Serverless Backend as a Service (BaaS)

ALLOWS YOU TO:

- Provide data for client (e.g. an app: web, native, Wechat Mini Program)
- Provide service to customers (e.g. sms, payment)

BaaS (Backend as a Service)

BaaS is a cloud service model that comes with:

- Server
- Database
- APIs

BaaS + Frontend = SaaS, aka App

Do we still need a **backend developer**? 🤔

	Backend Developer	BaaS
APIs	Develops functional APIs	Offers commonly-used APIs
Database	Manages the database they built themselves	Developers can create/update tables w/o building the backend themselves
Read/Write Data	Builds a read/write API	Tables come with a built-in read/write API
User Log-in	Builds a log-in API (usually takes 2-3 weeks)	Comes with a log-in support API
WeChat/SNS Notifications	Works with WeChat/SNS platforms to build the API	Comes with a notification support API

Data and Schema

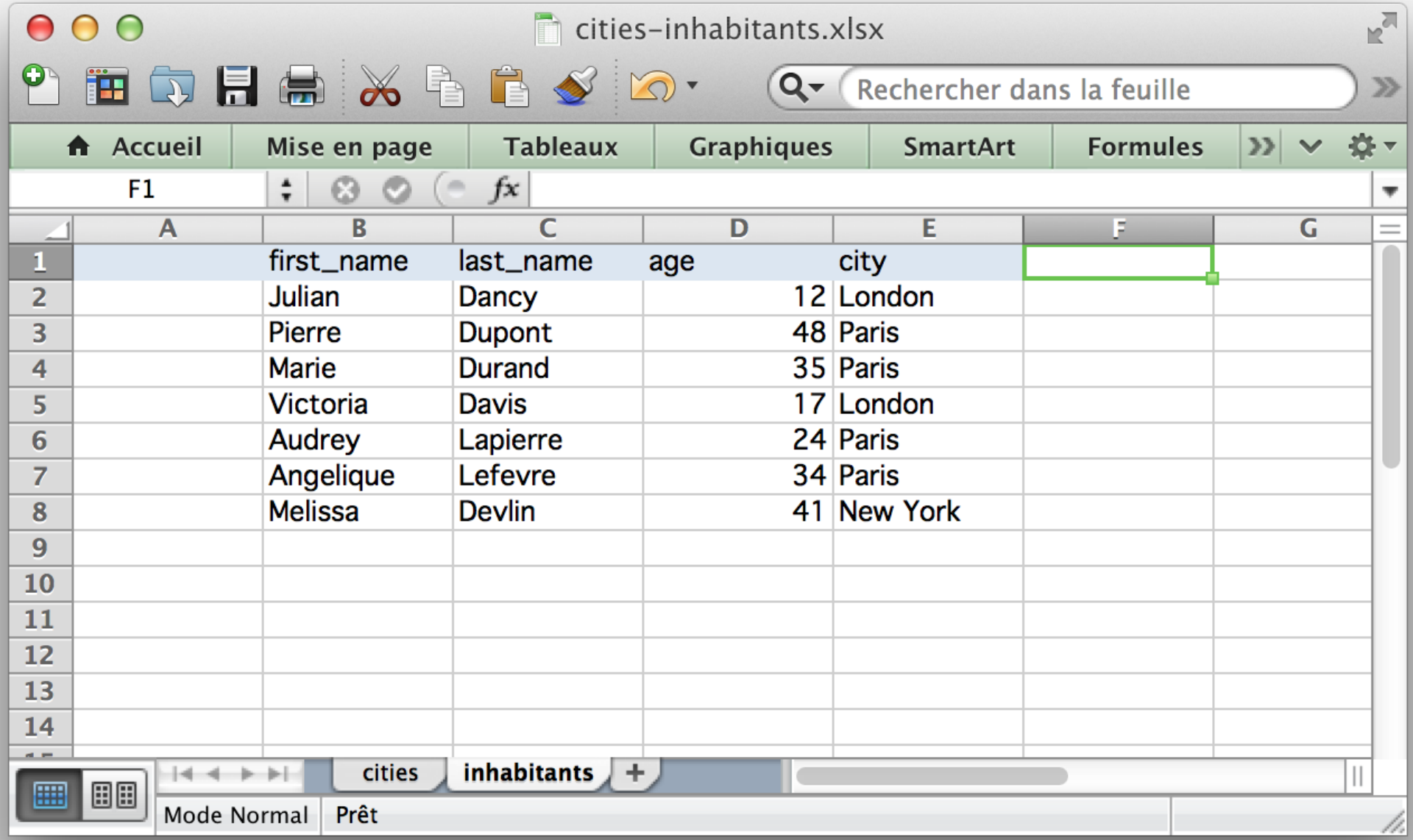
How is data stored?

