các khóa được tạo tự động này cho từng phần tử mới trong danh sách, một số khách hàng có thể thêm trẻ em vào cùng một vị trí cùng một lúc mà không cần viết xung đột. Khóa duy nhất được tạo bởi push () dựa trên dấu thời gian, vì vậy các mục danh sách được tự động sắp xếp theo thứ tự thời gian.

// Create a new post reference with an auto-generated id  
var newPostRef = postListRef.push();  
newPostRef.set({  
    // ...  
});



* 1. Cloud Firestore

Cloud Firestore là một cloud-hosted, NoSQL database mà các ứng dụng phía client có thể trực tiếp truy cập thông qua native SDKs. Nó lưu dữ liệu theo mô hình dữ liệu NoSQL. Dữ liệu được lưu trữ trong các file tài liệu chứa các trường được ánh xạ vào các giá trị. Các file tài liệu này được lưu trữ trong các tập hợp chúng có thể sử dụng nó để tổ chức dữ liệu và truy vấn dữ liệu. Cloud Firestore hỗ trợ rất nhiều kiểu dữ liệu từ đơn giản như String, Integer hay những kiểu dữ liệu phức tạp như các nested object.

* + 1. So sánh Firebase Realtime DB và Cloud Firestore

Điểm chung: đều lưu trữ dữ liệu người dùng trên nền tảng đám mây theo dạng noSQL, dữ liệu được đồng bộ tới tất cả các máy khách kết nối với nó theo thời gian thực và vẫn khả dụng khi ứng dụng ngoại tuyến. Chúng phù hợp để làm những ứng dụng cần thời gian cập nhật nhanh chóng như là một ứng dụng chat, chia sẻ dữ liệu tức thời khi nhiều người làm cùng một việc.

Khác biệt:

*-Cloud Firestore or Realtime Database (Lựa chọn cơ sở dữ liệu nào):*

Firebae Database: Là cơ sở dữ liệu gốc của FireBase hoạt động hiệu quả, có đỗ trễ thấp cho các ứng dụng yêu cầu các trạng thái

Cloud Firestore Là cơ sở dữ liệu hàng đầu của FireBase mới được đưa ra với việc cải thiện cơ sở dữ liệu trực quan hơn, có các truy vấn phong phú và nhanh hơn so với FireBase Database

*-Data Model ( Mô hình dữ liệu)*

Firebase Database lưu trữ dữ liệu dưới dạng một cây JSON lớn, điều này giúp cho việc lưu trữ dữ liệu một cách đơn giản hơn, nhưng cũng vì thế mà dữ liệu phân cấp phức tạp sẽ khó tổ chức hơn nếu quy mô lớn.

Trong khi đó Cloud Firestore lưu trữ dữ liệu được sắp xếp trong các bộ sưu tập, điều này cũng giúp việc lưu trữ dữ liệu dễ dàng vì nó tương tự như JSON. Dữ liệu phân cấp, phức tạp cũng dễ dàng tổ chức hơn để sắp xếp các tài liệu bên trong đó. Nó cũng yêu cầu ít chuẩn hóa và làm gọn dữ liệu

*Querying (Truy vấn)*

**Firebase Database** truy vấn với một số các phương thức có sẵn được cung cấp ví dụ như orderByChild(), orderByKey(), orderByValue(), limitToFirst(), limitToLast(), startAt(), endAt(), equalTo(). Việc này có hạn chế là việc bạn chỉ có thể lọc hoặc sắp xếp chứ không thể cả hai được. Việc truy vấn cũng theo mặc định luôn trả về toàn bộ các cây con.

