Firebase

@Chalach.mo

Develop & test your app

- Realtime Database
- Cloud Firestore
- Authentication
- Cloud Storage
- Test Lab
- Performance Monitoring
- Crashlytics
- Crash Reporting
- Cloud Functions
- Hosting

Grow & engage your audience

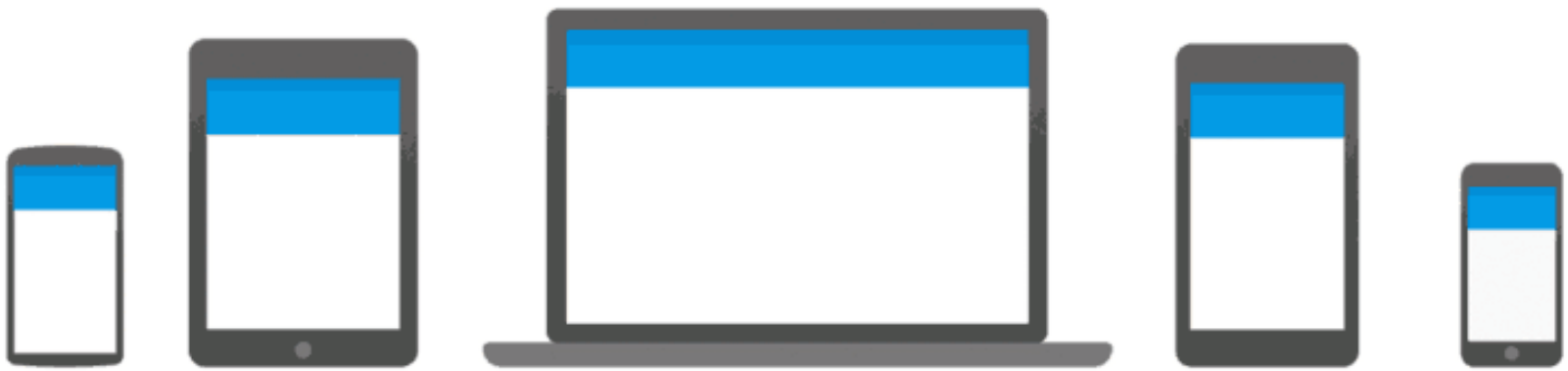
- Analytics
- Dynamic Links
- Invites
- AdWords
- Cloud Messaging
- Remote Config
- Predictions
- App Indexing
- AdMob

