



KINGSLAND  
UNIVERSITY

## Remote Databases



 kinvey



# Table of Contents

1. Relational and Non Relational Databases
2. Backend as a Service - BaaS
3. Firebase
  - ✔ Using Firebase BaaS
4. Kinvey
  - ✔ Using Kinvey mBaaS

# Relational Databases

- ✓ Represent and store **data** in tables and rows
- ✓ Use **Structured Querying Language (SQL)**
- ✓ Allows you to **link information** from different tables through the use of **foreign keys** (or indexes)





# Non Relational Databases

- ✔ No-SQL databases
- ✔ More **flexibility** and **adaptability**
- ✔ Allow us to **store unstructured data** in a single document
- ✔ **Additional processing** effort and **more storage** as the document sizes grow





# Relational and Non Relational Pros

## Relational

- ✔ Work with **structured data**
- ✔ They support **ACID** transactional consistency and support "**joins**"
- ✔ Built-in **data integrity** and a large ecosystem
- ✔ Relationships in this system have **constraints**
- ✔ Limitless **indexing**

## Non Relational

- ✔ They **scale** out **horizontally**
- ✔ Work with **unstructured** and semi-structured **data**
- ✔ Schema-free or Schema-on-read options
- ✔ **High availability**
- ✔ Many are **open source** and so "free"



# Backend As a Service

- ✔ Solutions that provide **pre-built, cloud hosted** components for developing application backends
- ✔ **Reduce** the **time** and **complexity** required
- ✔ Allow developers to **focus** on **core features** instead of low level tasks
- ✔ Types:
  - ✔ Cloud BaaS
  - ✔ Opensource BaaS



Real-Time Cloud DB and App Platform by Google

**firebase**





# Firestore

- ✔ Mobile and web development platform. It provides:
  - ✔ Realtime database
  - ✔ Backend as a service
  - ✔ JSON-based data structure





# Accessing Firebase REST API with Postman

The screenshot shows the Postman interface with a GET request to the Firebase REST API. The URL is `https://testapp-fc138.firebaseio.com/.json`. The response is a JSON object with a `books` array containing two book entries.

**Append .json to your DB object URL**

**Request:**

- Method: GET
- URL: `https://testapp-fc138.firebaseio.com/.json`

**Response:**

```
1 {
2   "books": [
3     null,
4     {
5       "author": "Ivan Vazov",
6       "title": "Under the Yoke"
7     },
8     {
9       "author": "Svetlin Nakov & Co.",
10      "title": "C# Fundamentals"
11    }
12  ]
13 }
```



# Firestore REST API – CRUD Operations

|        |   |
|--------|---|
| GET    | <a href="https://testapp-fc138.firebaseio.com/.json">https://testapp-fc138.firebaseio.com/.json</a>                             |
| GET    | <a href="https://testapp-fc138.firebaseio.com/books.json">https://testapp-fc138.firebaseio.com/books.json</a>                   |
| GET    | <a href="https://testapp-fc138.firebaseio.com/books/1.json">https://testapp-fc138.firebaseio.com/books/1.json</a>               |
| GET    | <a href="https://testapp-fc138.firebaseio.com/books/1/author.json">https://testapp-fc138.firebaseio.com/books/1/author.json</a> |
| POST   | <a href="https://testapp-fc138.firebaseio.com/books.json">https://testapp-fc138.firebaseio.com/books.json</a>                   |
| Body   | <code>{"title":"New title", "author":"New author"}</code>   |
| DELETE | <a href="https://testapp-fc138.firebaseio.com/books/6.json">https://testapp-fc138.firebaseio.com/books/6.json</a>               |



# Firestore REST API – CRUD Operations (2)

|        |   |
|--------|---|
| PUT    | <a href="https://testapp-fc138.firebaseio.com/books/7.json">https://testapp-fc138.firebaseio.com/books/7.json</a>               |
| Body   | <code>{"title":"Edited", "year":1980, "ISBN":"954X"}</code>   |
| PATCH  | <a href="https://testapp-fc138.firebaseio.com/books/7.json">https://testapp-fc138.firebaseio.com/books/7.json</a>               |
| Body   | <code>{"year":1981, "author":"Author Changed"}</code>   |
| PUT    | <a href="https://testapp-fc138.firebaseio.com/books/7/author.json">https://testapp-fc138.firebaseio.com/books/7/author.json</a> |
| Body   | <code>"New author was assigned"</code>  |
| DELETE | <a href="https://testapp-fc138.firebaseio.com/books/7/author.json">https://testapp-fc138.firebaseio.com/books/7/author.json</a> |



# Authentication vs. Authorization

## ✔ Authentication

- ✔ The process of verifying the identity of a user or computer
- ✔ Questions: "**Who are you?**", "**How you prove it?**"
- ✔ Credentials can be password, smart card, external token, etc.