**Cloud Firestore** truy vấn với việc đánh chỉ mục và sử dụng cả lọc và sắp xếp với nhau. Nó mạnh mẽ hơn với phương thức where() với 3 tham số như một trường để lọc, một phép toán và một giá trị để so sánh. Ví dụ một số hàm như whereEqualTo(), whereLessThan(), whereGreaterThanOrEqualTo, whereLessThanOrEqualTo, ... làm cho việc truy vấn trở nên hiệu quả hơn và dễ dàng hơn. Truy vấn mặc định được đánh chỉ số theo kết quả trả về. Tuy nhiên Cloud Firesotre lại không hỗ trợ các truy vấn sau: Mỗi truy vấn chỉ được thực hiện trên một bộ sưu tập nhất định, truy vấn các thành viên của mảng riêng lẻ (bạn có thể sử dụng các kỹ thuật sử dụng cho List, ArrayList, Set để truy vấn), truy vấn logic OR bạn nên tạo truy vấn riêng cho từng truy vấn và hợp nhất với nhau, truy vấn có mệnh đề "!=" (trường hợp này bạn nên tạo truy vấn cho từng trường hợp lớn hơn hoặc nhỏ hơn của truy vấn đó).

*Writes and Transactions*

Firebase Database thực hiện write và transaction một cách cơ bản như: ghi dữ liệu như một thao tác đơn lẻ, còn transaction trong SDKs thì sử yêu cầu phải có sự kiện trả về hoàn tất.

Cloud Firestore thực hiện write và transaction một cách nguyên bản: Hoạt động hàng loạt và hoàn thành chúng một cách nguyên bản, transaction tự động lặp lại cho đến khi nó hoàn tất.

*Reliability and performance ( Sự ổn định và hiệu năng)*

Về khoản này thì FireBase Database làm một sản phẩm đã hoàn thành: Sự ổn định của nó đã được kiểm nghiệm và độ trễ của nó cũng là thấp nhất, do đó nó là sự lựa chọn tốt nhất cho những ứng dụng cần thiết việc đồng bộ hóa trạng thái thường xuyên.

Cloud Firestore hiện tại đang trong giai đoạn thử nghiệm với phiên bản beta: Sự ổn định của bản thử nghiệm không phải lúc nào cũng được như bản đã hoàn thành, dữ liệu của bạn được đặt ở nhiều nơi đảm bảo tính mở rộng và độ tin cậy cao, khi nó vượt qua giai đoạn thử nghiệm sẽ hứa hẹn sự ổn định và hiệu năng tốt hơn so với FireBase Database.

*Scalability ( Khả năng mở rộng)*

FireBase Database nếu muốn mở rộng phải yêu cầu sự phân tán, có thể chia sẻ cho khoảng 100.000 kết nối và 1000 lần ghi / giây trong một database. Mở rộng thì phải yêu cầu phân tán dữ liệu của bạn trên nhiều cơ sở dữ liệu.

