

Angular & Firebase

Google's Special Gift

Idan Cohen | UVeye
April 2020

DEMO CODE:

<https://github.com/IdanCo/angular-firebase-demo>

@Idan_Co

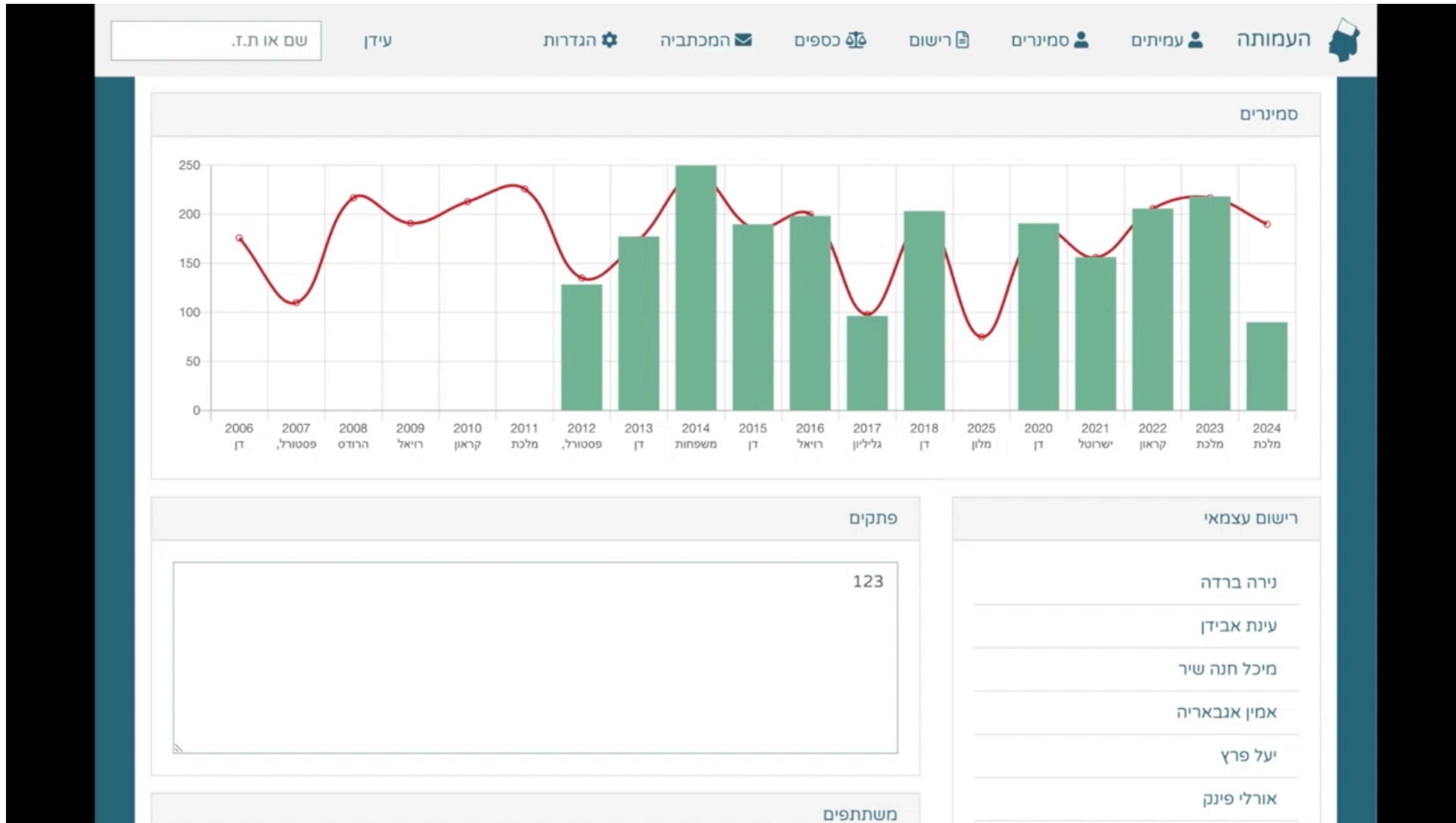
idancohen.com

UVeye

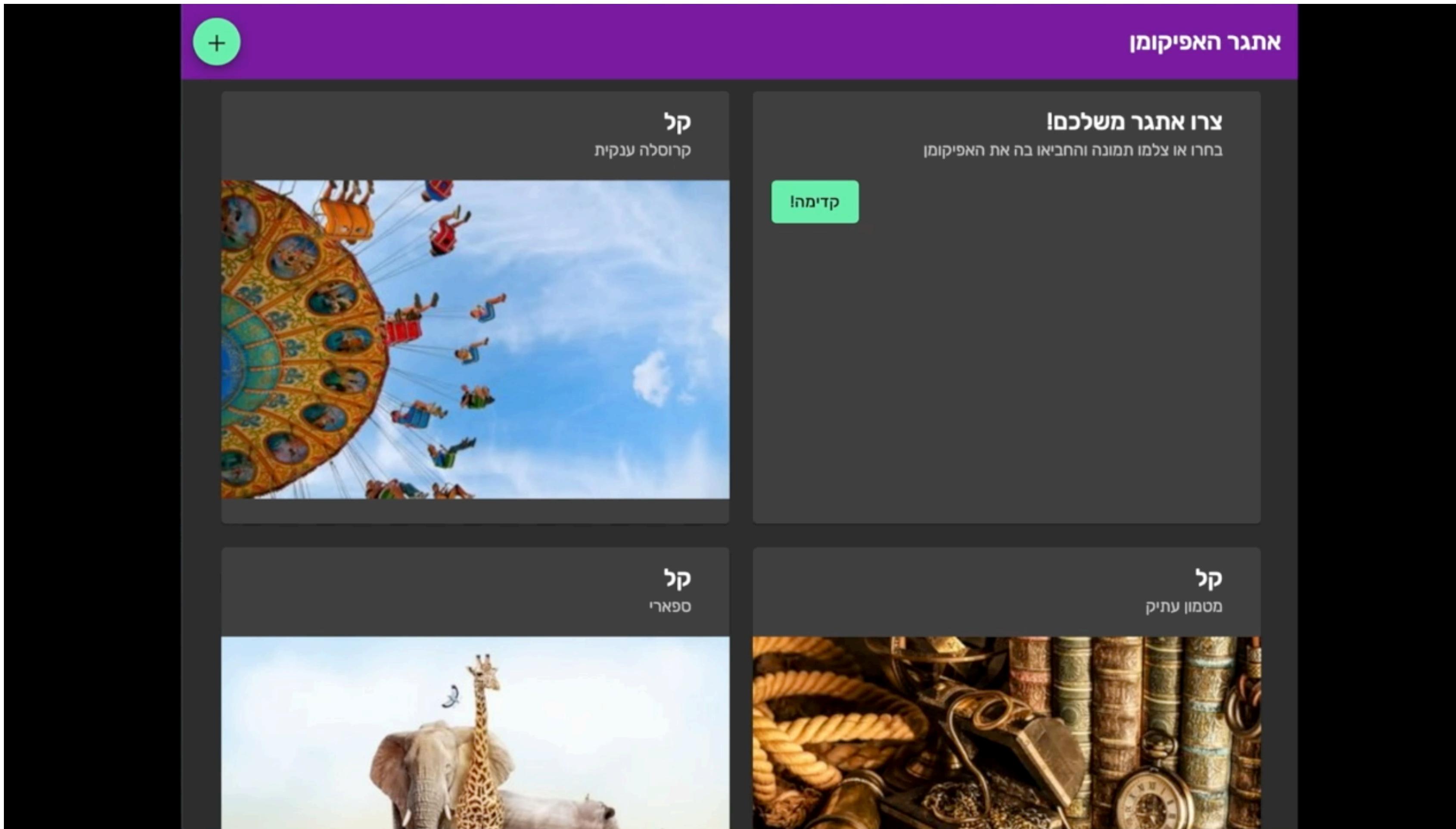
Scans

Overview	LP/VIN	Scanned at ↓	Make	Model	Year	Site	Status
	8844338	Thursday, March 26th 11:27	N/A	N/A	N/A	Demo lane	
	8844338	Wednesday, March 25th 1...	N/A	N/A	N/A	ICO lane	
	7133476	Tuesday, March 24th 00:12	N/A	Audi A7	N/A	Audi lane	
	74532601	Monday, March 23rd 23:58	N/A	Audi Q7	N/A	Audi lane	
	12176502	Monday, March 23rd 23:51	N/A	Audi A3	N/A	Audi lane	

