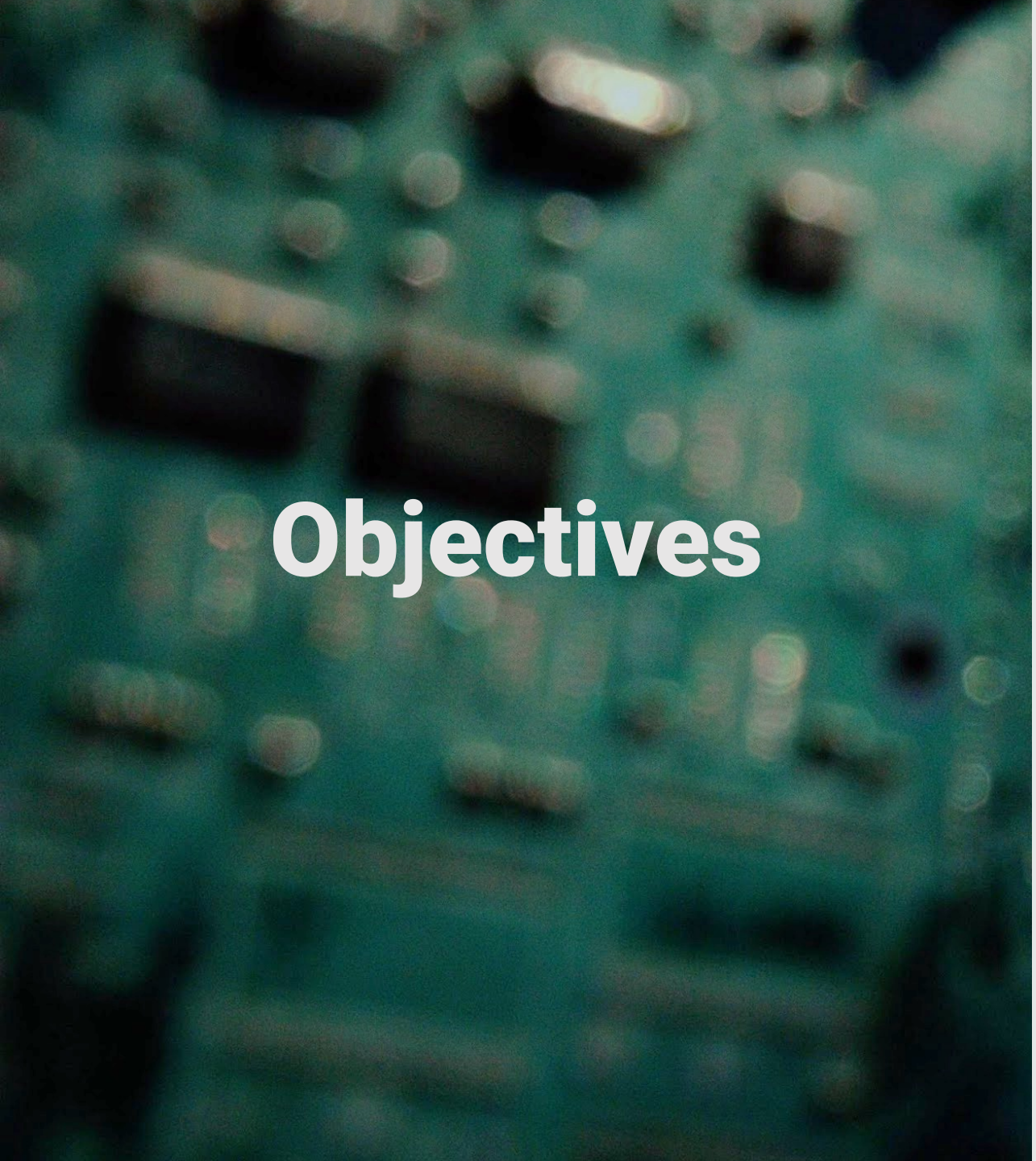
Realtime Databases and Authentications



Objectives

- Understand and use Realtime databases
- Connect to Firebase database
- Push and download data
- Enable Firebase Auth and use database rules