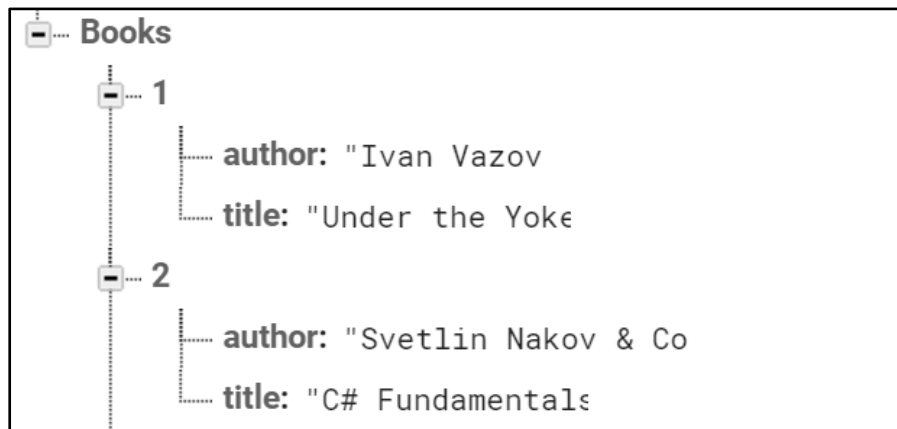
## ✔ Authorization

- ✔ The process of determining what a user is permitted to do on a computer or network
- ✔ Questions: " **What are you allowed to do?**", "**Can you see this page?**"



# Firebase and Postman: All Books

- ✓ Open Firebase and create "**TestApp**" project
- ✓ Create the following structure



- ✓ Get all books by AJAX request



# Firebase and Postman: Get Book

- ✔ Using the recently created project in **Firebase**
  - ✔ Get the book with **id: 1**
  - ✔ Don't forget the **.json** extension at the end



# Firebase and Postman: Create Book

- ✓ To create a book in our Firebase project
  - ✓ Send a "**POST**" request
    - ✓ JSON body should be in the following format:

```
{  
  "title": "New title",  
  "author": "New author"  
}
```





# Collection-Based Cloud DB (MBaaS)

**Kinvey**



# Kinvey As Back-End

- ✔ Mobile Back-End as a Service (mBaaS)
  - ✔ Holds your app / mobile **app data** in the cloud
  - ✔ Hold **users** and **user data**
  - ✔ Users (API for creating an account)
  - ✔ Data collections (API for CRUD operations)
  - ✔ Files (upload / download / delete)





# Kinvey and Postman: Register

✔ URL: [https://baas.kinvey.com/user/{app\\_id}](https://baas.kinvey.com/user/{app_id})