Excel

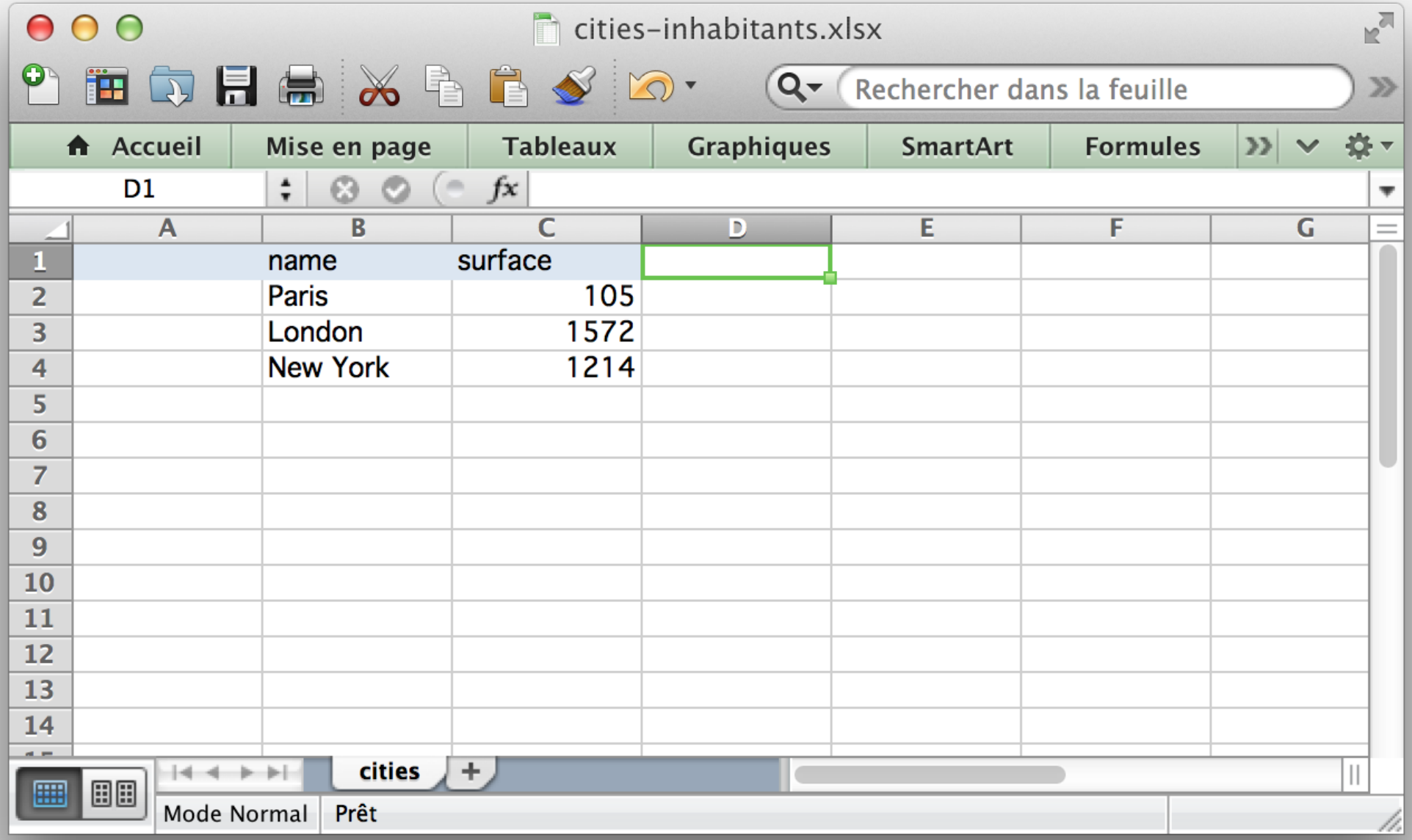
Let's start with something we all know

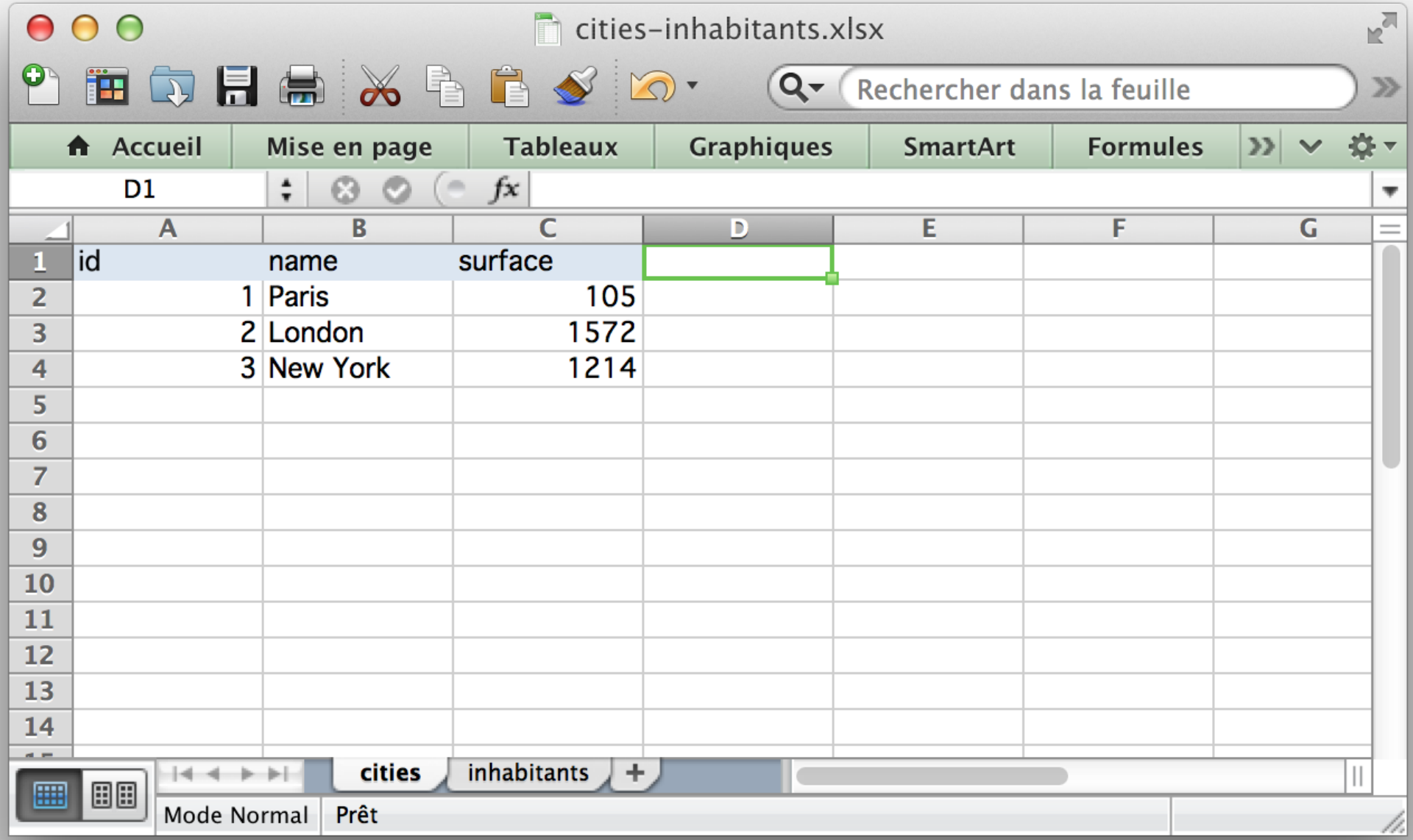