Firebase

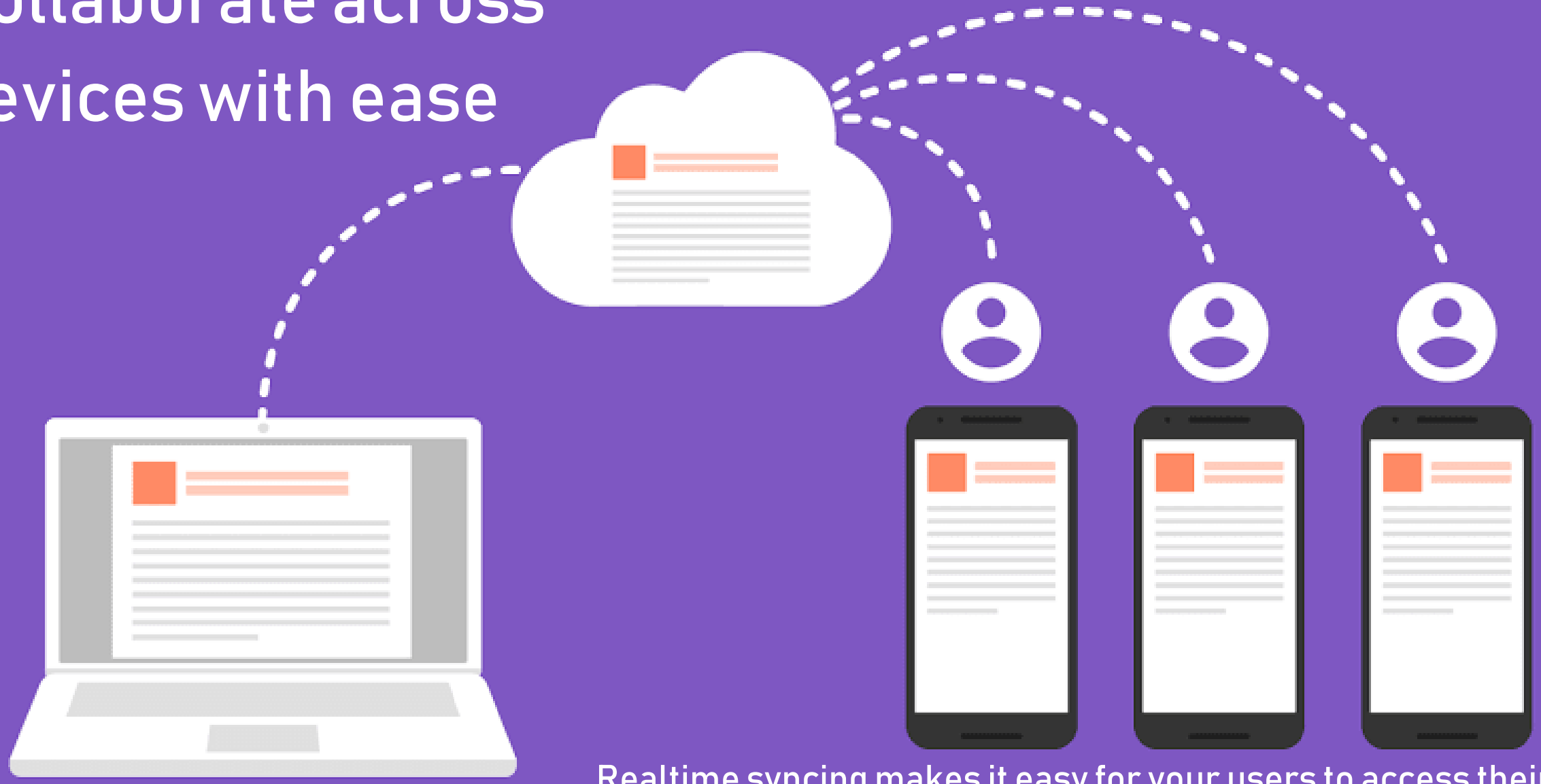
Realtime Database

Firebase
Realtime Database

- The Firebase Realtime Database is a cloud-hosted database.
- Data is stored as JSON and synchronized in realtime to every connected client.
- When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.



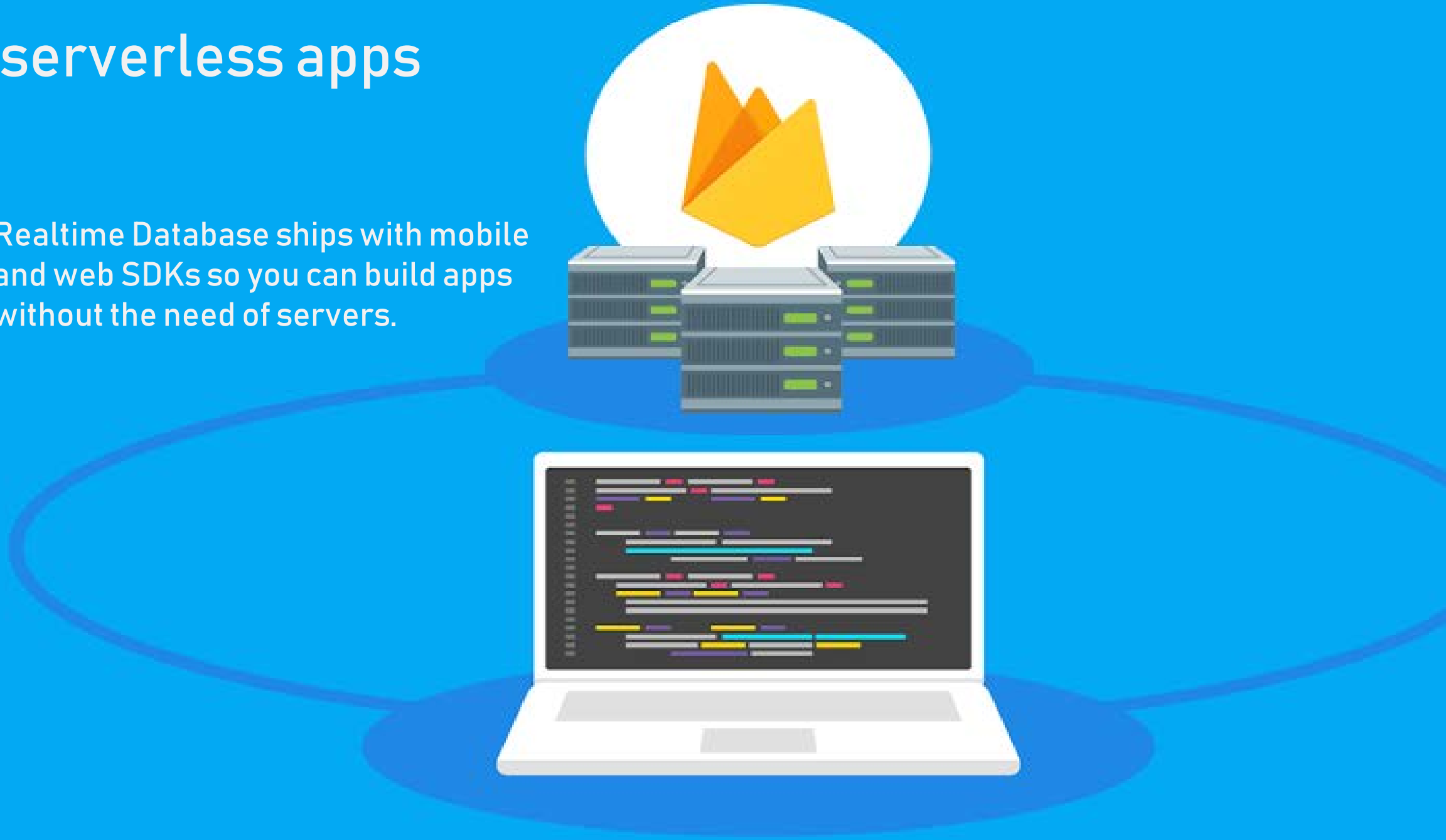
Collaborate across devices with ease



Realtime syncing makes it easy for your users to access their data from any device: web or mobile, and it helps your users collaborate with one another.

Build serverless apps

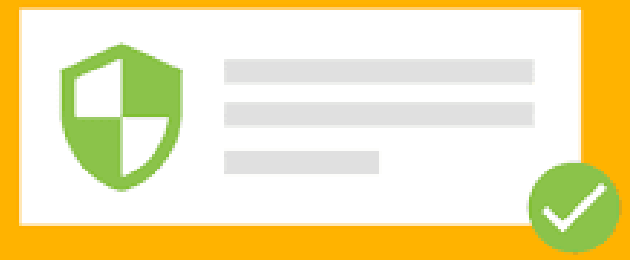
Realtime Database ships with mobile and web SDKs so you can build apps without the need of servers.