Firebase

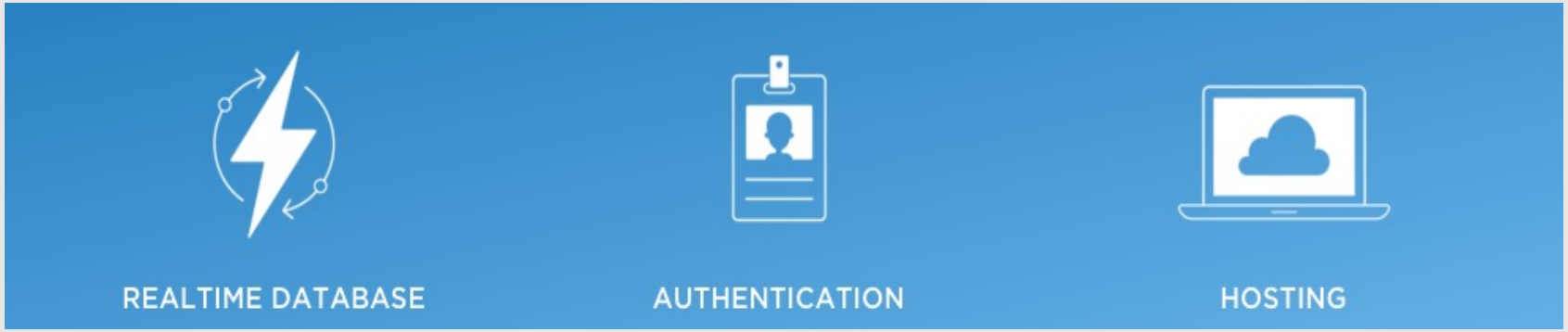
With SQL database and on a Server:

- Need to build Web Service, design Relational Schema and worry about concurrency and security

- this one is phpmyadmin which acts as ui for mysql for php

What is Firebase?

- What is Firebase?
  - BaaS (Backend-as-a-Service), which powers the backend of the App / Website
  - Main Features



  - Drive: Small businesses, Shorten development cycles, avoid repeated framework development.

What is NoSQL?

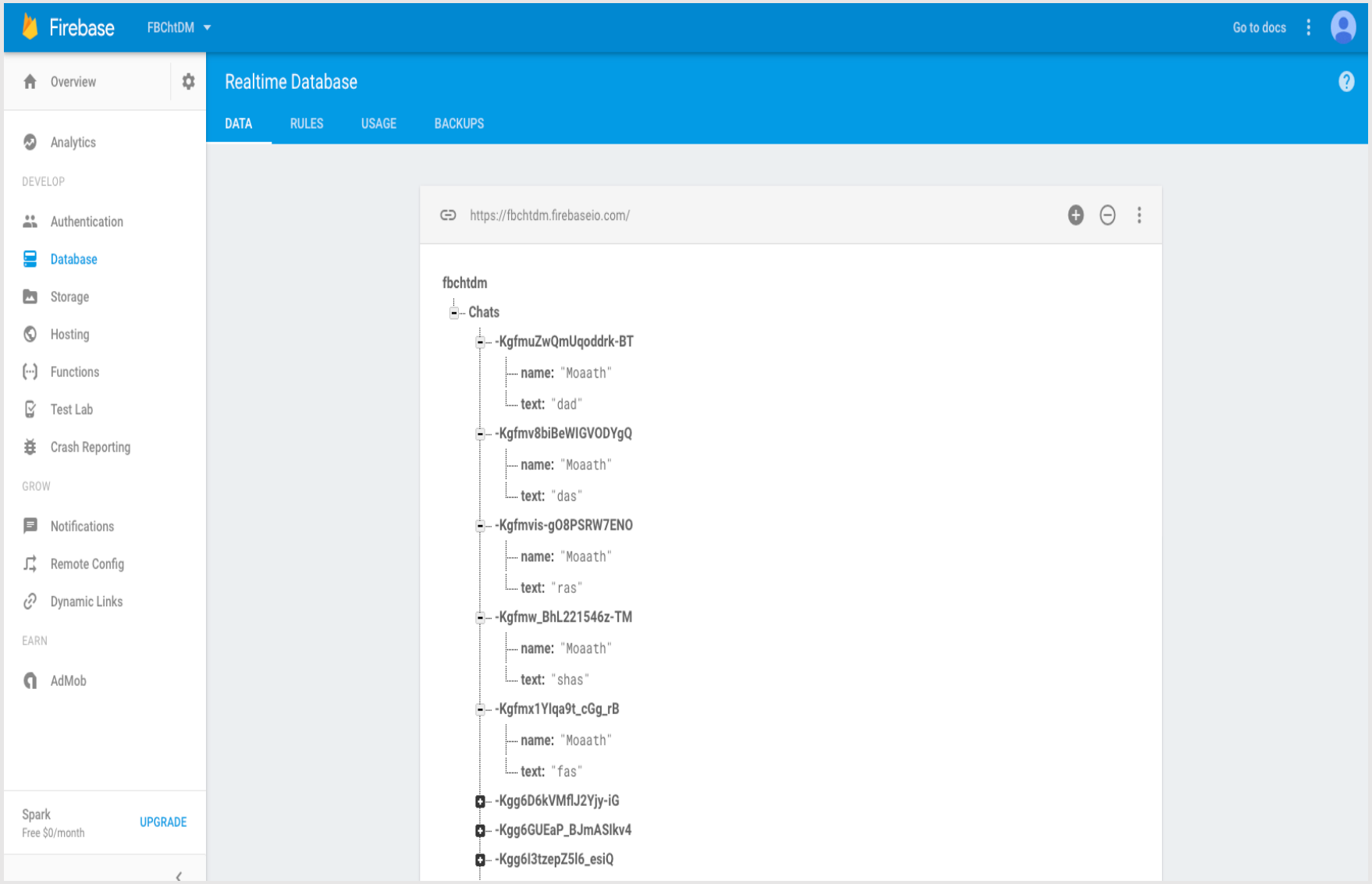
- What is NoSQL?
- not only SQL database.
  - Non-relational, distributed, open-source, and horizontally scalable.
  - Drive: Big Data, Cloud Technology, bad performance of traditional relational databases.