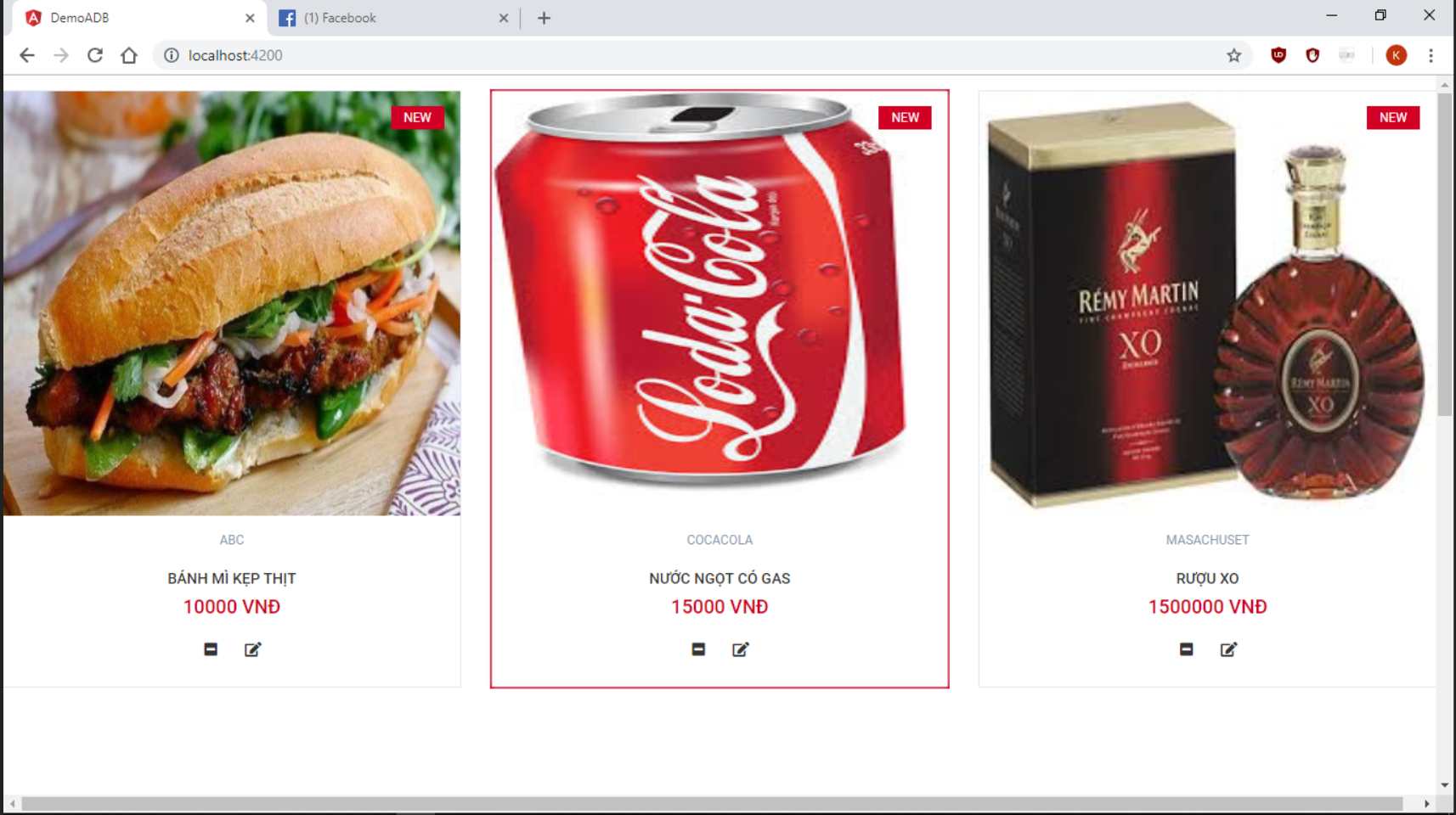
Cloud Firestore: Việc mở rộng này là hoàn toàn tự động, vậy nên bạn sẽ không cần phải quan tâm đến việc dữ liệu của mình lưu trữ theo nhiều phiên bản.