Optimized for offline use

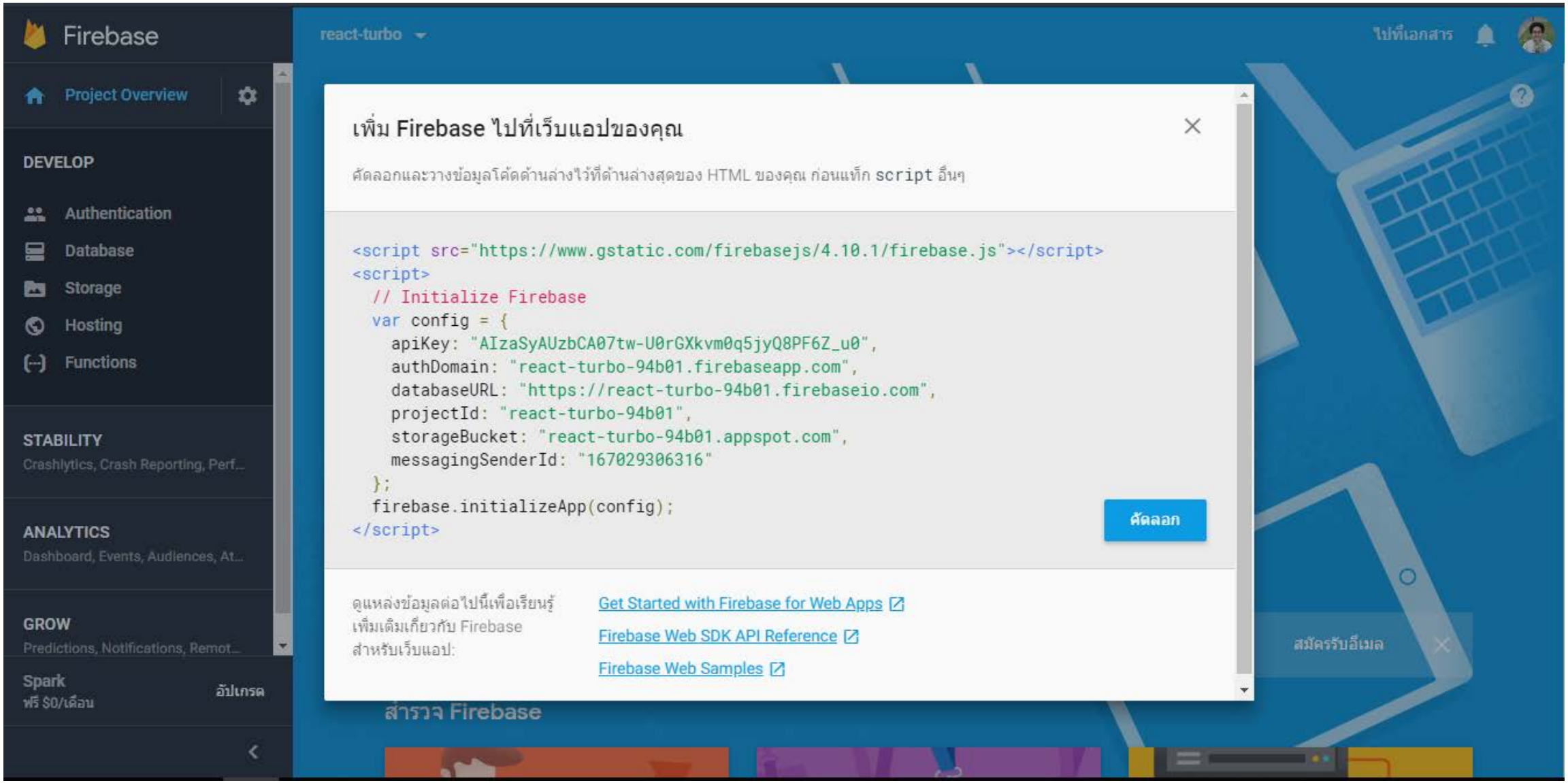
When your users go offline, the Realtime Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

Strong user-based security



The Realtime Database integrates with Firebase Authentication to provide simple and intuitive authentication for developers.

Configuration Firebase



Firebase

Realtime Database

Read Data

Example Read with React

```
react-turbo-94b01
  messages
    -L6Px39KEk2MaAxaTKR6
      message: "11111111111111
    -L6Px3K-9GMxuM5Mxuln
      message: "react te
    -L6Px3Qc2-9WGIEjx8ia
      message: "test dai loey
    -L6Px3Z8QjgbG5hpVA8K
      message: "4
    -L6Px3gtRUYyRFLYR-J2
      message: "5
    -L6RDDP691xo4RPkmrgi
      message: "aaaaaaaaaaaaaaaaaaa
    -L6aHX-Q_PinkSYPNFzt
      message: "asdsac
```