Example

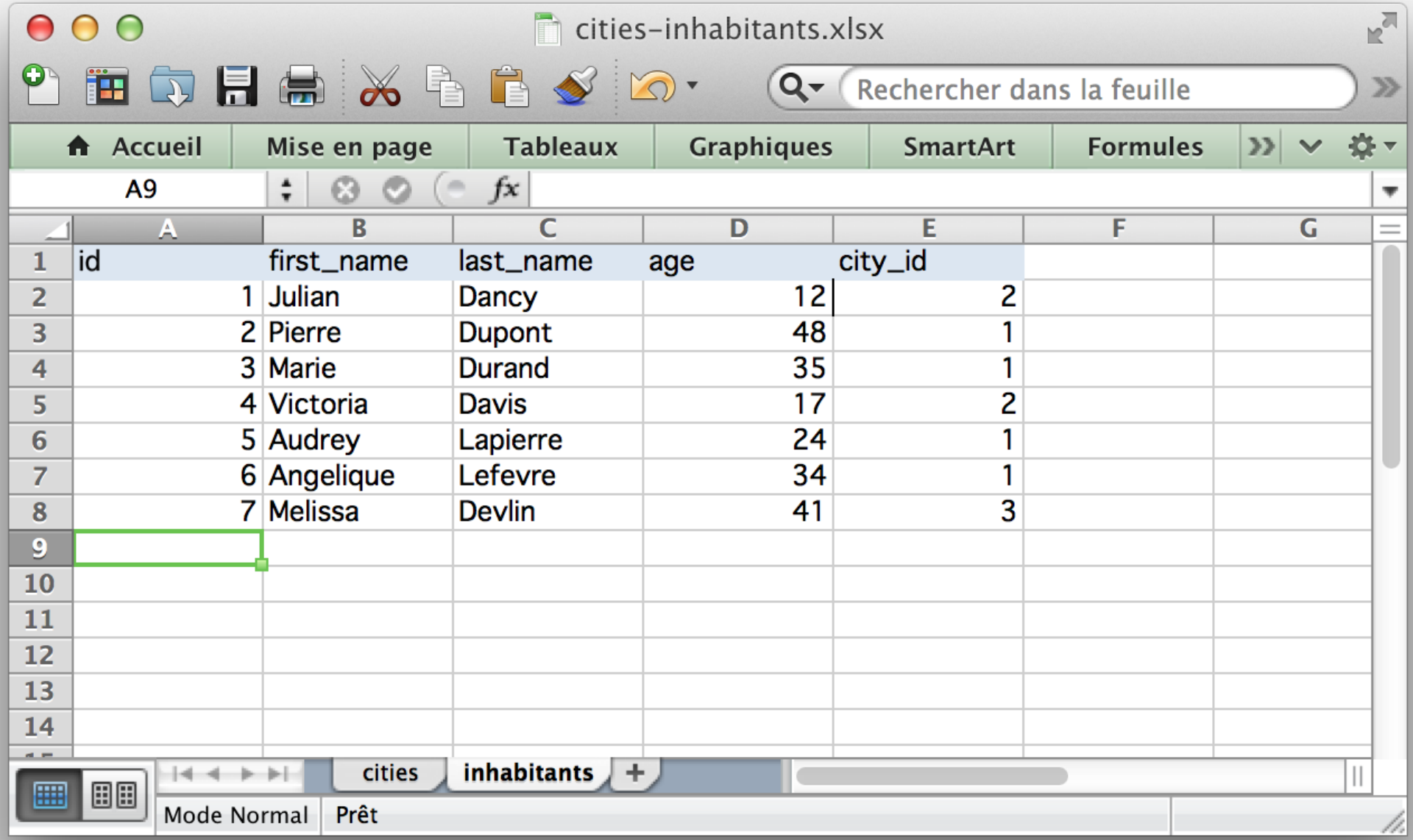
Let's store **cities** and their **inhabitants** using Excel. How would you do it?