✔ Method: **POST**

✔ Authentication: Kinvey

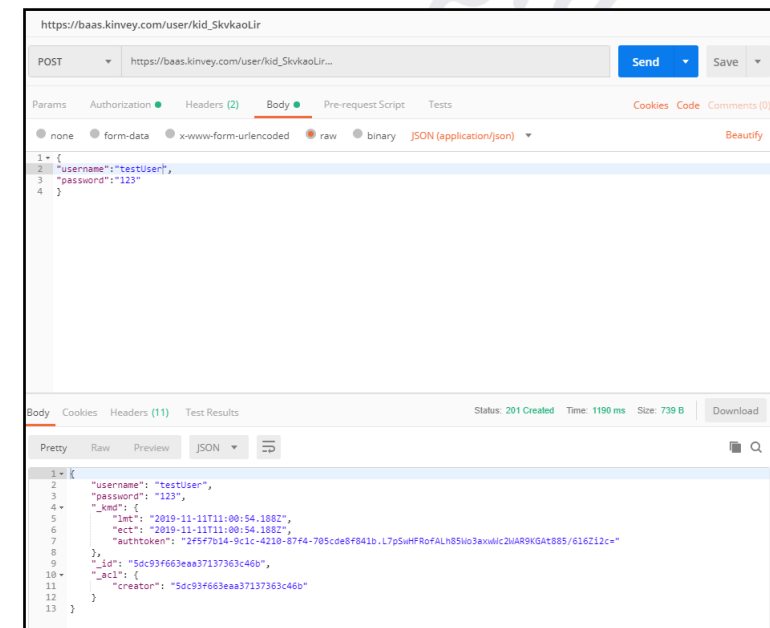
✔ User/pass: **app\_key : app\_secret**

✔ Body:

```
{"username": "...", "password": "..."} 
```

✔ Returns: **authtoken**

```
"authtoken": "fd6d989d-0930-4c...wI=" 
```





# Kinvey and Postman: Login

✓ URL: [https://baas.kinvey.com/user/{app\\_id}/login](https://baas.kinvey.com/user/{app_id}/login)

✓ Method:

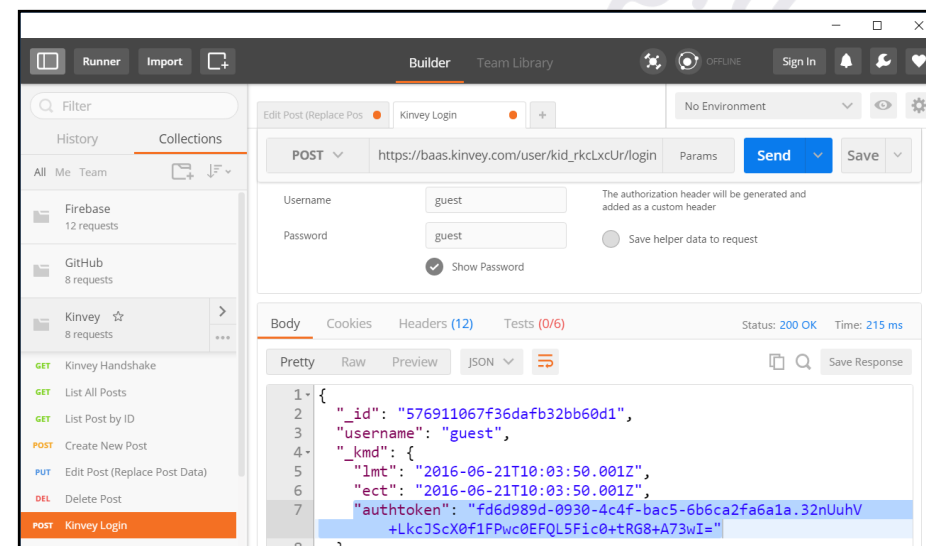
✓ Authentication: Kinvey

✓ User/pass:

✓ Body:

```
{"username": "...", "password": "..."} 
```

✓ Returns: "authtoken": "fd6d989d-0930-4c...wI="



# Kinvey and Postman: Logout

✓ URL: [https://baas.kinvey.com/user/{app\\_id}/\\_logout](https://baas.kinvey.com/user/{app_id}/_logout)