1

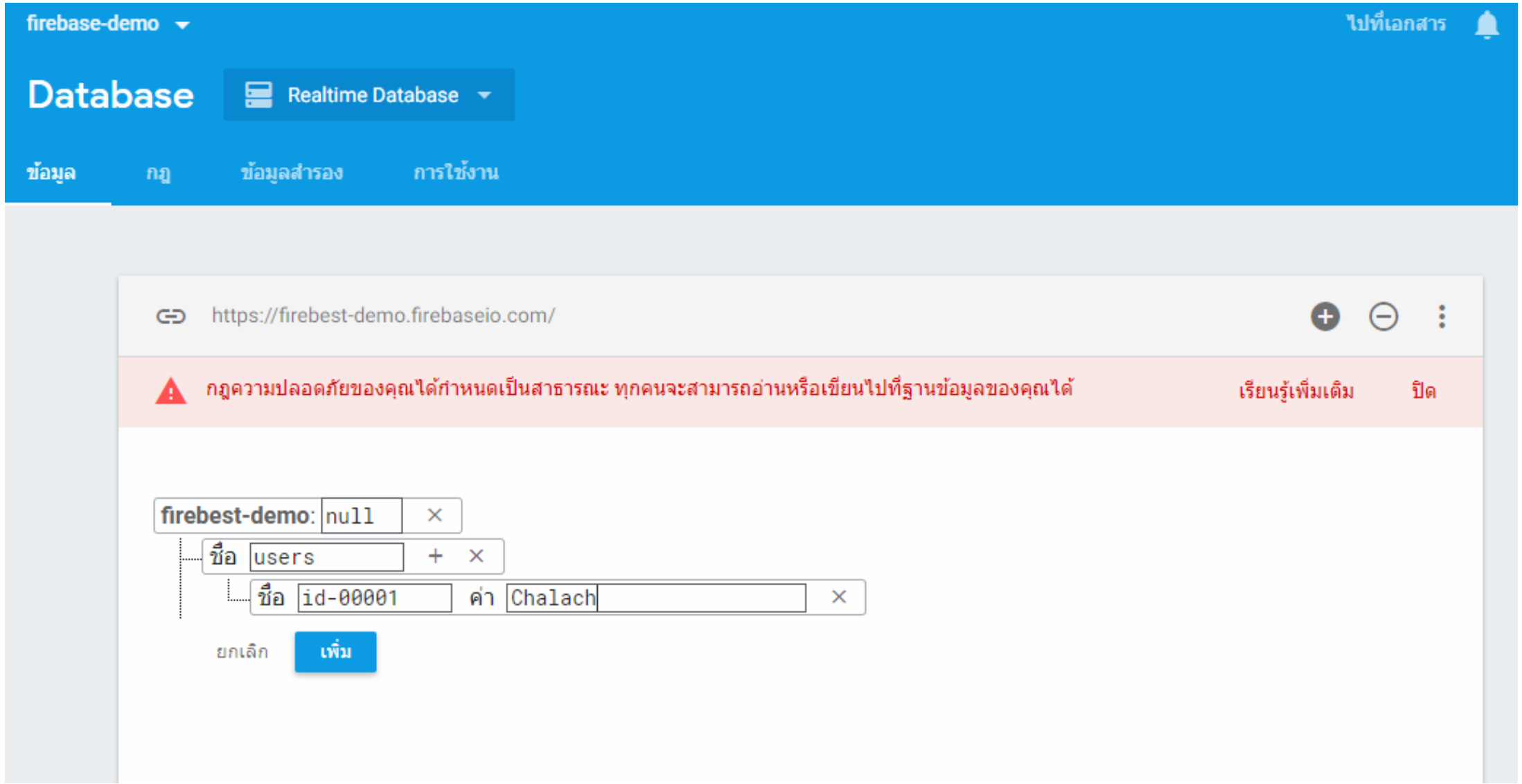
```
let dbCon = firebase.database().ref('messages');

dbCon.on('value', snapshot => {
  console.log(snapshot.val());
});
```

Firebase

Realtime Database

Saving Data

Manual Saving(Add/Insert/Create) Data



Way To Saving(Add/Insert/Create) Data [Coding]

- This document covers the four methods for writing data to your Firebase Realtime Database: set, update, push, and transactions support.

4 Way To Saving

set	Write or replace data to a **defined path**, like `messages/users/<username>`
update	Update some of the keys for a defined path without replacing all of the data
push	**Add to a list of data** in the database. Every time you push a new node onto a list, your database generates a unique key, like `messages/users/<unique-user-id>/<username>`
transaction	Use transactions when working with complex data that could be corrupted by concurrent updates

Example Saving with React

```
react-turbo-94b01
  messages
    -L6Px39KEk2MaAxaTKR6
      message: "11111111111111
    -L6Px3K-9GMxuM5Mxuln
      message: "react te
    -L6Px3Qc2-9WGIEjx8ia
      message: "test dai loey
    -L6Px3Z8QjgbG5hpVA8K
      message: "4
    -L6Px3gtRUYyRFLYR-J2
      message: "5
    -L6RDDP691xo4RPkmrgi
      message: "aaaaaaaaaaaaaaaaaaa
    -L6aHX-Q_PinkSYPNFzt
      message: "asdsac
```

```
let dbCon = firebase.database().ref('/messages' + messageId);
let dbConNormal = firebase.database().ref('/messages');
```