	A	B	C	D	E	F	G
1		first_name	last_name	age	city		
2		Julian	Dancy	12	London		
3		Pierre	Dupont	48	Paris		
4		Marie	Durand	35	Paris		
5		Victoria	Davis	17	London		
6		Audrey	Lapierre	24	Paris		
7		Angelique	Lefevre	34	Paris		
8		Melissa	Devlin	41	New York		
9							
10							
11							
12							
13							
14							


cities-inhabitants.xlsx
Rechercher dans la feuille
Accueil
Mise en page
Tableaux
Graphiques
SmartArt
Formules
D1
A
B
C
D
E
F
G
1 name surface
2 Paris 105
3 London 1572
4 New York 1214
5
6
7
8
9
10
11
12
13
14
cities
Mode Normal
Prêt


cities-inhabitants.xlsx
Rechercher dans la feuille
Accueil
Mise en page
Tableaux
Graphiques
SmartArt
Formules
D1
A
B
C
D
E
F
G
1 id name surface
2 1 Paris 105
3 2 London 1572
4 3 New York 1214
5
6
7
8
9
10
11
12
13
14
cities
inhabitants
Mode Normal
Prêt

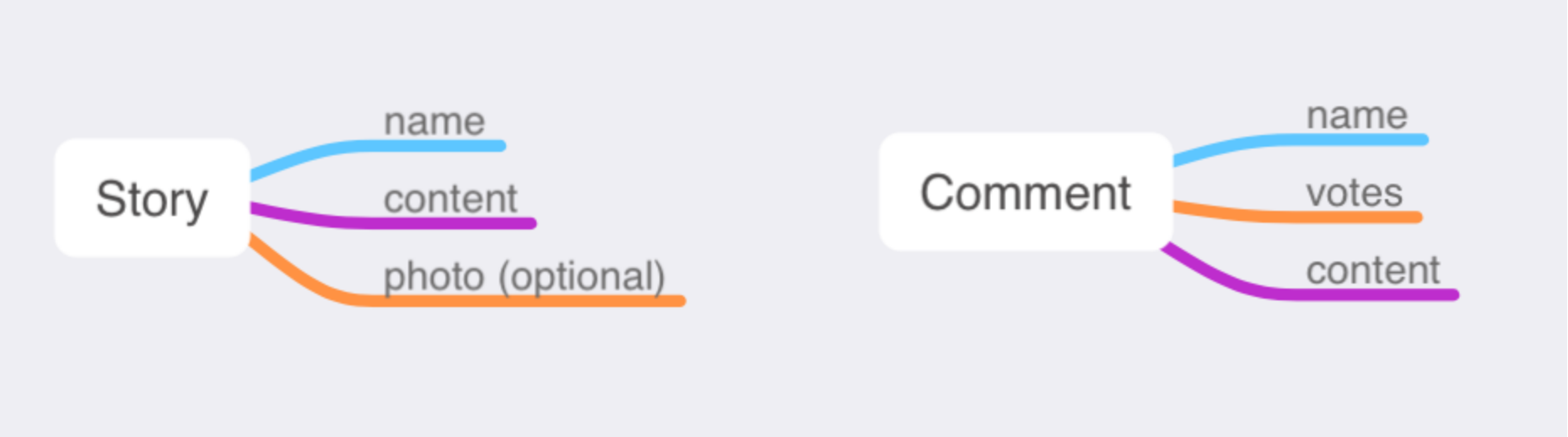


	A	B	C	D	E	F	G
1	id	first_name	last_name	age	city_id		
2	1	Julian	Dancy	12	2		
3	2	Pierre	Dupont	48	1		
4	3	Marie	Durand	35	1		
5	4	Victoria	Davis	17	2		
6	5	Audrey	Lapierre	24	1		
7	6	Angelique	Lefevre	34	1		
8	7	Melissa	Devlin	41	3		
9							
10							
11							
12							
13							
14							