CRM System



afikoman.idancohen.com

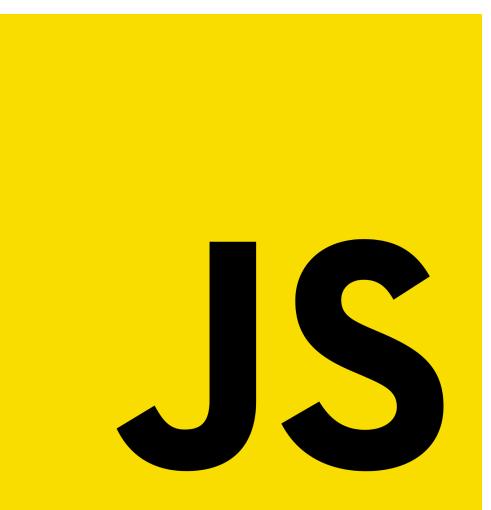




Angular



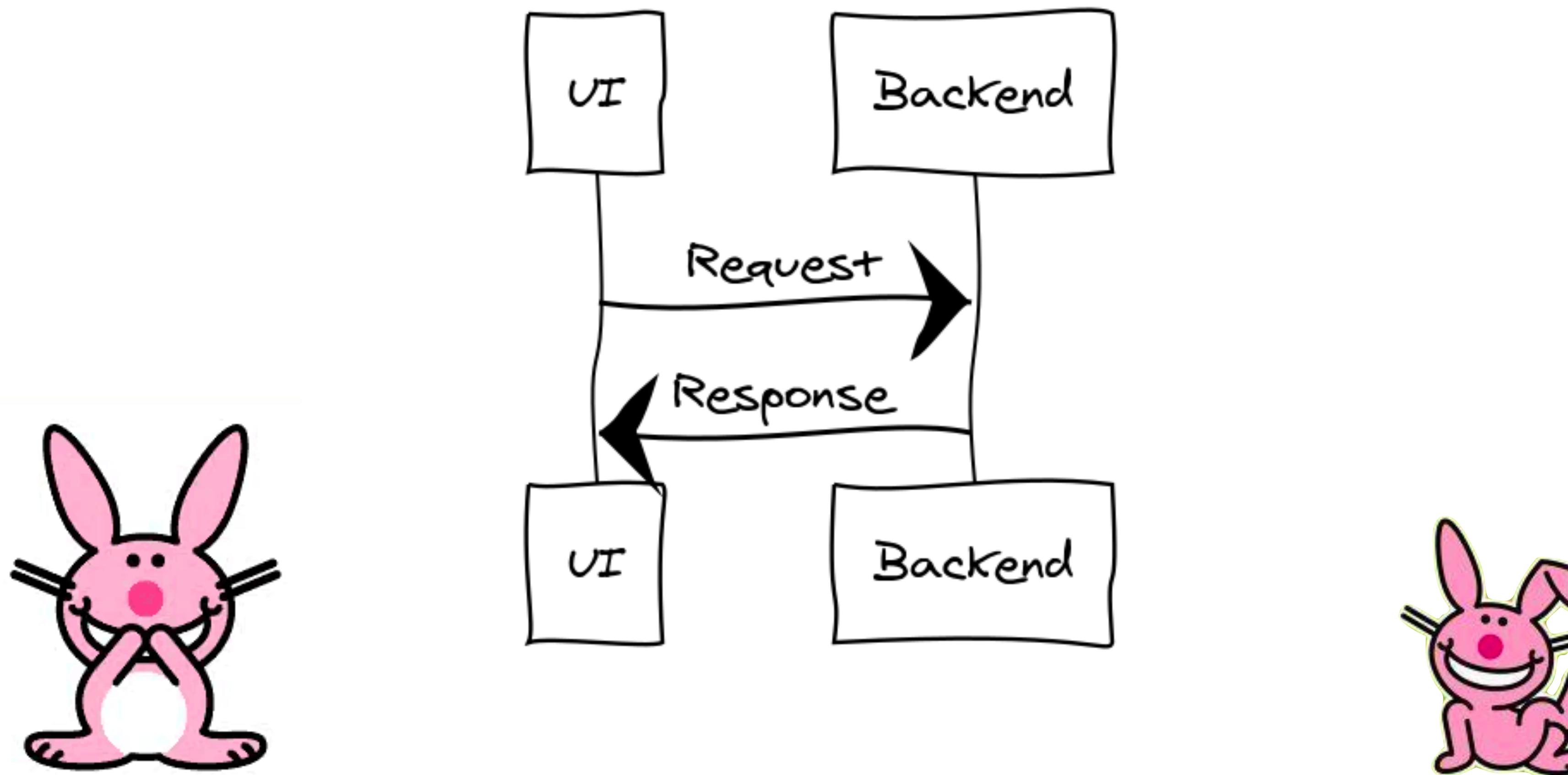
Firebase



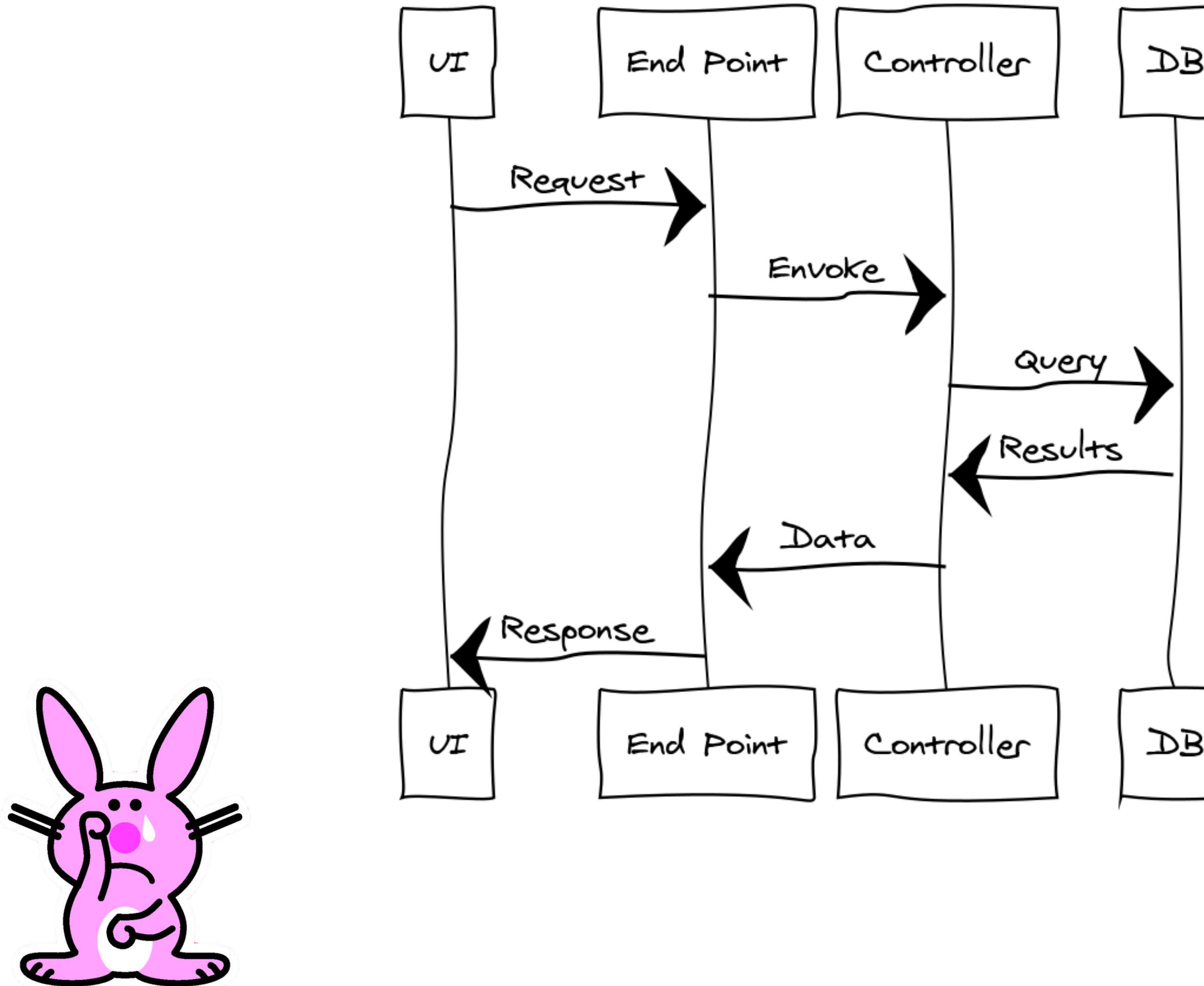
Stepping into the Fire

- Database
- Hosting
- Authentication
- File Storage
- Cloud Functions
- Analytics
- Messaging