1

```
dbCon.set({
message: xxx
});
```

2

```
dbConNormal.push({
message: xxx
});
```

3

```
var obj = {message: xxx};
dbConNormal.child(messageId).update(obj);
```

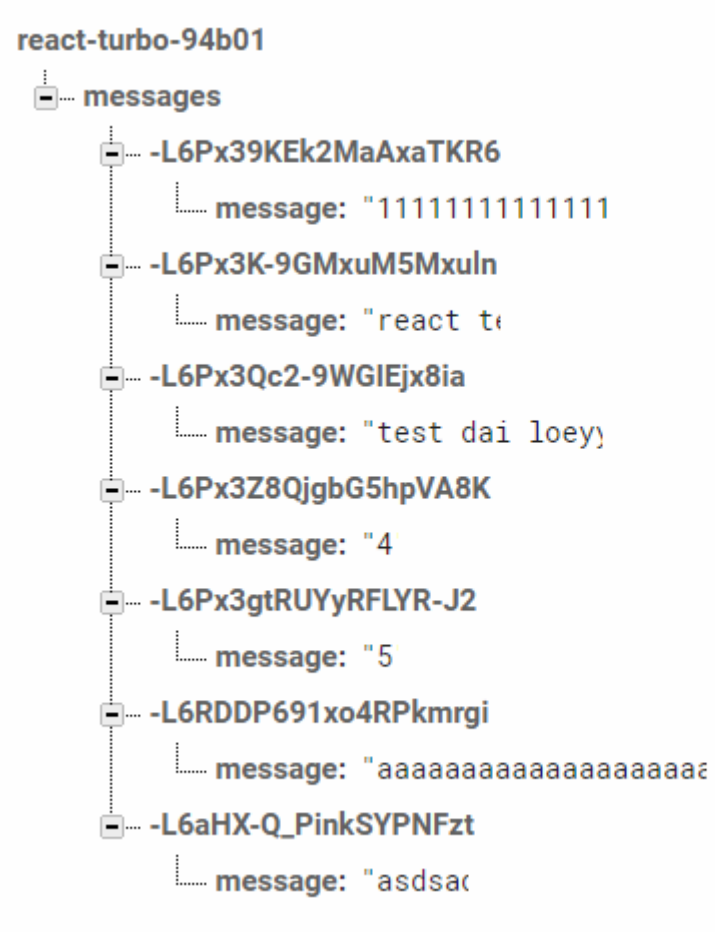
4

Firebase

Realtime Database

Update Data

Example Update with React



1

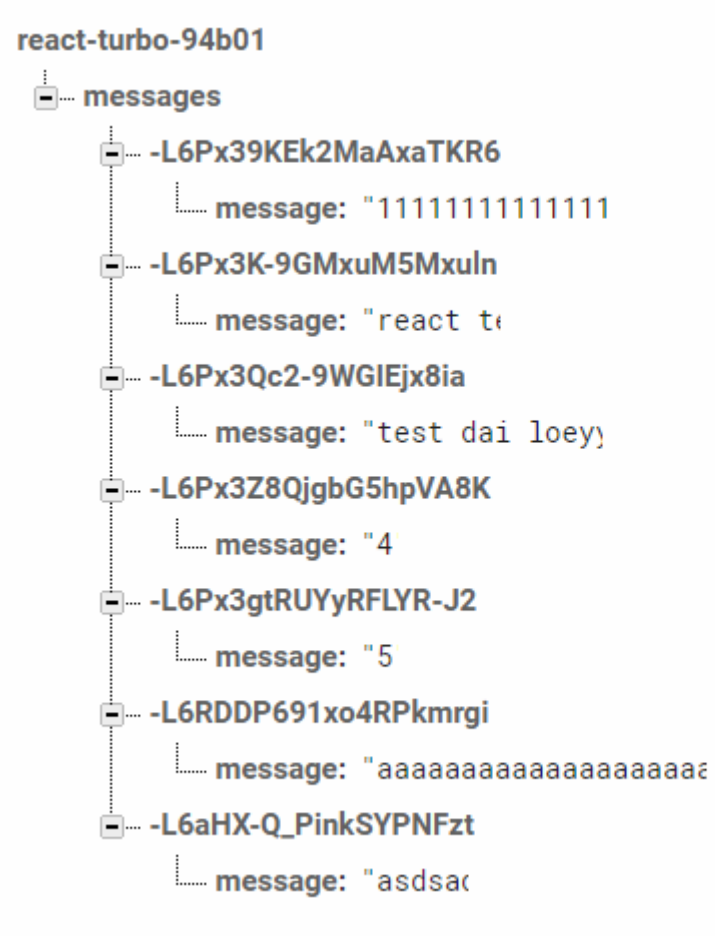
```
let dbCon = firebase.database().ref('/messages');

var obj = {message: xxx};
dbCon.child(messageId).update(obj);
```

Firebase

Realtime Database

Delete Data

Example Delete with React




```
let dbCon = firebase.database().ref('/messages');
```

1

```
dbCon.child(messageId).remove();
```

Firebase

Realtime Database

Sorting Data

Sort Data

- You can use the Realtime Database **Query** class to retrieve data sorted by key, by value, or by value of a child. You can also filter the sorted result to a specific number of results or a range of keys or values.

Method	Usage
`orderByChild()`	Order results by the value of a specified child key or nested child path.
`orderByKey()`	Order results by child keys.
`orderByValue()`	Order results by child values.

```
let dbCon = firebased.database().ref('messages' + messageId).orderByChild('xxx');

let dbCon = firebased.database().ref('messages' + messageId).orderByKey();

let dbCon = firebased.database().ref('messages' + messageId).orderByValue();
```

Firebase

Filtering Data

Realtime Database

Filter Data

- To filter data, you can combine any of the limit or range methods with an order-by method when constructing a query.

Method	Usage
`limitToFirst()`	Sets the maximum number of items to return from the beginning of the ordered list of results.
`limitToLast()`	Sets the maximum number of items to return from the end of the ordered list of results.
`startAt()`	Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
`endAt()`	Return items less than or equal to the specified key or value, depending on the order-by method chosen.
`equalTo()`	Return items equal to the specified key or value, depending on the order-by method chosen.