NoSQL databases

- **NoSQL database:** Does not store data into tables and does not use SQL.
  - became popular in mid-2000s
  - *benefits:* simplicity; flexibility; "horizontal" scalability to many servers
  - *drawbacks:* less standardized; data inconsistency/loss; lack "ACID"

- Types of NoSQL databases
  - column stores (**Cassandra**, Vertica, Druid, Accumulo)
  - document stores (**MongoDB**, CouchDB, Qizx, MarkLogic, Hyperdex)
  - key/value stores (Memcached, Scalaris, Voldemort, Dynamo)
  - data structure servers (**Redis**)
  - graph stores (Allegro, Neo4J, Virtuoso, MarkLogic)

Firebase


Firebase
FBChtDM
Go to docs
Overview
Realtime Database
Analytics
DATA
RULES
USAGE
BACKUPS
DEVELOP
Authentication
Database
Storage
Hosting
Functions
Test Lab
Crash Reporting
GROW
Notifications
Remote Config
Dynamic Links
EARN
AdMob
Spark
Free $0/month
UPGRADE
https://fbchtdm.firebaseio.com/
fbchtdm
Chats
-KgfmuZwQmUqoddrk-BT
name: "Moaath"
text: "dad"
-Kgfmv8biBeWIGVODYgQ
name: "Moaath"
text: "das"
-Kgfmvis-gO8PSRW7ENO
name: "Moaath"
text: "ras"
-Kgfmw_BhL221546z-TM
name: "Moaath"
text: "shas"
-Kgfmx1Yiqa9t_cGg_rB
name: "Moaath"
text: "fas"
-Kgg6D6kVMflJ2Yjy-iG
-Kgg6GUEaP_BJmASIkv4
-Kgg6I3tzepZ5l6_esiQ

DIY setup Firebase

- Gradle link to the current version of Firebase DB

https://firebase.google.com/docs/android/setup

How Firebase Store Data

The database is a giant nested map of string keys to values.
text, numbers, boolean, lists, or maps
object: {"key" => value} map
list: {index => value} map
overall database is tree-like map structure you can view on the web

Install Firebase into Android Studio project

- option 1: use UI: Tools → FireBase → Realtime Database → ...
- option 2: do it yourself (Gradle as above)

Writing Firebase data

```
// create a key/value pairing
val fb = FirebaseDatabase.getInstance().getReference();
fb.child("name").setValue(value);
```

Firebase stores data as key/value pairs
the keys are strings representing data object names
the values can be one of many types:
Boolean, Long, Double, List, Map<String, Object>
think of Firebase as a HashMap on steroids in the cloud

DatabaseReference methods

Method	Description
dbr.child("name")	return child object with given name (creates if missing)
dbr.getKey()	return key for a given data value
dbr.getParent()	return data one level up in the map
dbr.getRoot()	return data at top of map
dbr.push()	create/return an auto-created new child
dbr.removeValue();	
dbr.removeValue(handler);	delete value associated with this key
dbr.runTransaction(handler);	run multiple queries in sequence
dbr.setPriority(priority);	gives this data a 'priority' rating for sorting
dbr.setValue(value);	
dbr.setValue(value, handler);	
dbr.setValue(value, priority, handler);	sets new data value, with optional listener to be notified when sync is complete
dbr.updateChildren(map);	
dbr.updateChildren(map, handler);	updates some of object's fields ("children") using the key/value data in the given map

SetValue with callback

When you call setValue, the data may not update immediately on the server.
Your data might be distributed across many servers; it takes time to sync them.
To be notified when the data is fully written:

```
val database = Firebase.database
val myRef = database.getReference("message")

myRef.setValue("Hello, World!")
```

Auto-generated keys

Some tables don't have an "id" column.