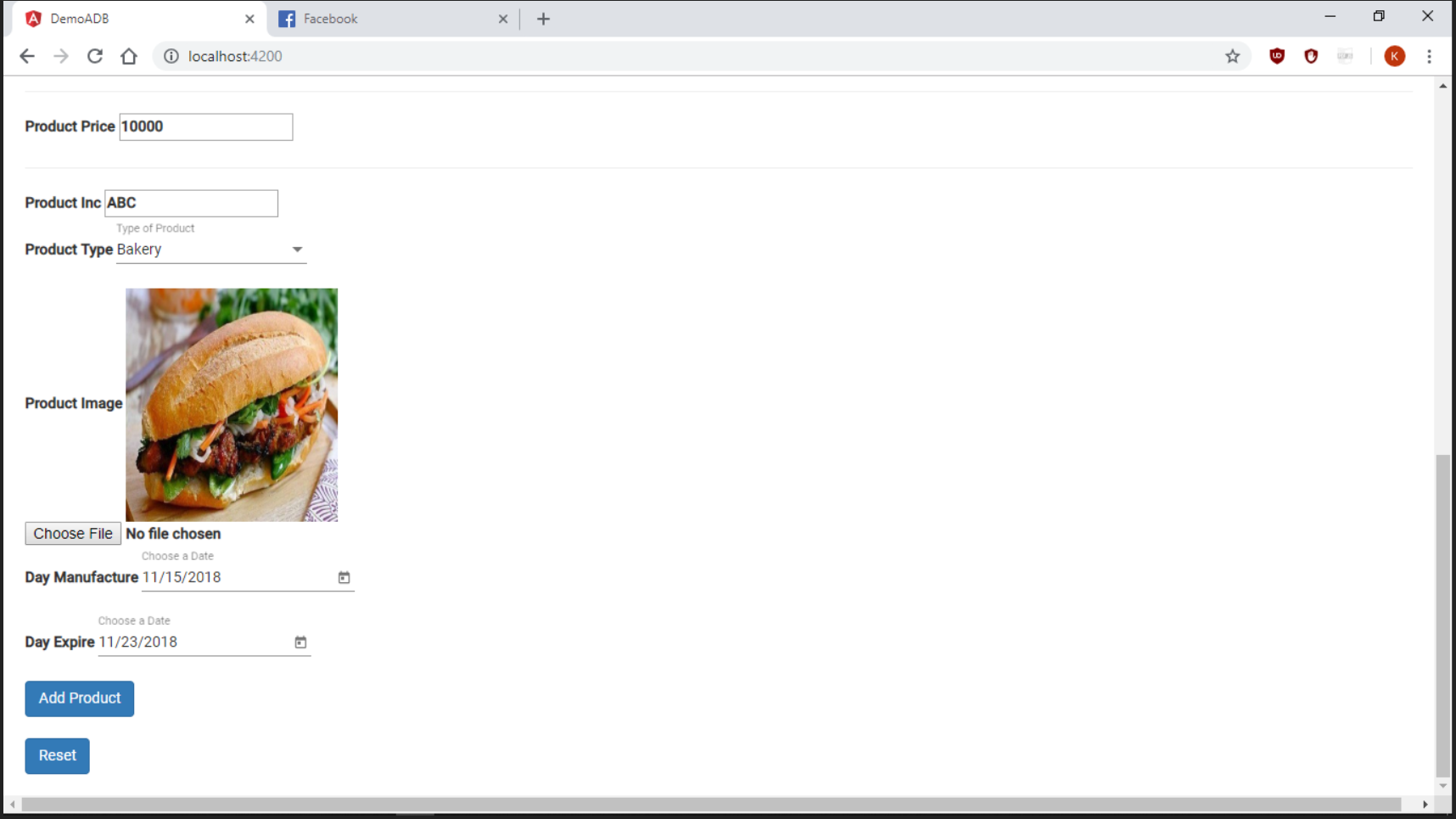
*Security (Bảo mật)*

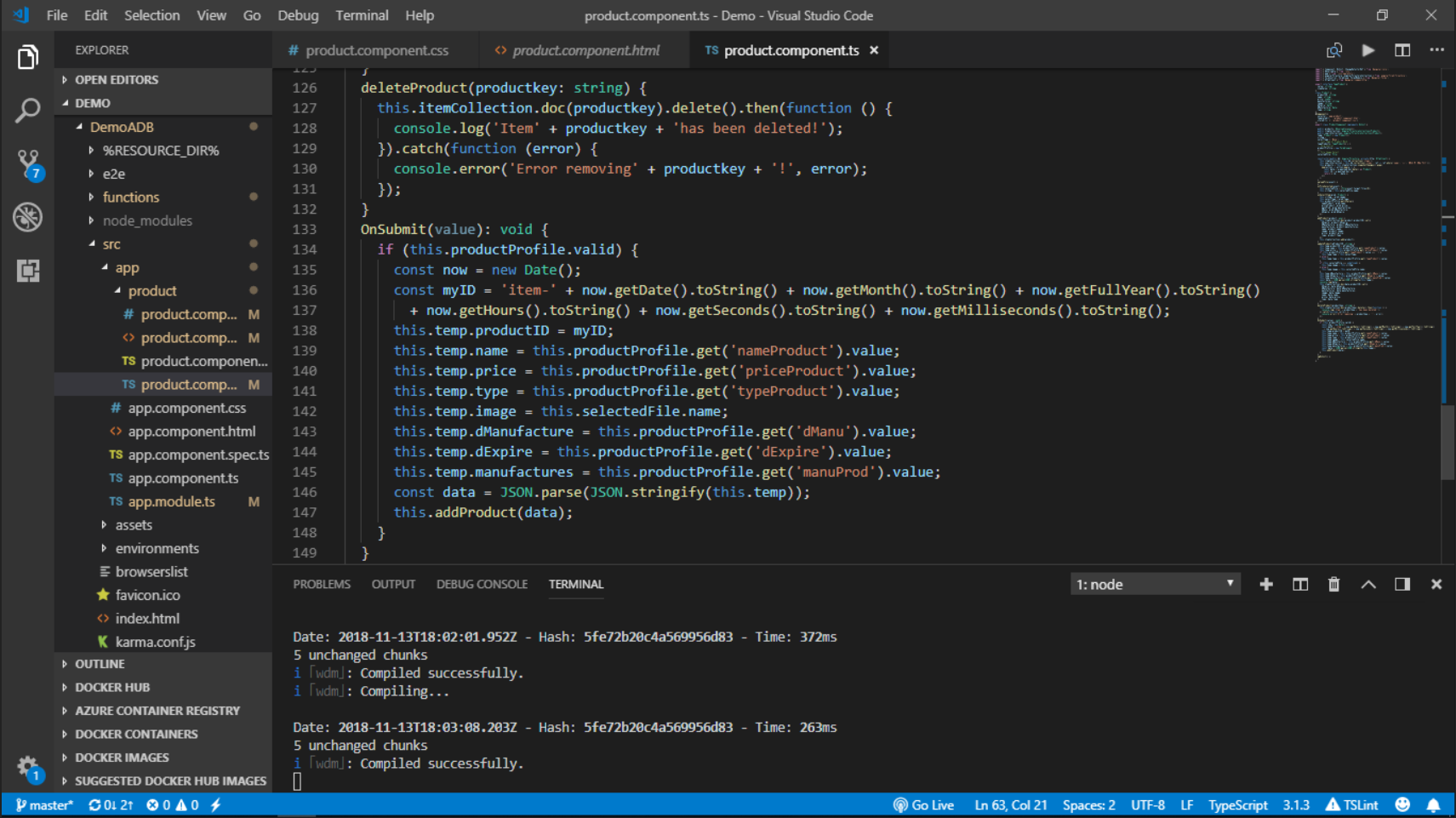
FireBase Database có các quy tắc xác thực xếp tầng riêng: FireBase Database Rules là quy tắc bảo mật duy nhất bao gồm các quy tắc đọc và ghi, bạn cần xác thực dữ liệu riêng biệt bằng các quy tắc xác thực của FireBase.