Different Between SQL and Firebase Realtime Database

SQL Database (Relational Database)

ID	First_Name	Title	Team
10011	Debra	Programmer	Eng
10018	Yolanda	Programmer	Eng
10019	Glen	Product Designer	Mkt
10028	Casey	Account Exec	Sal
10049	Tang	Support Tech	Sup
10051	Serge	UX Designer	Eng
10059	Maria	Sales Director	Sal

Create Table

```sql
CREATE TABLE Customers (
    Id INT(5) NOT NULL AUTO_INCREMENT,
    FirstName VARCHAR(100) NOT NULL,
    Birthday DATE,
    Location VARCHAR(250),
    PRIMARY KEY(Id)
);
```

Insert Data

```
INSERT INTO Customers
   (FirstName, Birthday, Location)
VALUES
   ("David", Now(), "SF");
```

Insert Data(2)

```
INSERT INTO Customers
   (FirstName, LastName, Birthday
   ,Location)
VALUES
   ("David", "East", NOW(), "SF");
```

ERROR: LastName does not exist!

Alter Table

ID	First_Name	Title	Team
10011	Debra	Programmer	Eng
10018	Yolanda	Programmer	Eng
10019	Glen	Product Designer	Mkt
10028	Casey	Account Exec	Sal
10049	Tang	Support Tech	Sup
10051	Serge	UX Designer	Eng
10059	Maria	Sales Director	Sal

```
ALTER TABLE Customers
ADD COLUMN LastName
VARCHAR(100) NOT NULL
```

Alter Table

```
ALTER TABLE Customers
ADD COLUMN LastName
VARCHAR(100)
```

Alter Table

```
ALTER TABLE Customers
MODIY LastName VARCHAR(100)
NOT NULL
```

Firebase Database (NoSQL Database)

```
{
  "customers": {
    "customer_one": {
      "firstName": "David",
      "birthday": 1475189812156,
      "location": "SF"
    }
  }
}
```

```
customers.child("customer_one").child("lastname").setValue("New Lastname")
```

Firebase
Realtime Database

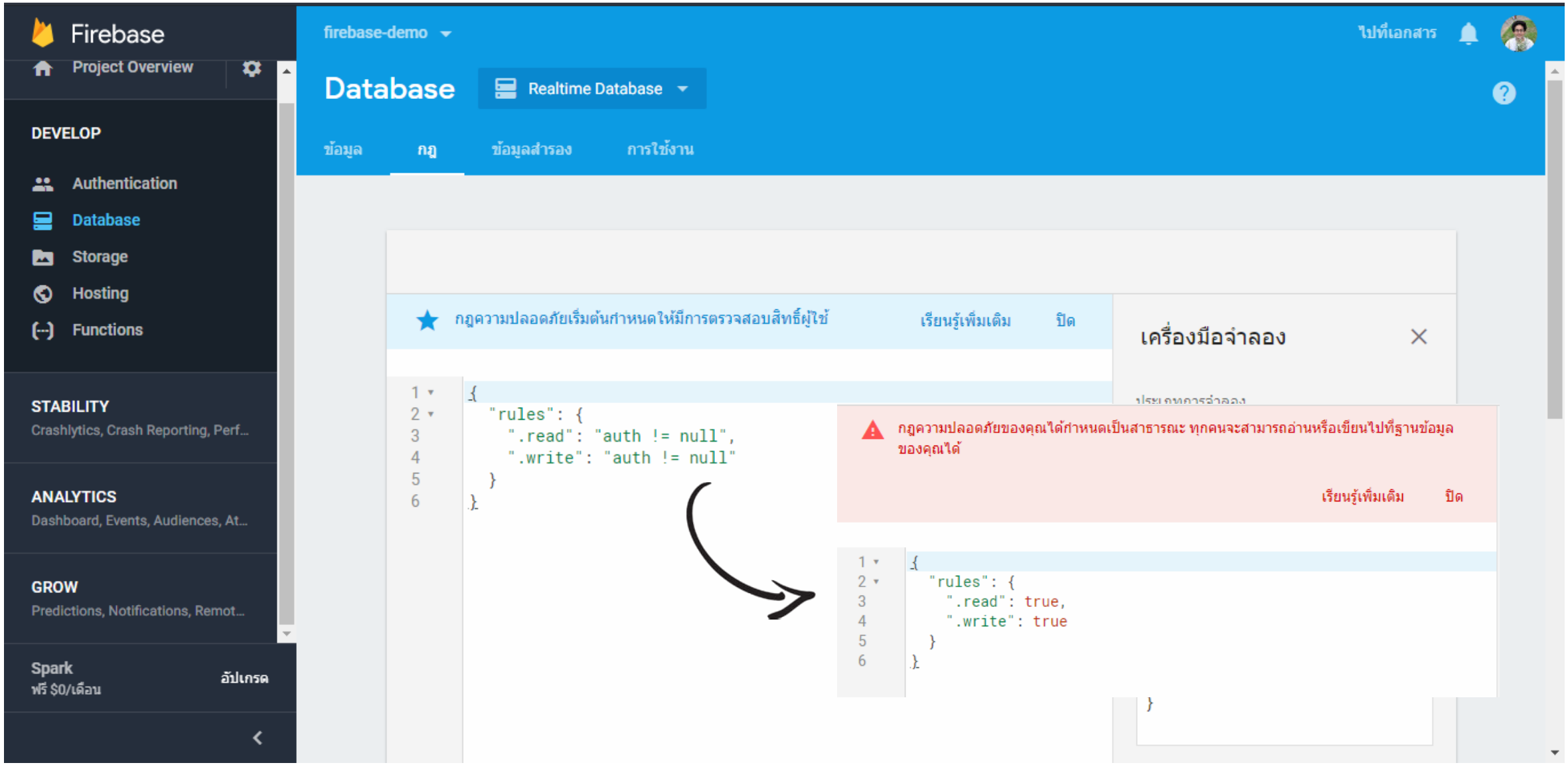
Understand Firebase Realtime Database Rules