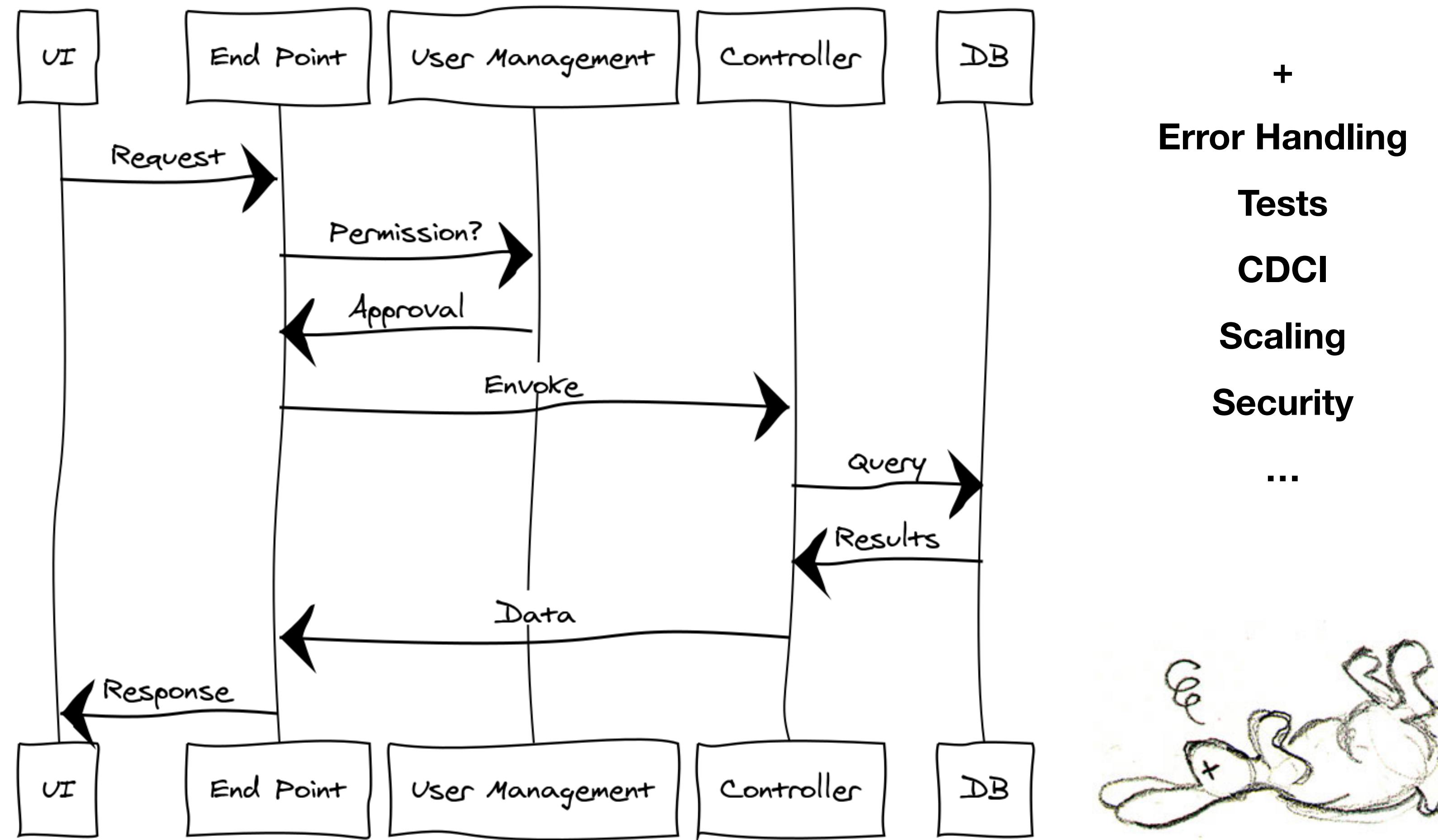
Basic Update Flow

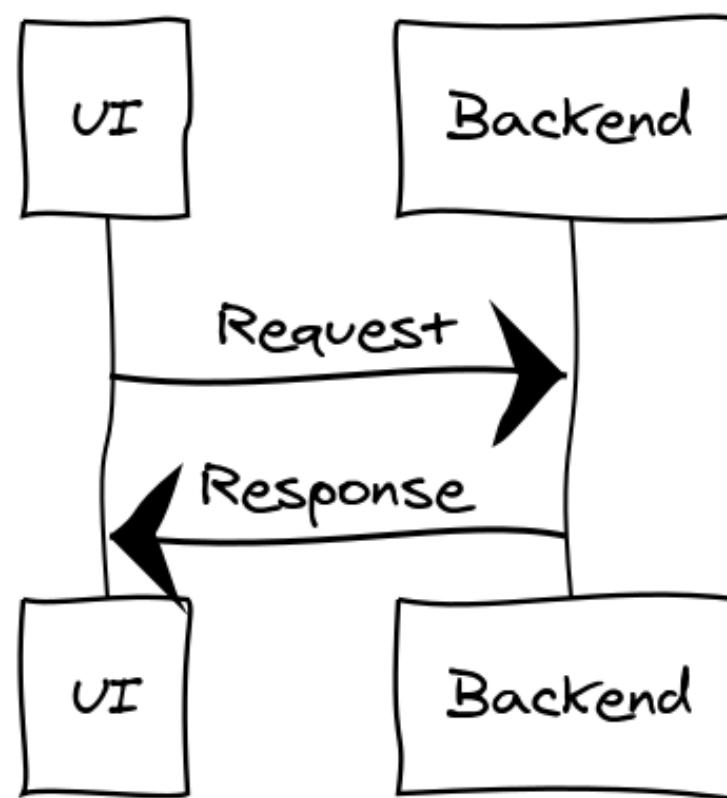
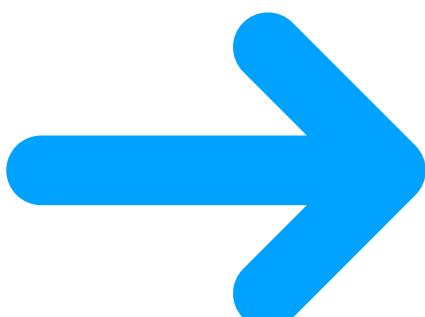
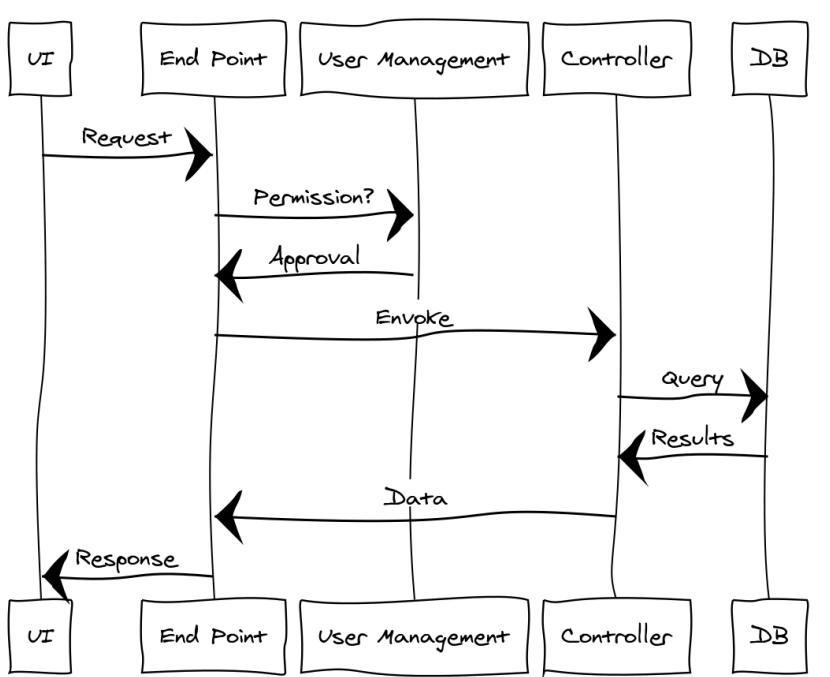
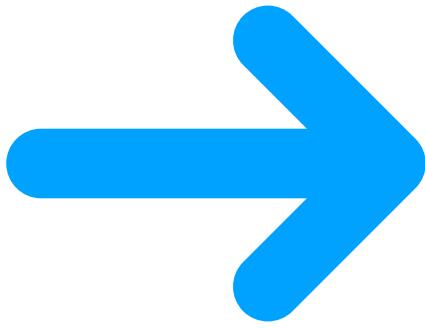


~~Basic Normal Update Flow~~



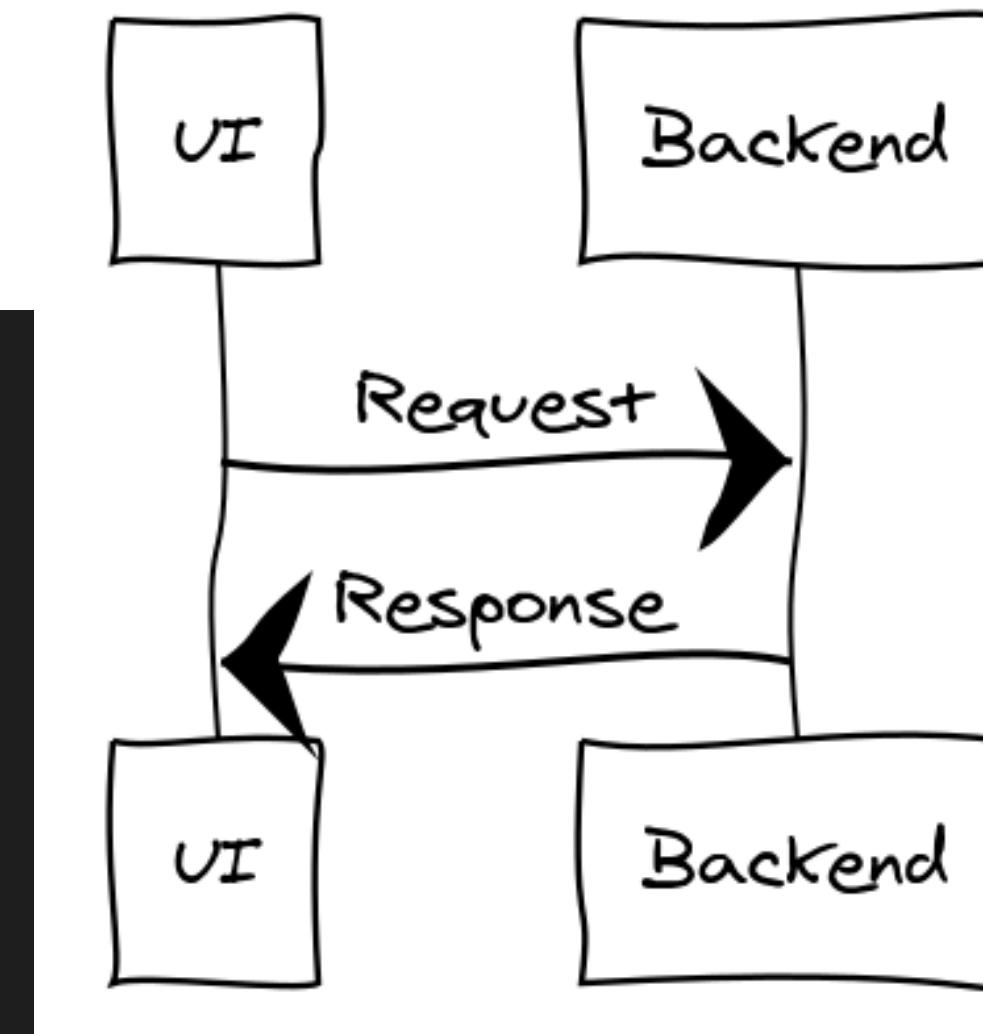
~~Basic Normal Full Update Flow~~





Firebase Bunnyland

```
db.collection("users").add({
  first: "Ada",
  last: "Lovelace",
  born: 1815
})
.then(function(docRef) {
  console.log("Document written with ID: ", docRef.id);
})
.catch(function(error) {
  console.error("Error adding document: ", error);
});
```