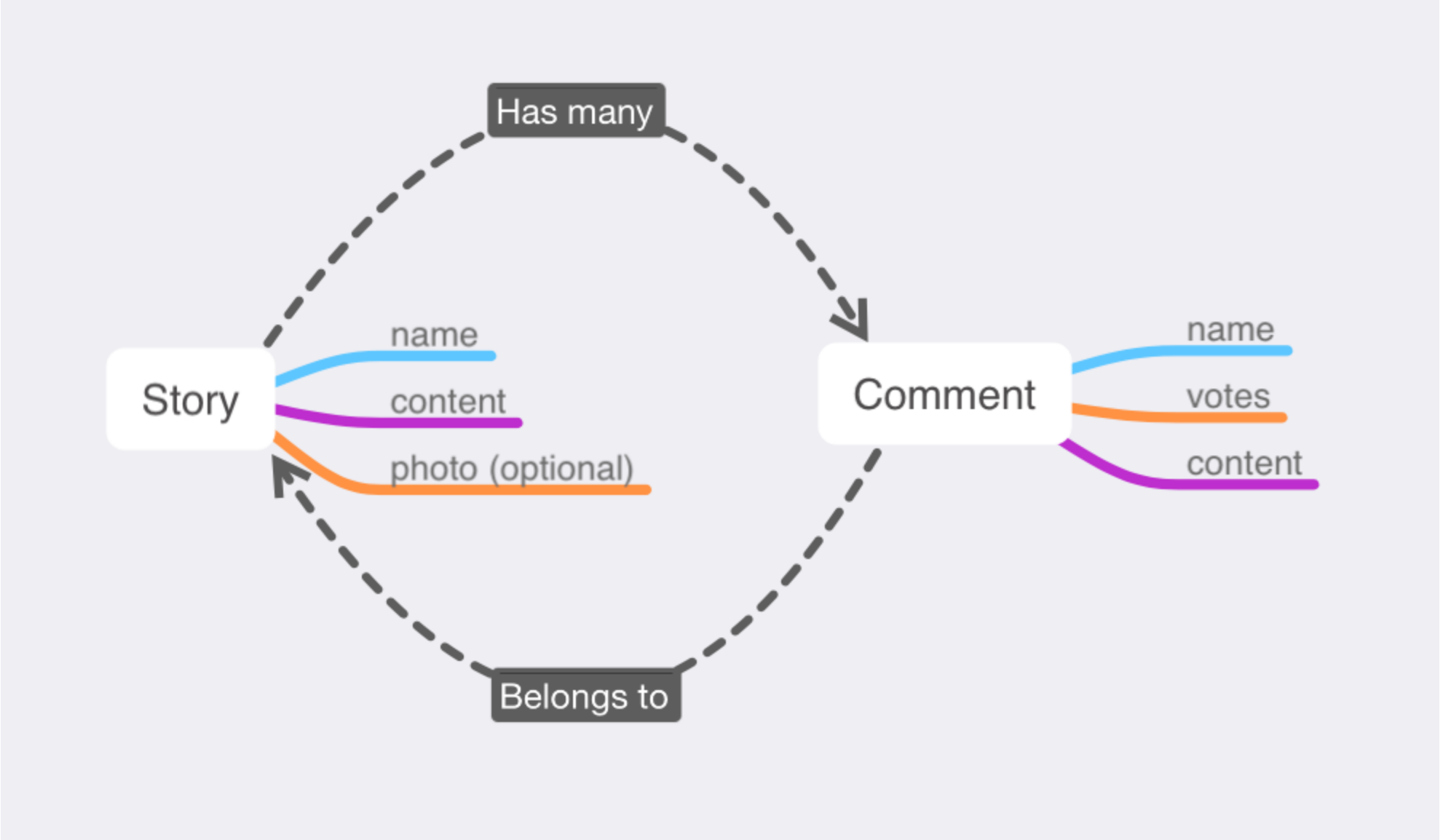
1:n relation (one to many)

An inhabitant **belongs to** one city (or has one city)