- Firebase Realtime Database Rules determine who has read and write access to your database, how your data is structured, and what indexes exist. These rules live on the Firebase servers and are enforced automatically at all times. Every read and write request will only be completed if your rules allow it. By default, your rules are set to allow only authenticated users full read and write access to your database. This is to protect your database from abuse until you have time to customize your rules or set up authentication.

- Firebase Database Rules have a JavaScript-like syntax and come in four types:

Rule Types	
.read	Describes if and when data is allowed to be read by users.
.write	Describes if and when data is allowed to be written.
.validate	Defines what a correctly formatted value will look like, whether it has child attributes, and the data type.
.indexOn	Specifies a child to index to support ordering and querying.

Rules .read .write



Rules .validate

```
{
  "rules": {
    "customers": {
      "$uid": {
        ".validate": "newData.child('firstName').isString() &&
                      newData.child('birthday').isNumber() &&
                      newData.child('location').isString()"
    }
  }
}
```

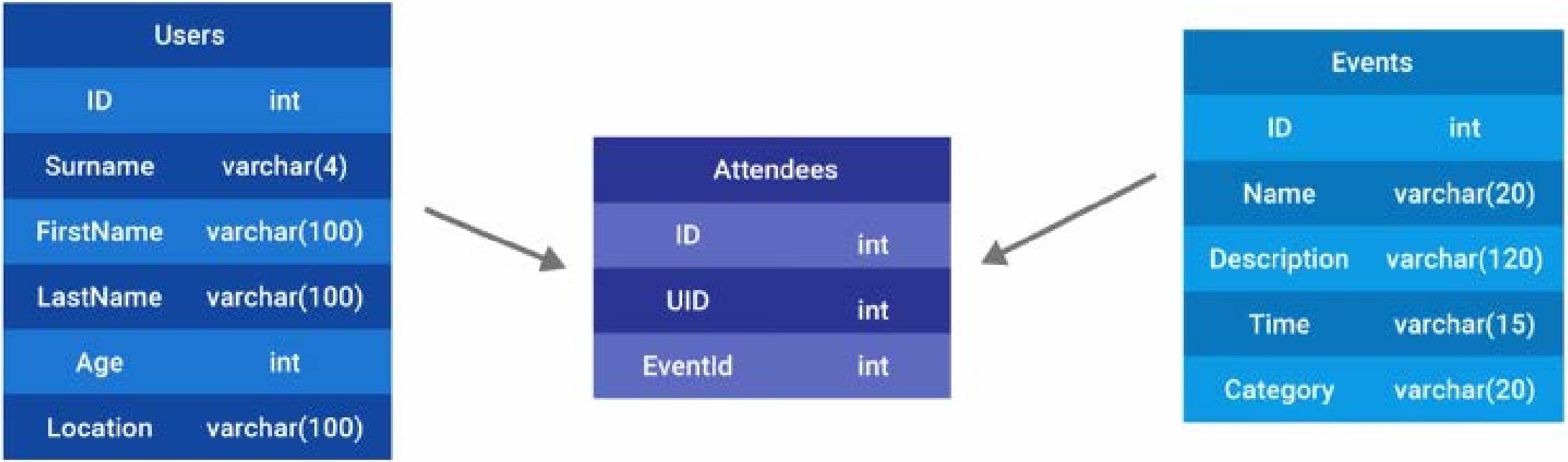
SQL Database

Convert TO

Firebase

Realtime Database

SQL Structure Example



NoSQL Structure Example

```
{
   "users": {
      "1": {
        "name": "David"
      },
      "9": {
        "name": "Alice"
      }
   },
   "events": {
      "fm": {
         "name": "Firebase Meetup",
         "date": 983275235320,
         "attendees": {
            "1": "David",
            "9": "Alice"
         }
      }
   }
}
```

```
{
   "users": {
      "1": {
        "name": "David"
      },
      "9": {
        "name": "Alice"
      }
   },
   "events": {
      "fm": {
         "name": "Firebase Meetup",
         "date": 983275235320,
      }
   },
   "eventAttendees": {
      "fm": {
         "1": "David",
         "9": "Alice"
      }
   }
}
```

SELECT

```
SELECT *
FROM Events
WHERE Name == "Firebase Meetup";
```

```
const db = firebase.database();
const eventsRef = db.child('events');

eventsRef.orderFunction().queryFunction();

eventsRef.orderByKey().limitToFirst(10);
```

SELECT (2)

```sql
SELECT event.Name as EventName
      ,event.Date as EventDate
      ,user.Name as AttendeeName
  FROM Events as event
  INNER JOIN Attendees as a
    ON e.Id === a.EventId
  INNER JOIN Users as user
    ON u.UId = a.UId
  WHERE e.Id == 4;
```

```javascript
const db = firebase.database();
const events = db.child('events/fm');
const attendees = db.child('eventAttendees/fm');

events.on('value', snap => {
   // render data to HTML
});

attendees.on('child_added', snap => {
   // append attendees to list
});
```