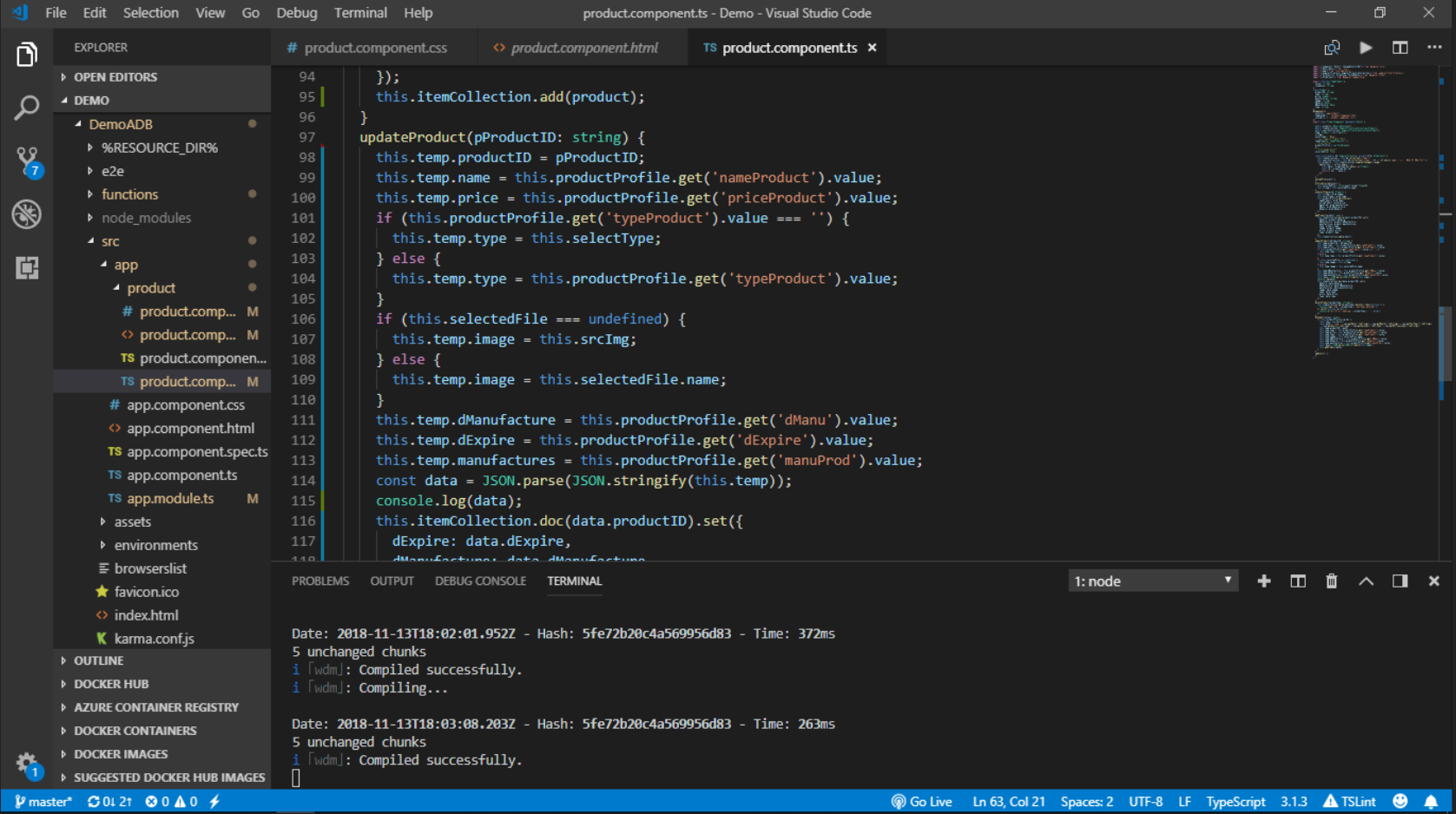
Cloud Firestore thì bảo mật đơn giản và mạnh mẽ hơn cho SDK di động, web và server: SDK trên thiết bị di động và web sử dụng các quy tắc bảo mật của Cloud Firestore. SDK máy chủ sử dụng quản lý danh tính và truy cập ( Identity and Access Management - IAM), các quy tắc không xếp tầng trừ khi bạn sử dụng ký tự đại diện, xác thực dữ liệu diễn ra một cách tự động, quy tắc có thể hạn chế việc truy vấn nếu như người dùng truy vấn vào dữ liệu không có quyền truy cập thì toàn bộ truy vấn sẽ không thành công.