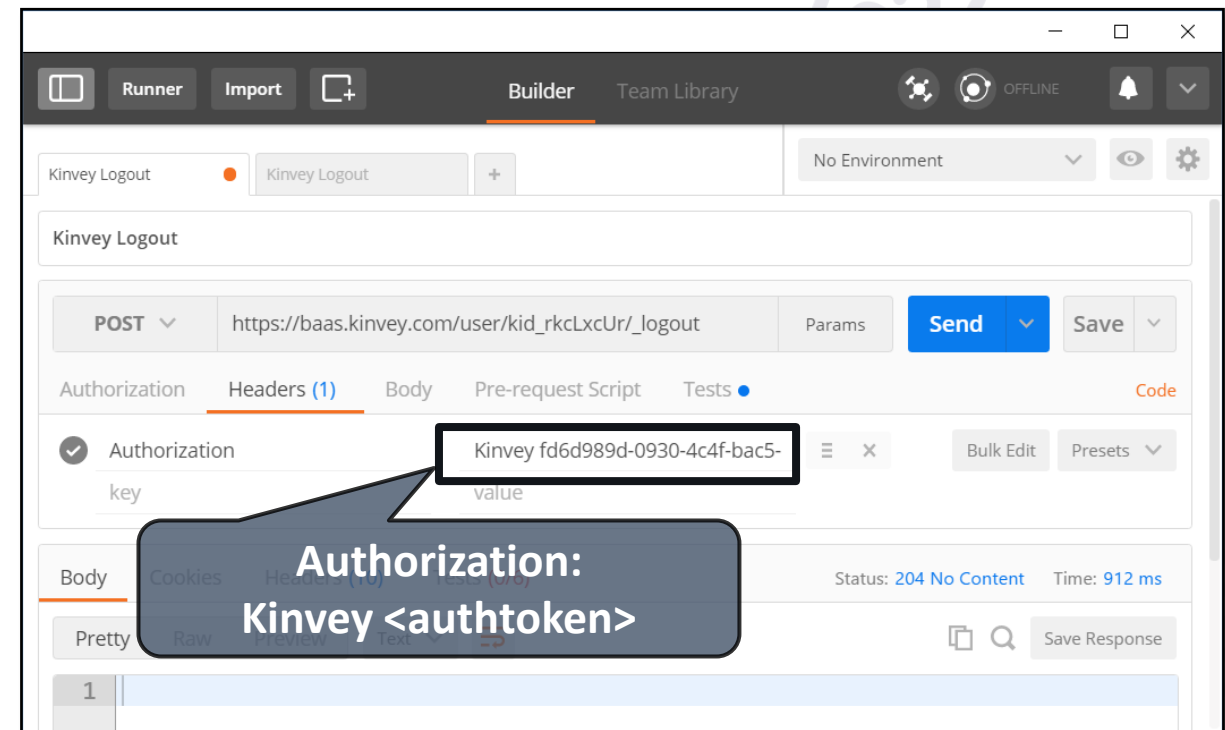
✓ Method: **POST**

✓ Authorization:

✓ **Kinvey <authtoken>**

✓ Use the token given  
by the login request

✓ Returns **204 No Content**





# Test Your Backend with Postman

The screenshot shows the Postman interface for a GET request. The URL bar contains `https://baas.kinvey.com/appdata/kid_HJa2FEHV`, which is highlighted with a red box and a callout bubble. The **Authorization** tab is selected, showing **Basic Auth** with a **Username** of `guest` and a **Password** of `.....`, both highlighted with red boxes. A callout bubble explains: **Basic authentication user: guest pass: guest**. The **Body** tab is selected at the bottom, showing a JSON response in **Pretty** format:

```
1 {
2   "version": "3.10.21",
3   "kinvey": "hello MyApp",
4   "appName": "MyApp",
5   "environmentName": "Development"
6 }
```

At the top right of the interface, a callout bubble displays the URL template: `https://baas.kinvey.com/appdata/{app_id}`.



# Kinvey and Postman: List All Posts

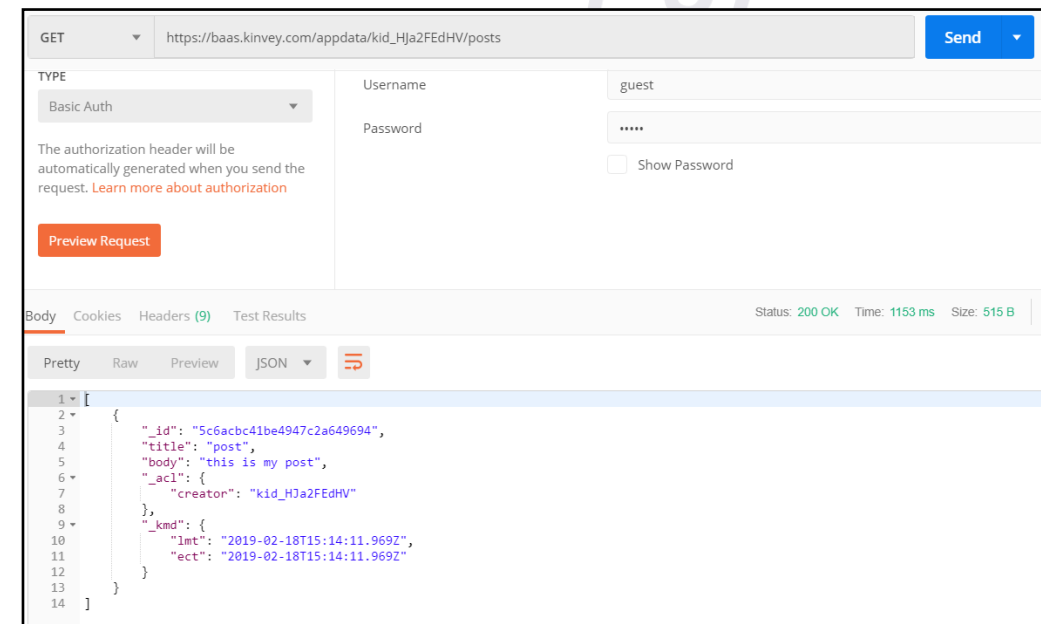
✓ URL: [https://baas.kinvey.com/appdata/{app\\_id}/posts](https://baas.kinvey.com/appdata/{app_id}/posts)