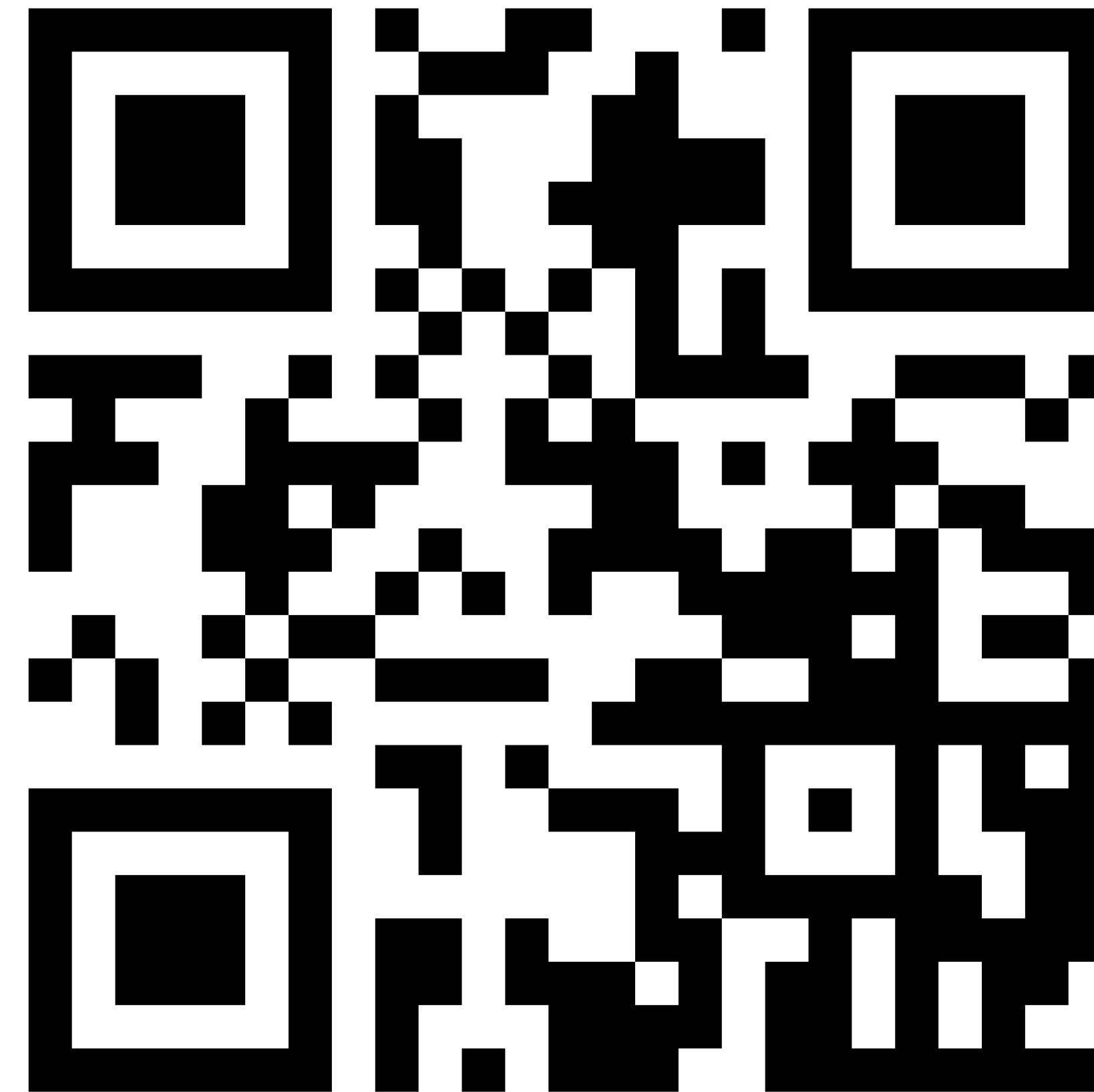


Firebase Bunnyland

- End Point
- Controller
- DB
- Authorization
- Error Handling
- Tests
- CDCI
- Scaling
- Security

```
db.collection("users").add({  
    first: "Ada",  
    last: "Lovelace",  
    born: 1815  
})  
.then(function(docRef) {  
    console.log("Document written with ID: ", docRef.id);  
})  
.catch(function(error) {  
    console.error("Error adding document: ", error);  
});
```

all-the-talks.web.app



Resources

- Angular docs - <https://angular.io/docs>
- Firebase docs - <https://firebase.google.com/docs>
- Angular library for Firebase -<https://github.com/angular/angularfire>
- Firebase Youtube Channel - https://www.youtube.com/channel/UCP4bf6IHJJQehibu6ai_cg

Firestore Data Structure

Documents

alovelace

```
first : "Ada"  
last : "Lovelace"  
born : 1815
```

alovelace

```
name :  
first : "Ada"  
last : "Lovelace"  
born : 1815
```

```
db.collection('users').doc('alovelace');
```

```
db.doc('users/alovelace');
```

Firestore Data Structure

Collections



users



alovelace

first : "Ada"

last : "Lovelace"

born : 1815



aturing

first : "Alan"