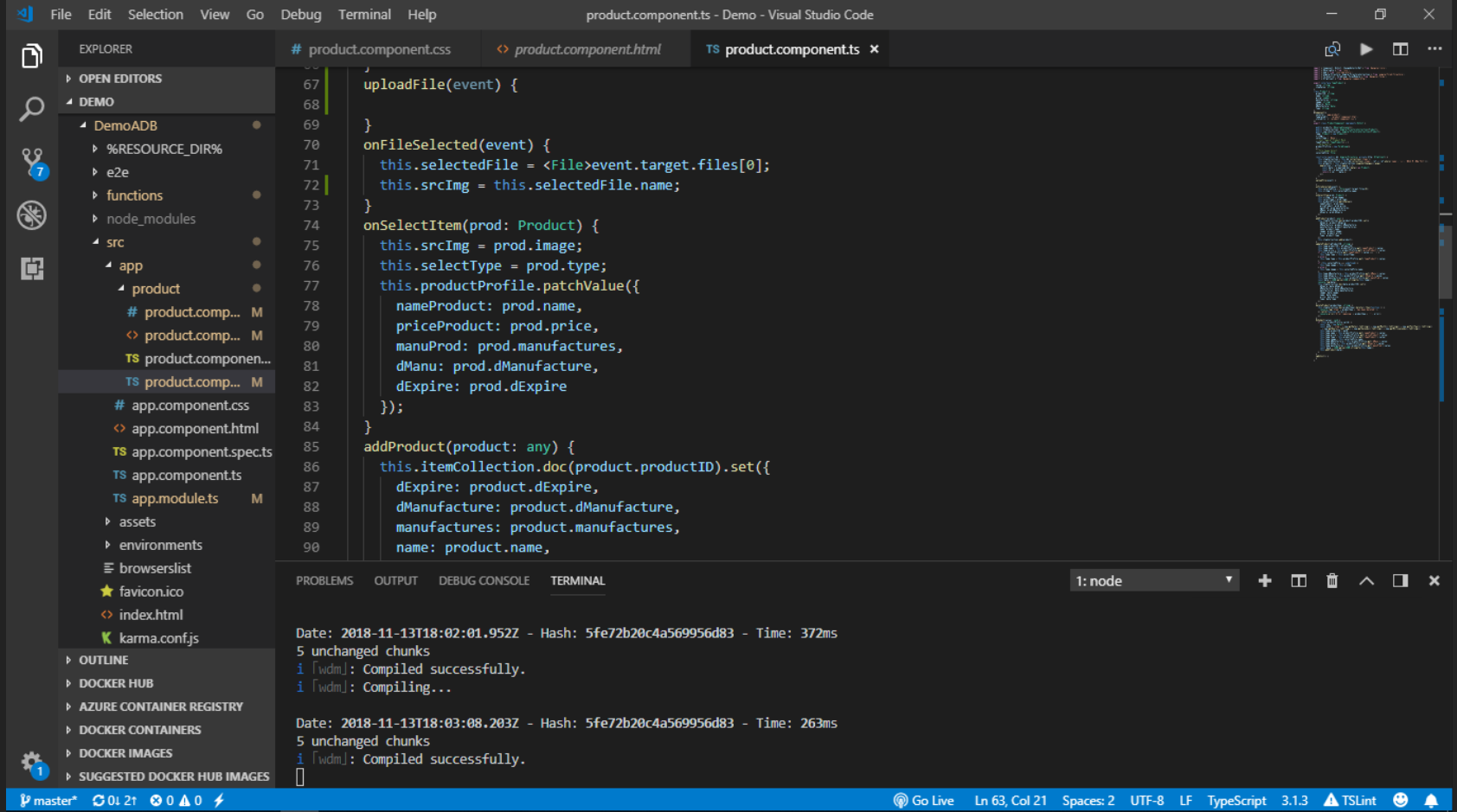
1. Demo

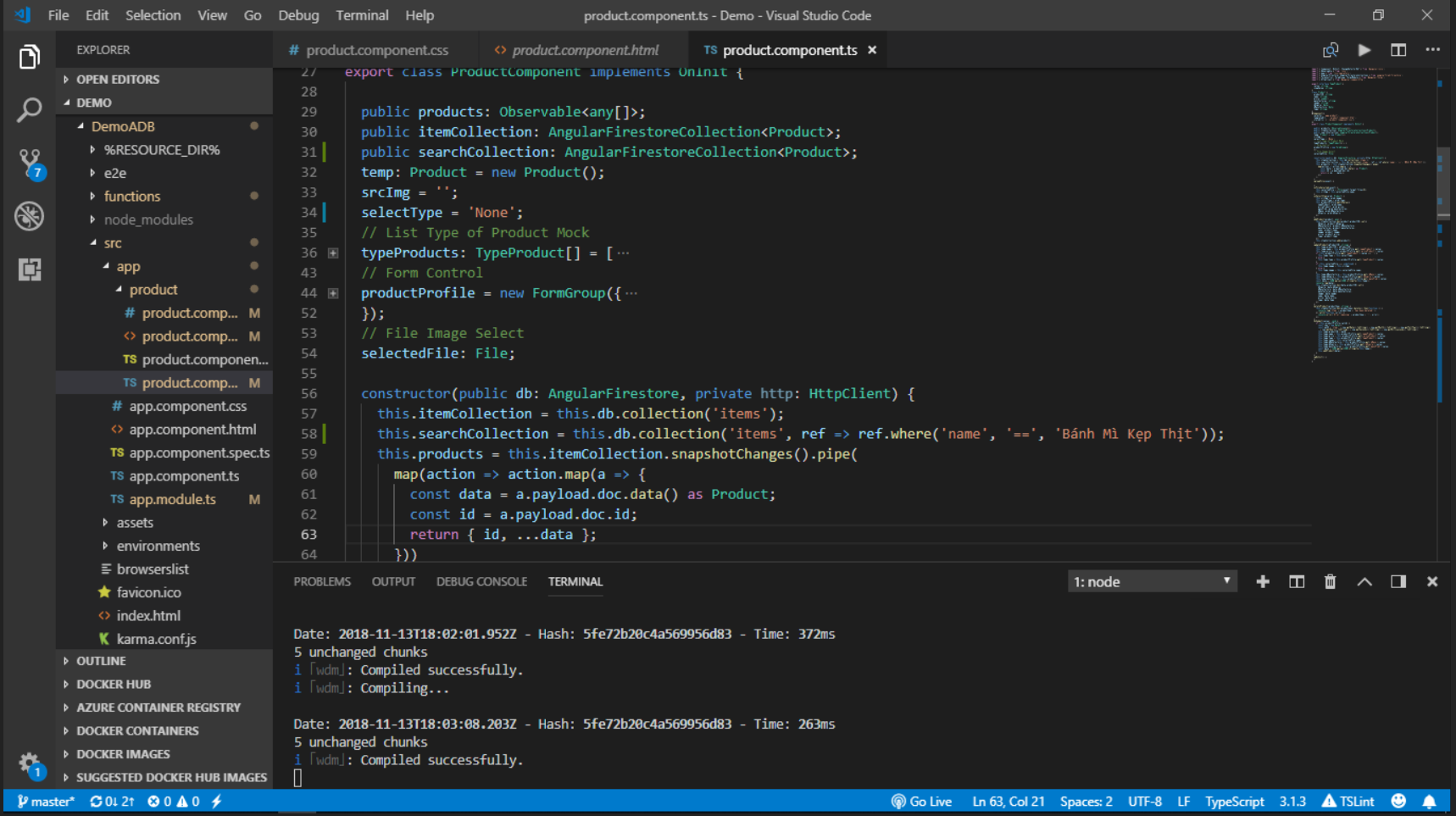














1. Tài liệu tham khảo

<https://firebase.google.com>

<https://viblo.asia/p/gioi-thieu-ve-nosql-database-djeZ1a9jZWz>

<https://hackernoon.com/introduction-to-firebase-218a23186cd7?fbclid=IwAR3vslo3kDmbccZYRwgB9Vfq4WTcDfxqmf76ELOsBlc7huEC9_D1jsZ6m-w>

<https://medium.baqend.com/real-time-databases-explained-why-meteor-rethinkdb-parse-and-firebase-dont-scale-822ff87d2f87>