SELECT (3)

```
const db = firebase.database();
const events = db.child('events');
const query = events
                .orderByChild('name')
                .equalTo('Firebase Meetup')
                .limitToFirst(1);

query.on('value', snap => {
   // render data to HTML
});
```

Firebase

Realtime Database

Querying Data

SQL Database

uid	name	email	age	location
1	Jirawat	jirawatee@gmail.com	18	Surat Thani
2	FirebaseThailand	firebasethailand@gmail.com	32	Bangkok

Firebase Database

```
{
  "users" : {
    "1" : {
        "age" : 18,
        "email" : "jirawatee@gmail.com",
        "location" : "SuratThani",
        "name" : "Jirawat"
    },
    "2" : {
        "age" : 32,
        "email" : "firebasethailand@gmail.com",
        "location" : "Bangkok",
        "name" : "FirebaseThailand"
    }
  }
}
```

SQL Query

```
# 1. Select a user by UID
    • SELECT * FROM Users WHERE uid = 1
# 2. Find a user by email address
    • SELECT * FROM users WHERE email = 'firebasethailand@gmail.com'
# 3. Limit to 10 users
    • SELECT * FROM users LIMIT 10
# 4. Get all users names that start with 'F'
    • SELECT * FROM users WHERE name LIKE 'F%'
# 5. Get all users who are age less than 25
    • SELECT * FROM user WHERE age < 25
# 6. Get all users who are age greater than 25
    • SELECT * FROM users WHERE age >= 25;
# 7. Get all users who are age between 18 and 32
    • SELECT * FROM users WHERE age >= 18 && age <= 32;
# 8. Get all users who are 32 and live in Bangkok
    • SELECT * FROM users WHERE age = 32 && Location = 'Bangkok';
```

Firebase Query

```
const mRootRef = firebase.database().ref('...');
```

```
# 1. Select a user by UID
    • mRootRef.child(“users”).child(“1”);
# 2. Find a user by email address
    • mRootRef.child(“users”).orderByChild(‘email’).equalTo(“firebasethailand@gmail.com”);
# 3. Limit to 10 users
    • mRootRef.child(“users”).orderByKey().limitToFirst(10);
# 4. Get all users names that start with 'F'
    • mRootRef.child(“users”).orderByChild(“name”).startAt(“F”);
# 5. Get all users who are age less than 25
    • mRootRef.child(“users”).orderByChild("age").endAt(25);
# 6. Get all users who are age greater than 25
    • mRootRef.child(“users”).orderByChild("age").startAt(25);
# 7. Get all users who are age between 18 and 32
    • mRootRef.child(“users”).orderByChild("age").startAt(18).endAt(32);
# 8. Get all users who are 32 and live in Bangkok
    • mRootRef.child(“users”).orderByChild("age").equalTo(32)
                              .orderByChild("location").equalTo("Bangkok");
```

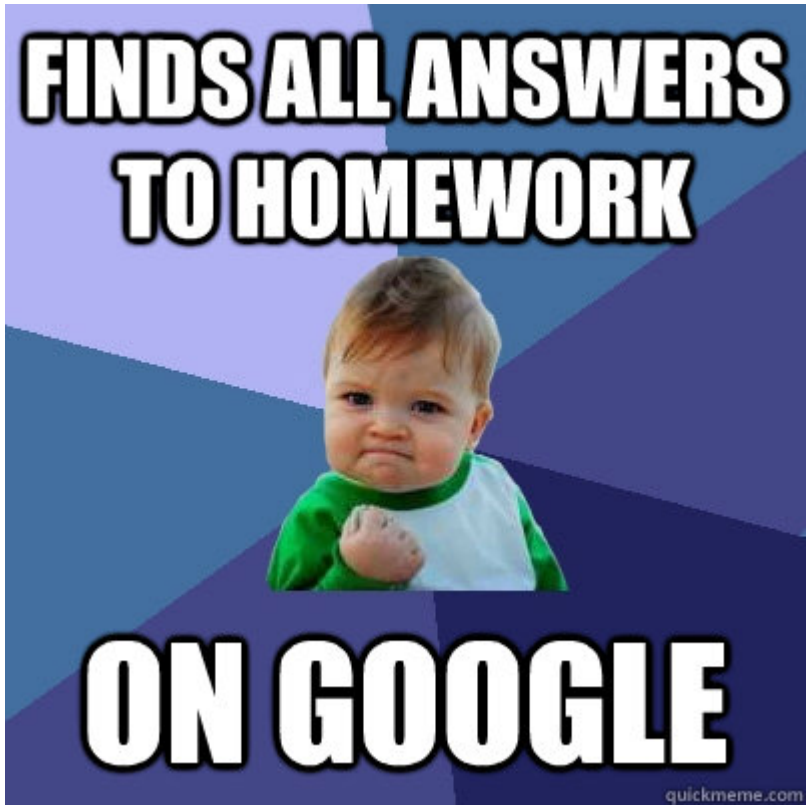
Firebase

Realtime Database

Comment Workshop

Homework

1. -



Reference

- https://firebase.google.com/docs
- https://developers.ascendcorp.com/รู้จัก-firebase-realtime-database-ตั้งแต่-zero-จนเป็น-hero-5d09210e6fd6ข
- https://developers.ascendcorp.com/มาทำความเข้าใจกับ-sql-database-และ-firebase-database-กันเถอะ-4aed4a19e339
- https://appdividend.com/2017/07/22/react-firebase-tutorial/
- **React: Functional Web Development with React and Redux** 1st Edition, Kindle Edition