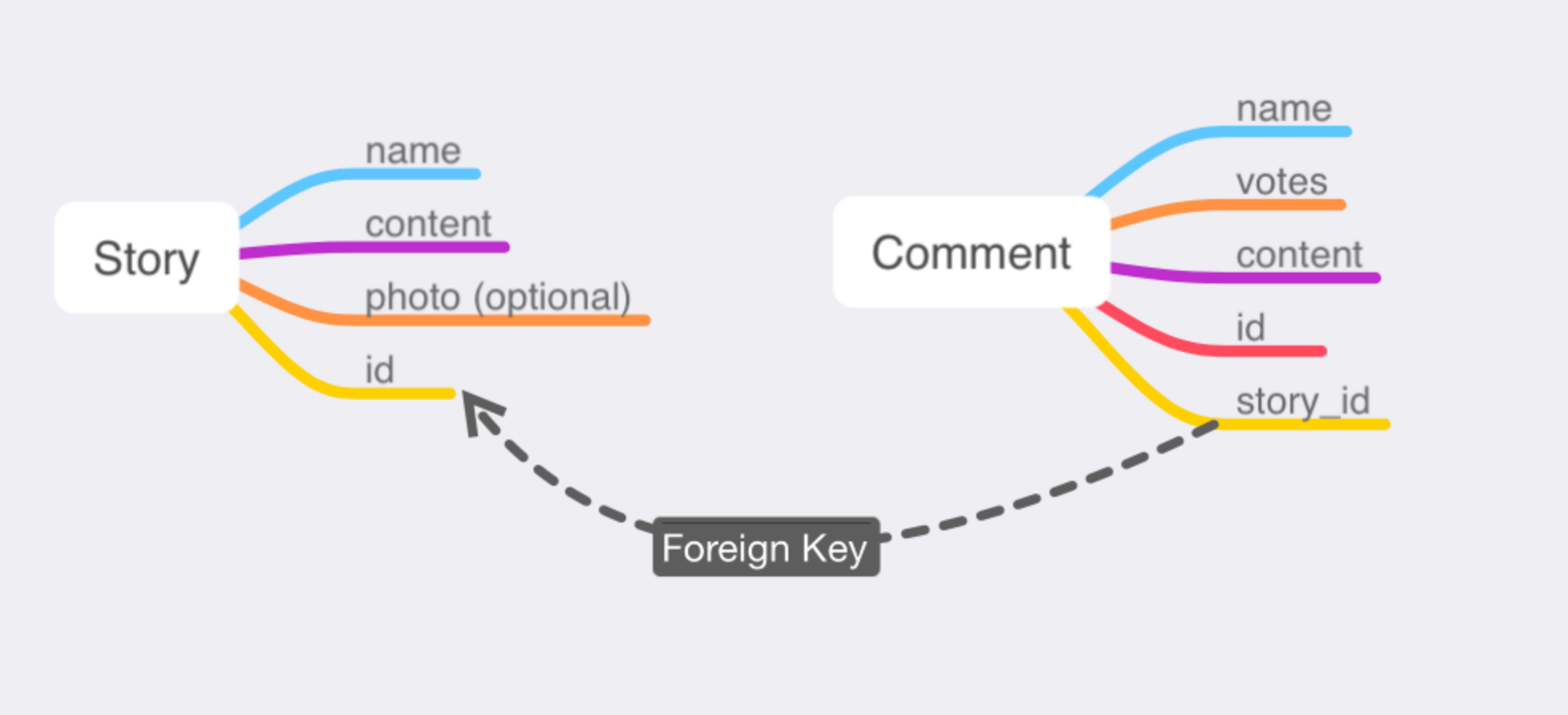
Toutiao Data Schema



Relationship



IDs → Foreign Key



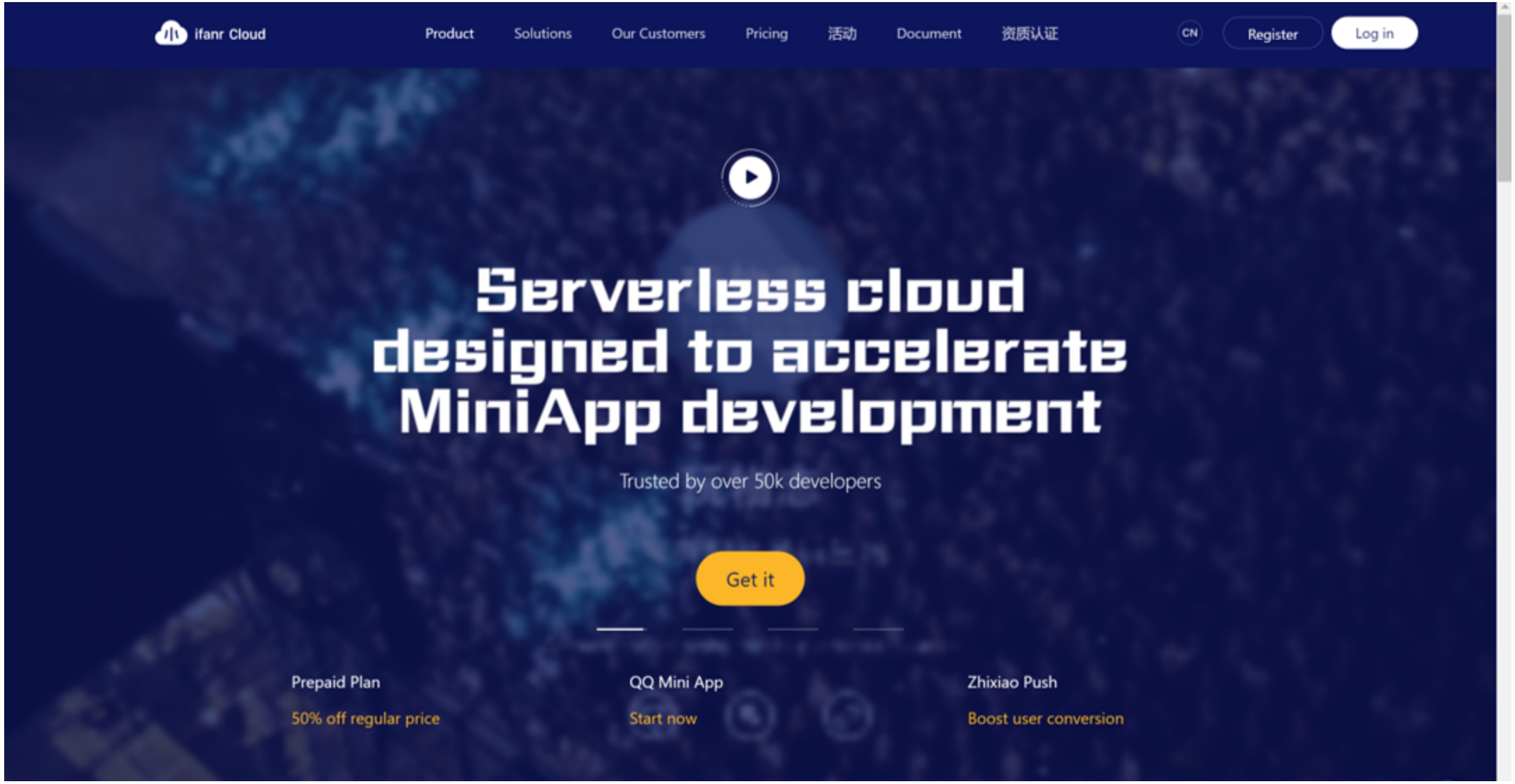
Agenda

You will build the same backend for your Toutiao mini-program with an **SDK** (instead of an API)

> An SDK (software development kit) is a collection of software development tools in one installable package

Let's set up the backend!