✓ Method: **GET**

✓ Authentication: Kinvey + token

✓ User: guest

✓ Pass: guest



# Kinvey and Postman: Create a New Post

✓ URL: [https://baas.kinvey.com/appdata/{app\\_id}/posts](https://baas.kinvey.com/appdata/{app_id}/posts)

✓ Method: **POST**

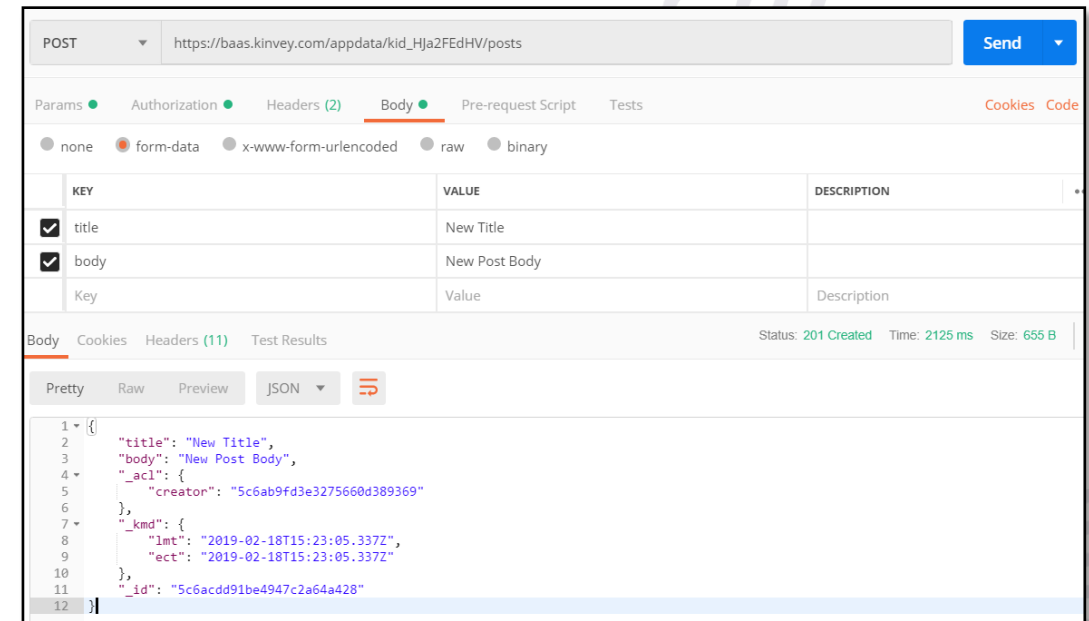
✓ Authentication: Kinvey + token

✓ User / pass: guest / guest

✓ Request body

✓ title: New Title

✓ body: New Post Body





# Kinvey and Postman: Delete an Existing Post

✓ URL: [https://baas.kinvey.com/appdata/{app\\_id}/posts/id](https://baas.kinvey.com/appdata/{app_id}/posts/id)