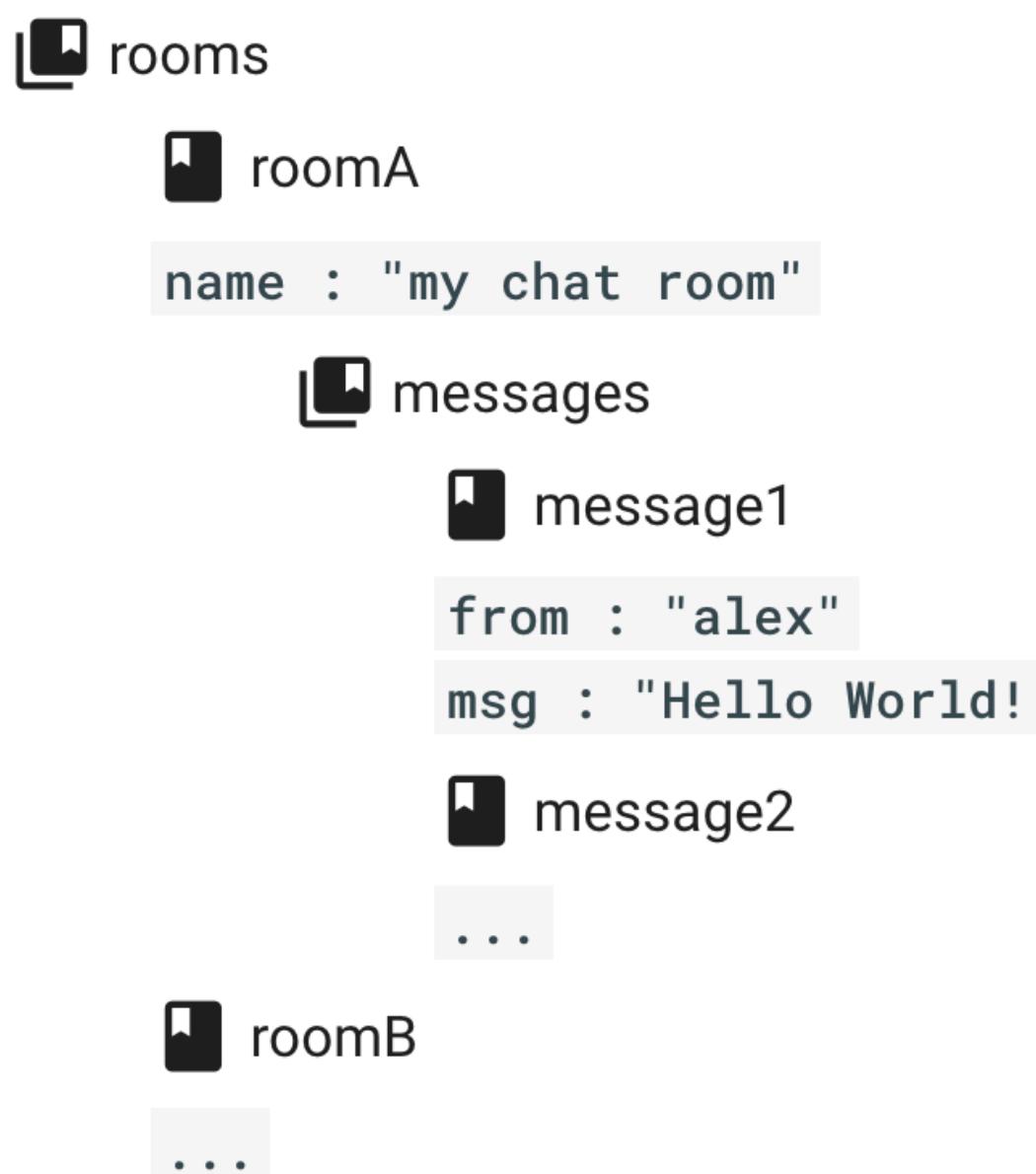
last : "Turing"

born : 1912

```
db.collection('users');
```

Firestore Data Structure

Sub-Collections



```
db.collection('rooms').doc('roomA')
  .collection('messages').doc('message1');
```

Firestore Data Structure

Queries

```
db.collection("cities").where("capital", "==", true)
```

```
citiesRef.where("regions", "array-contains", "west_coast")
```

```
citiesRef.where('state', '==', 'CO').where('name', '==', 'Denver');  
citiesRef.where('state', '==', 'CA').where('population', '<', 1000000);
```

```
citiesRef.where('state', '>=', 'CA').where('population', '>', 1000000);
```

 OR

 !=

 Join

Firestore Data Structure

Order & Limit

```
citiesRef.orderBy('name').limit(3);
```

```
citiesRef.orderBy('name', 'desc').limit(3);
```

```
citiesRef.orderBy('state').orderBy('population', 'desc');
```

```
citiesRef.where('population', '>', 2500000).orderBy('population').limit(2);
```

```
citiesRef.where('population', '>', 2500000).orderBy('country');
```

Authorization

```
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /<some_path>/ {  
      allow read, write: if <some_condition>;  
    }  
  }  
}
```

Authorization

```
service cloud.firestore {  
  match /databases/{database}/documents {  
  
    // Match any document in the 'cities' collection  
    match /cities/{city} {  
      allow read: if <condition>;  
      allow write: if <condition>;  
    }  
  }  
}
```

Authorization

```
service cloud.firestore {
  match /databases/{database}/documents {
    // A read rule can be divided into get and list rules
    match /cities/{city} {
      // Applies to single document read requests
      allow get: if <condition>;

      // Applies to queries and collection read requests
      allow list: if <condition>;
    }

    // A write rule can be divided into create, update, and delete rules
    match /cities/{city} {
      // Applies to writes to nonexistent documents
      allow create: if <condition>

      // Applies to writes to existing documents
      allow update: if <condition>

      // Applies to delete operations
      allow delete: if <condition>;
    }
  }
}
```

Authorization

```
service cloud.firestore {  
    match /databases/{database}/documents {  
        // Allow the user to access documents in the "cities" collection  
        // only if they are authenticated.  
        match /cities/{city} {  
            allow read, write: if request.auth.uid != null;  
        }  
    }  
}
```

```
service cloud.firestore {  
    match /databases/{database}/documents {  
        // Make sure the uid of the requesting user matches name of the user  
        // document. The wildcard expression {userId} makes the userId variable  
        // available in rules.  
        match /users/{userId} {  
            allow read, update, delete: if request.auth.uid == userId;  
            allow create: if request.auth.uid != null;  
        }  
    }  
}
```

Authorization

```
service cloud.firestore {  
  match /databases/{database}/documents {  
    // Allow the user to read data if the document has the 'visibility'  
    // field set to 'public'  
    match /cities/{city} {  
      allow read: if resource.data.visibility == 'public';  
    }  
  }  
}
```

```
service cloud.firestore {  
  match /databases/{database}/documents {  
    // Make sure all cities have a positive population and  
    // the name is not changed  
    match /cities/{city} {  
      allow update: if request.resource.data.population > 0  
                    && request.resource.data.name == resource.data.name;  
    }  
  }  
}
```

Authorization

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /cities/{city} {
      // Make sure a 'users' document exists for the requesting user before
      // allowing any writes to the 'cities' collection
      allow create: if exists(/databases/${database}/documents/users/${request.auth.uid}))