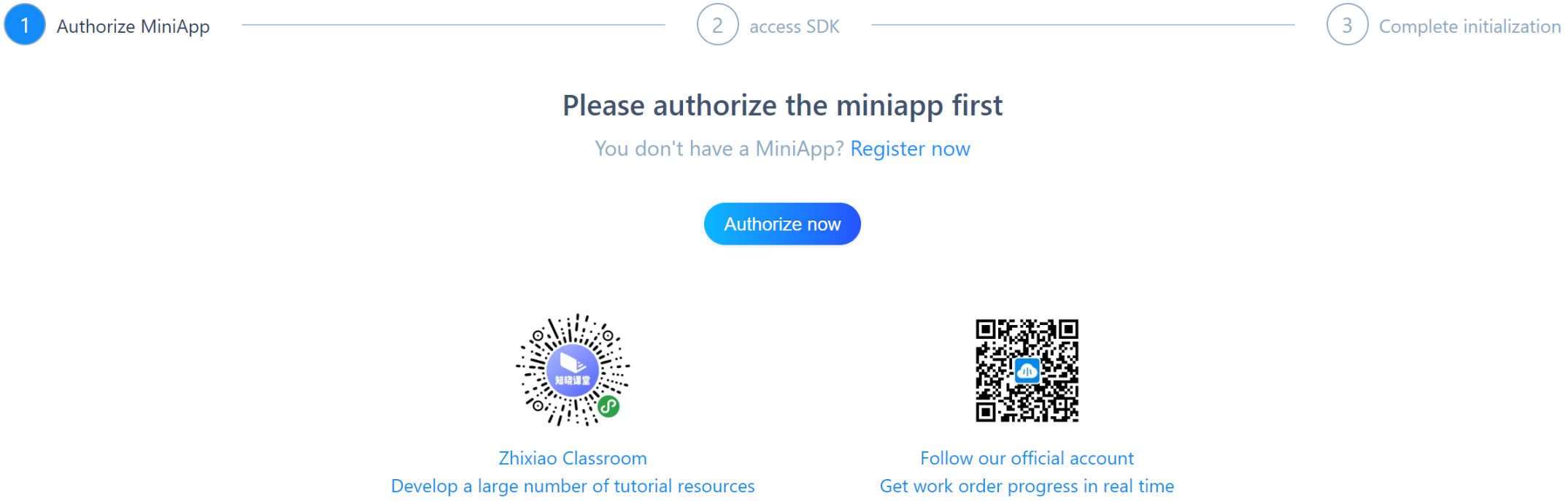
We'll use ifanr moving forward



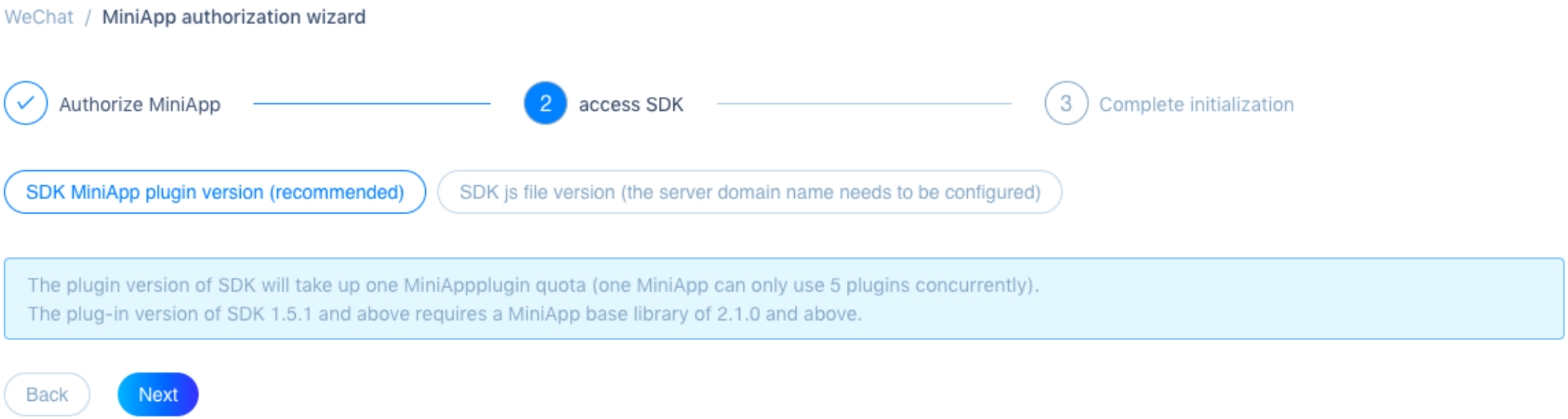
Register an account now

Live Code 1: Setting Up BaaS 👩‍💻

- Install the SDK
- Create tables
- Add data



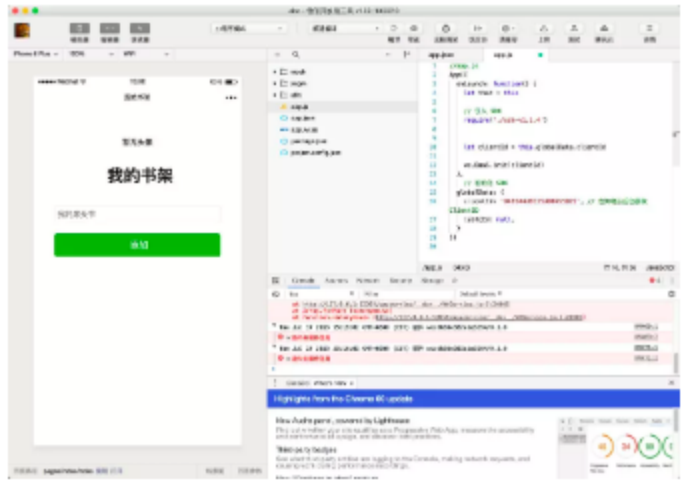
To link the BaaS to our mini-program