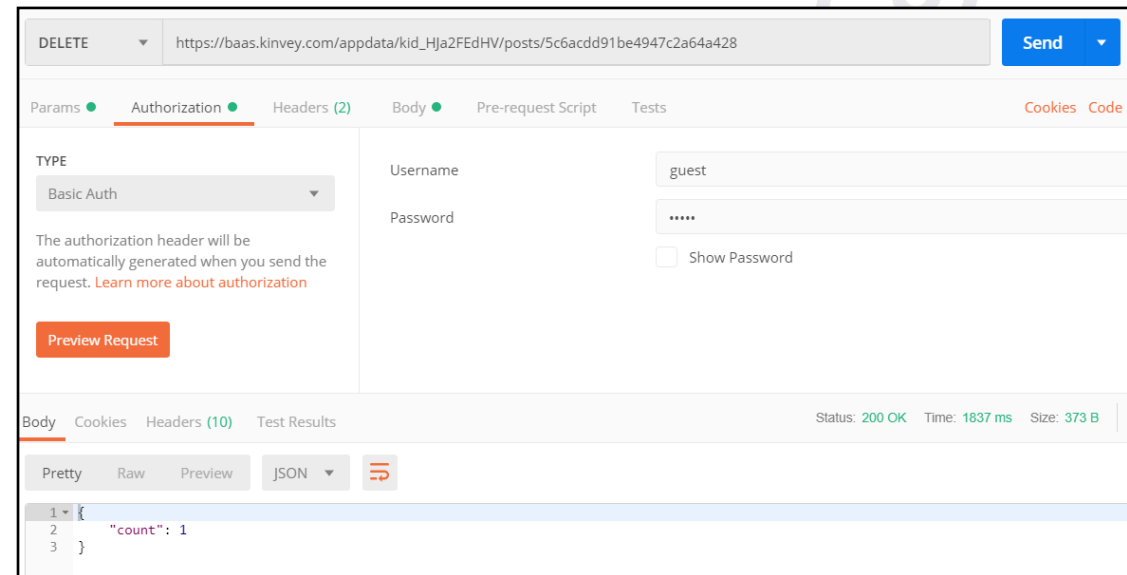
✓ Choose an existing post's ID

✓ Method: **DELETE**

✓ Authentication:  
Kinvey + token

✓ User / pass:  
guest / guest

✓ Body: (empty)





# Kinvey and Postman: Edit an Existing Post

✔ URL: [https://baas.kinvey.com/appdata/{app\\_id}/posts/id](https://baas.kinvey.com/appdata/{app_id}/posts/id)

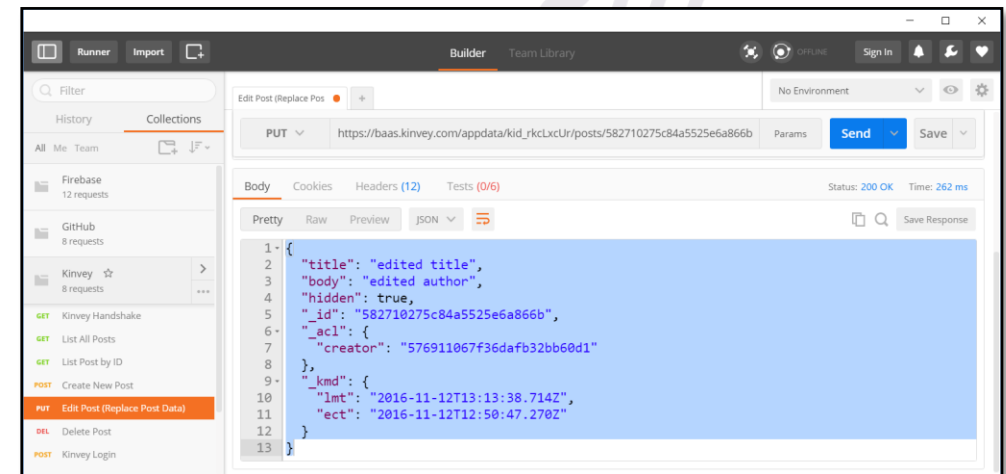
✔ Choose an existing post's ID

✔ Method: **PUT**

✔ Authentication: Kinvey + token

✔ User / pass: guest / guest

✔ Body (JSON):



```
{"title": "edited title", "body": "edited author", "hidden": true}
```

The background of the slide is a dark blue, blurred image of a classroom. In the foreground, the backs of several students' heads are visible as they sit at desks. In the background, a whiteboard is mounted on a wall, and the silhouettes of other students can be seen. The overall atmosphere is that of a lecture or workshop setting.

# Live Exercises



# Summary

- Relational Databases
- Non Relational Databases
- BaaS
- Firebase is **JSON-based** cloud database
- (mBaaS) with **REST API**
- Kinvey is **collection-based** cloud database (mBaaS)





# Questions?





# License

- ✔ This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- ✔ Unauthorized copy, reproduction or use is illegal
- ✔ © Kingsland University – <https://kingslanduniversity.com>





THANK YOU