      // Allow the user to delete cities if their user document has the
      // 'admin' field set to 'true'
      allow delete: if get(/databases/${database}/documents/users/${request.auth.uid}).data.admin == true
    }
  }
}
```

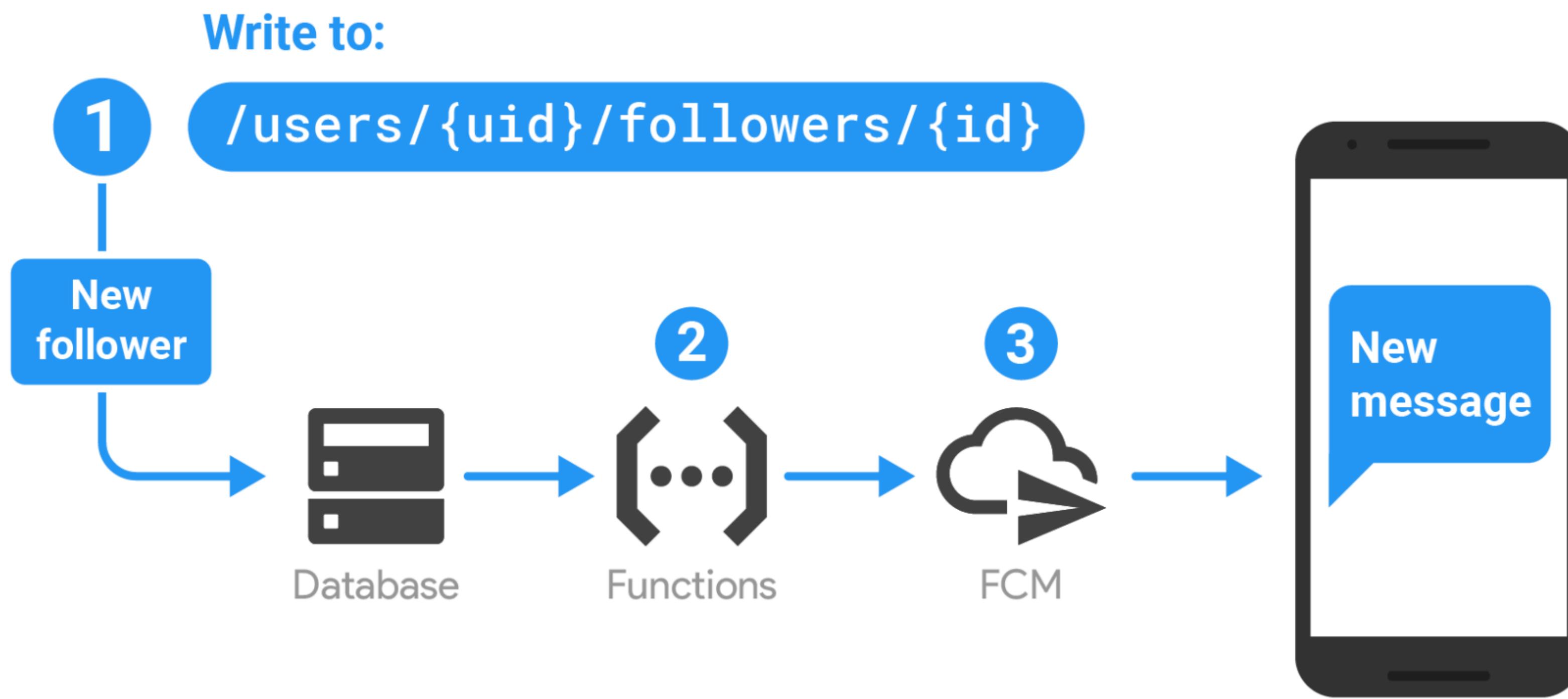
Cloud Functions

- Triggers:
 - Database Events
 - Authentication Events
 - Storage Events
 - HTTP Events

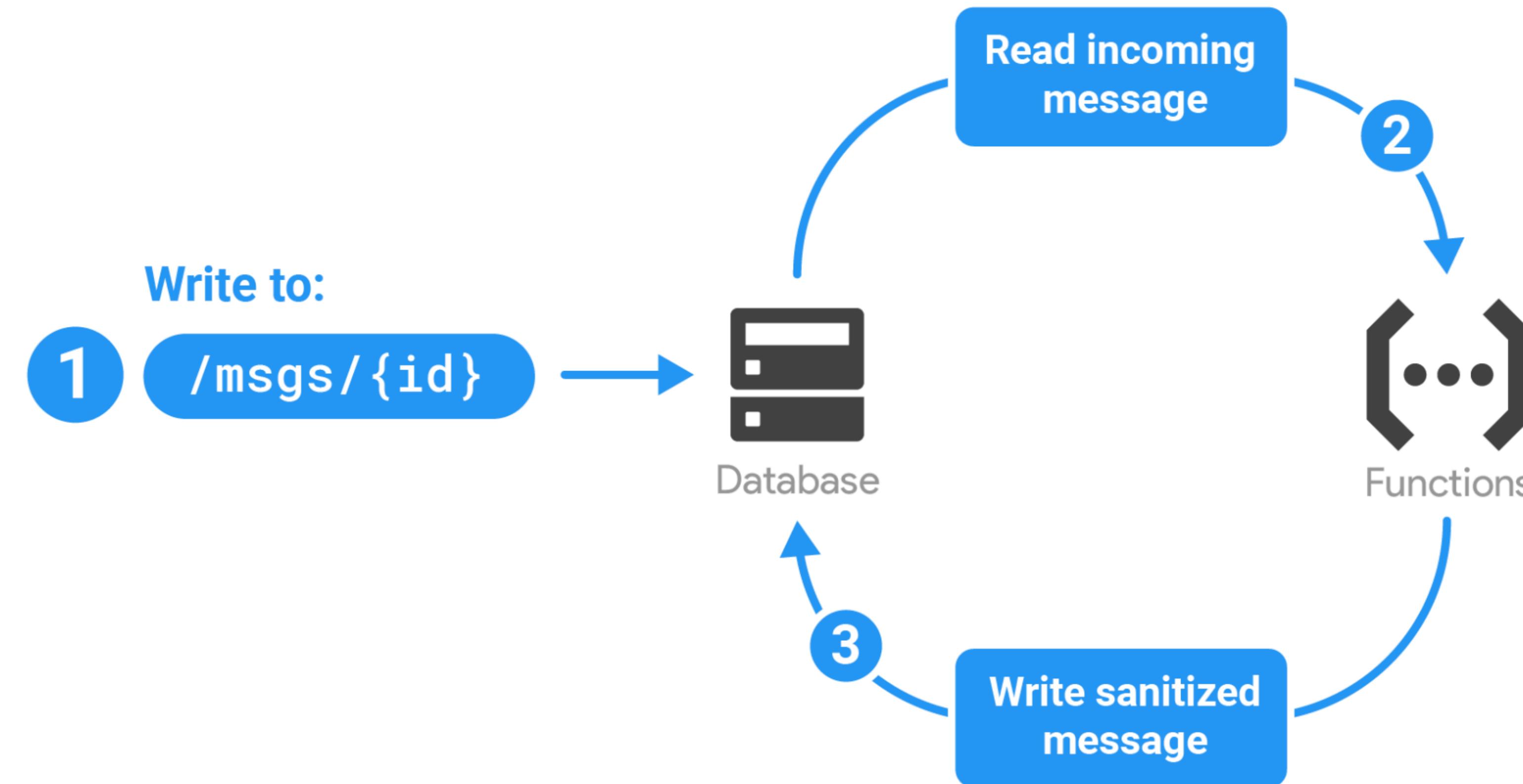
* Auto Scale *

* Test Locally *

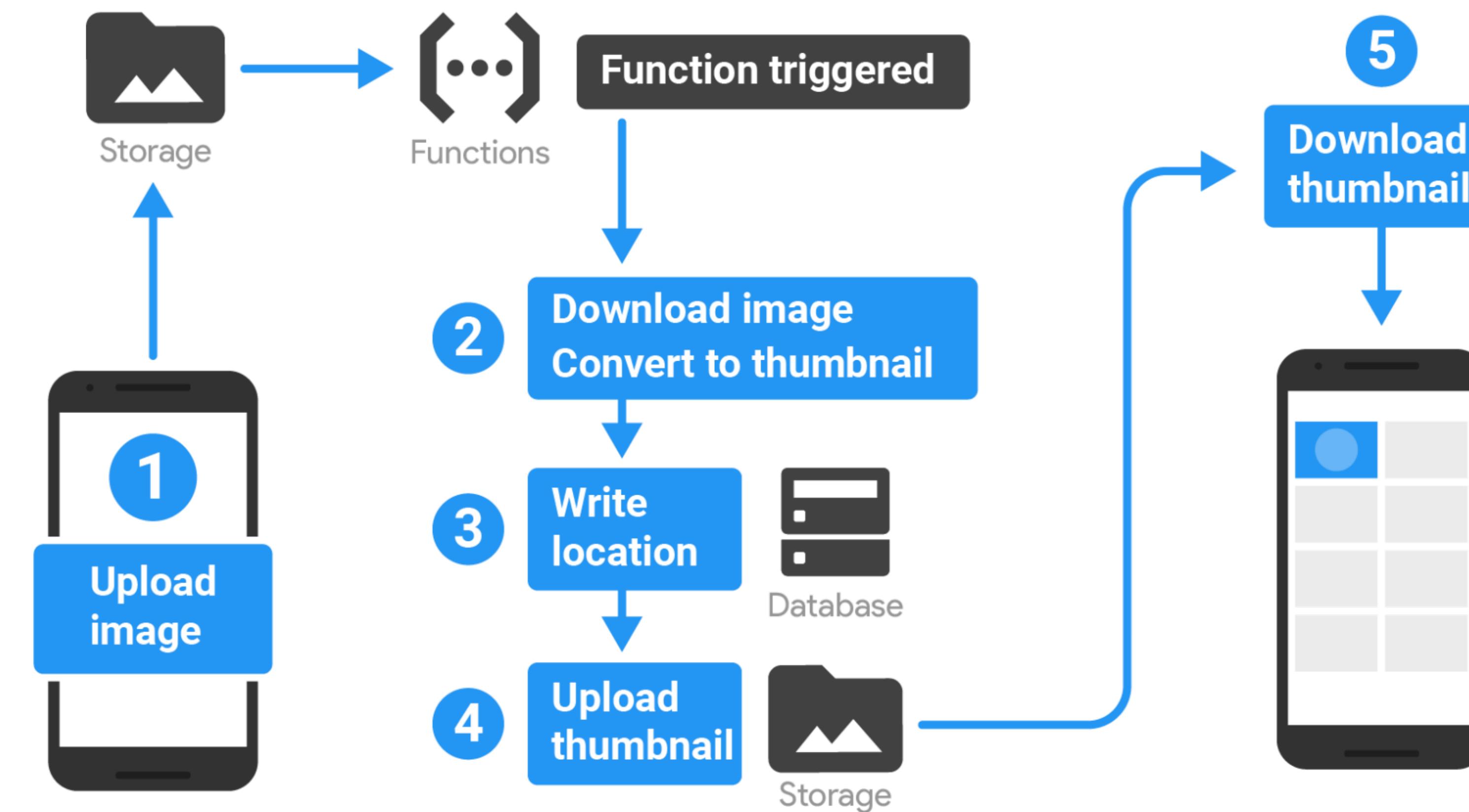
Cloud Functions



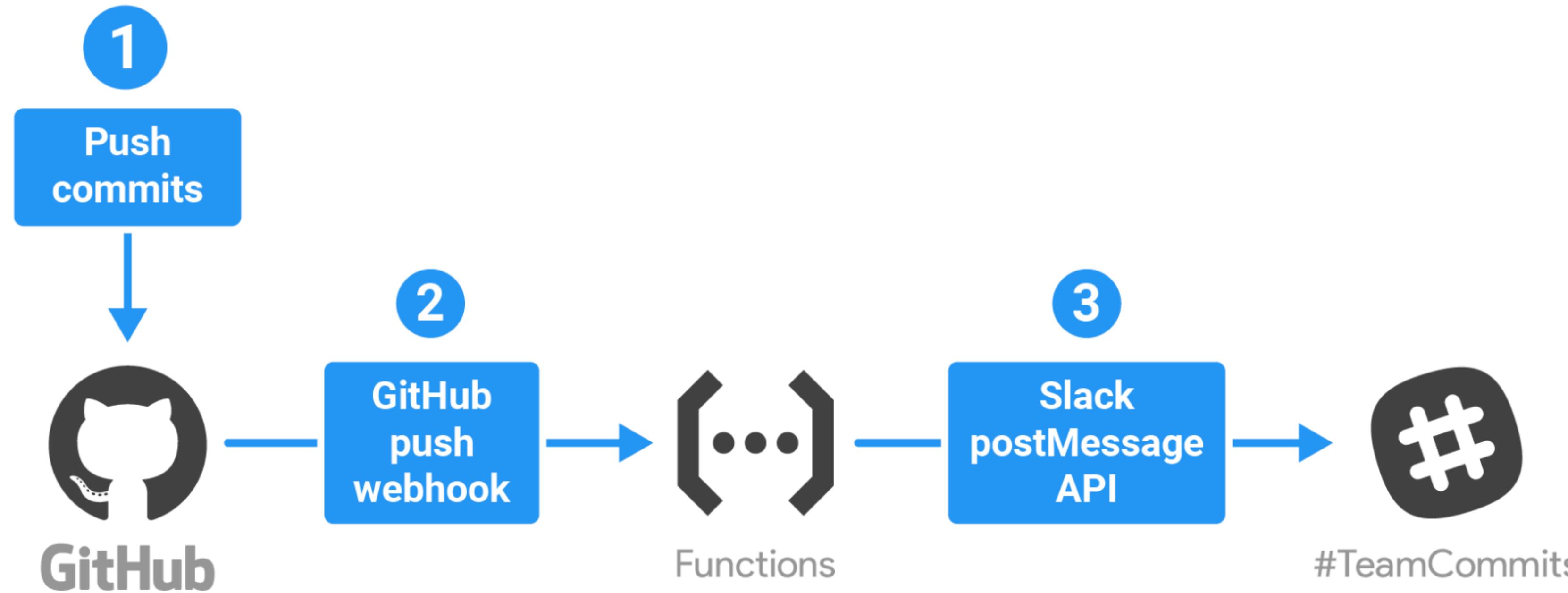
Cloud Functions



Cloud Functions



Cloud Functions



Cloud Functions

```
exports.updateUser = functions.firestore
  .document('users/{userId}')
  .onUpdate((change, context) => {
    // Get an object representing the document
    // e.g. {'name': 'Marie', 'age': 66}
    const newValue = change.after.data();

    // ...or the previous value before this update
    const previousValue = change.before.data();

    // access a particular field as you would any JS property
    const name = newValue.name;

    // perform desired operations ...
  });
}
```

Angular & Firebase

Google's Special Gift

Idan Cohen | UVeye
April 2020

@Idan_Co

www.idancohen.com