Firebase can make up IDs with push().

Also useful in highly parallel situations where many users modify the data at once.

```
DatabaseReference fb = ...;
val chat= fb.child(”Msgs/chat");
val newChat = chat.push();
newChat .child(”msg_id").setValue(123);
newChat .child(”name_id").setValue(10001);
newChat .child(”other").setValue(”XX");
```

Retrieving data

Getting data is more complex than setting it.
Must grab the Firebase object for that data, and bind an event handler to it.
Will be notified initially and on state changes.

```
// Read from the database
myRef.addValueEventListener(object: ValueEventListener() {

    override fun onDataChange(snapshot: DataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        val value = snapshot.getValue<String>()
        Log.d(TAG, "Value is: " + value)
    }

    override fun onCancelled(error: DatabaseError) {
        Log.w(TAG, "Failed to read value.", error.toException())
    }

})
```

DataSnapshot methods

Method	Description
`ds.child("path")`	returns child for given key
`ds.exists()`	true if this data value is non-null
`ds.getChildren()`	returns iterable list of children (use with for-each loop)
`ds.getKey()`	returns key used to fetch this data snapshot
`ds.getPriority()`	priority of this data's root node
`ds.getRef()`	returns reference to `Firebase` object
`ds.getValue()`	returns data associated with this snapshot's key
`ds.getValue(class)`	returns data, converted into the given class (must have a `()` constructor and public `get` methods)
`ds.hasChild("path")`	true if the given child node/path exists in this data
`ds.hasChildren()`	true if this snapshot contains any data
`ds.toString()`	text representation of all the data

Querying data

More info:

https://firebase.google.com/docs/database/android/read-and-write?utm_source=studio

Security and authentication

There are several ways to allow/deny access to your database

- Firebase-specific accounts; Google accounts; etc.
- Use Firebase web UI to add email/password user accounts

Modify code to sign in with email and password:

```
private FirebaseAuth mAuth;

mAuth = FirebaseAuth.getInstance();
mAuth.signInWithEmailAndPassword("username", "password");
// optional: addOnCompleteListener, addAuthStateListener
```

FirebaseAuth methods

Method	Description
createUserWithEmailAndPassword("email", "pw")	create new account
signInWithEmailAndPassword("email", "pw")	log in a standard user
signInWithCredential(auth)	log in using access creds
signInAnonymously()	log in as anon. user
signInWithCustomToken("token")	log in with an auth token
signOut()	disconnect
getCurrentUser()	return active user account

Firebase Rules

- Firebase allows three main rule types: .read, .write. And .validate.
- Each of these can be set to “true” or “false” and can apply to the whole database or a particular location in the database depending on how they are configured.

Rule Type	Description
.read	Describes whether data can be read by the user.
.write	Describes whether data can be written by the user.
.validate child	Defines what a correctly formatted value looks like, whether it has nodes, and the data type.

Rule Types

Predefined Variables

- Firebase Database Security includes a set of predefined variables that enable you to customize data accessibility. Below is a list of predefined variables and a link to each API reference.

Variable	Description
• now	The current time in milliseconds since Unix epoch time (January 1, 1970)
• root	Corresponds to the current data at the root of the database. You can use this to read any data in your database in your rule expressions.
• newData	Corresponds to the data that will result if the write is allowed
• data	Corresponds to the current data in Firebase Realtime Database at the location of the currently executing rule.
• $variables	A wildcard path used to represent ids and dynamic child keys.
• auth	Contains the token payload if a user is authenticated, or null if the user isn't authenticated.

- We will expand on the auth variable because we will use it in database security examples.

Auth

- The auth variable contains the JSON web token for the user. A JSON Web Token is a standard that defines a way of securely transmitting information between parties, like the database and a client, as a JSON object. Once a user is authenticated, this token contains the provider, the uid, and the Firebase Auth ID token.

- The provider is the method of authentication, such as email/password, Google Sign In, or Facebook Login.

- The uid is a unique user ID. This ID is guaranteed to be unique across all providers, so a user that authenticates with Google and a user that authenticates with email/password do not risk having the same identification.

- The Firebase Auth ID is a web token. Yes, this means that there is a web token inside of the Auth web token! This token can contain data such as email, password, name and so on.

End of Slides

- Make sure to watch the lecture videos