To access the plugin SDK

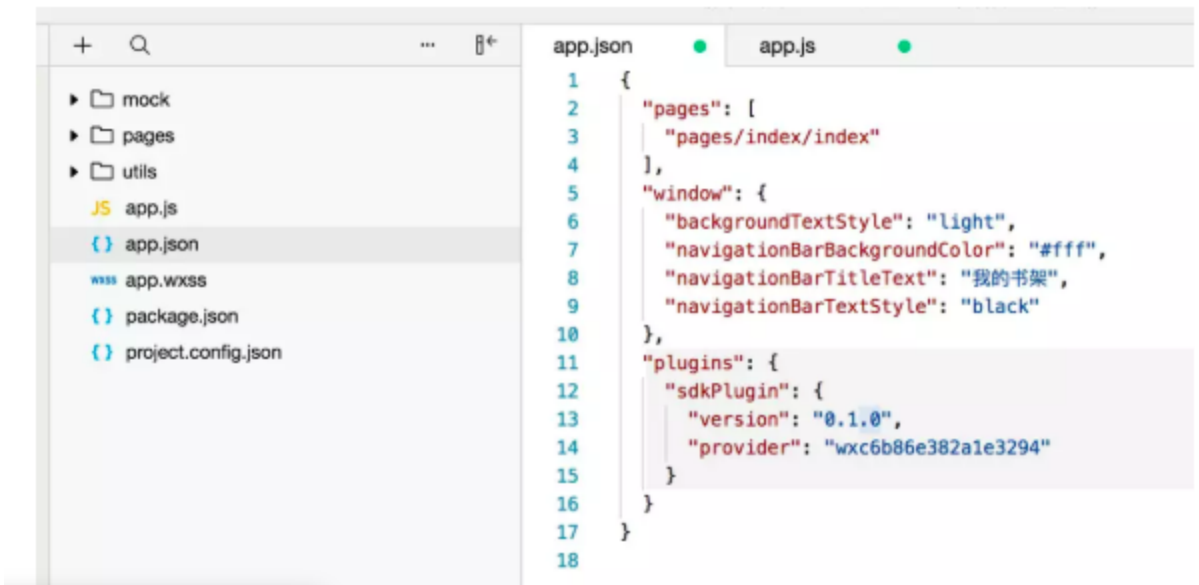
Authorize MiniApp — access SDK — 3 Complete initialization

1. Download WeChat Developer Tools, open the MiniApp project and enter the editor




```
AppID: wx6e1e01322fd9850e
```

add a reference statement of plugin in app.json

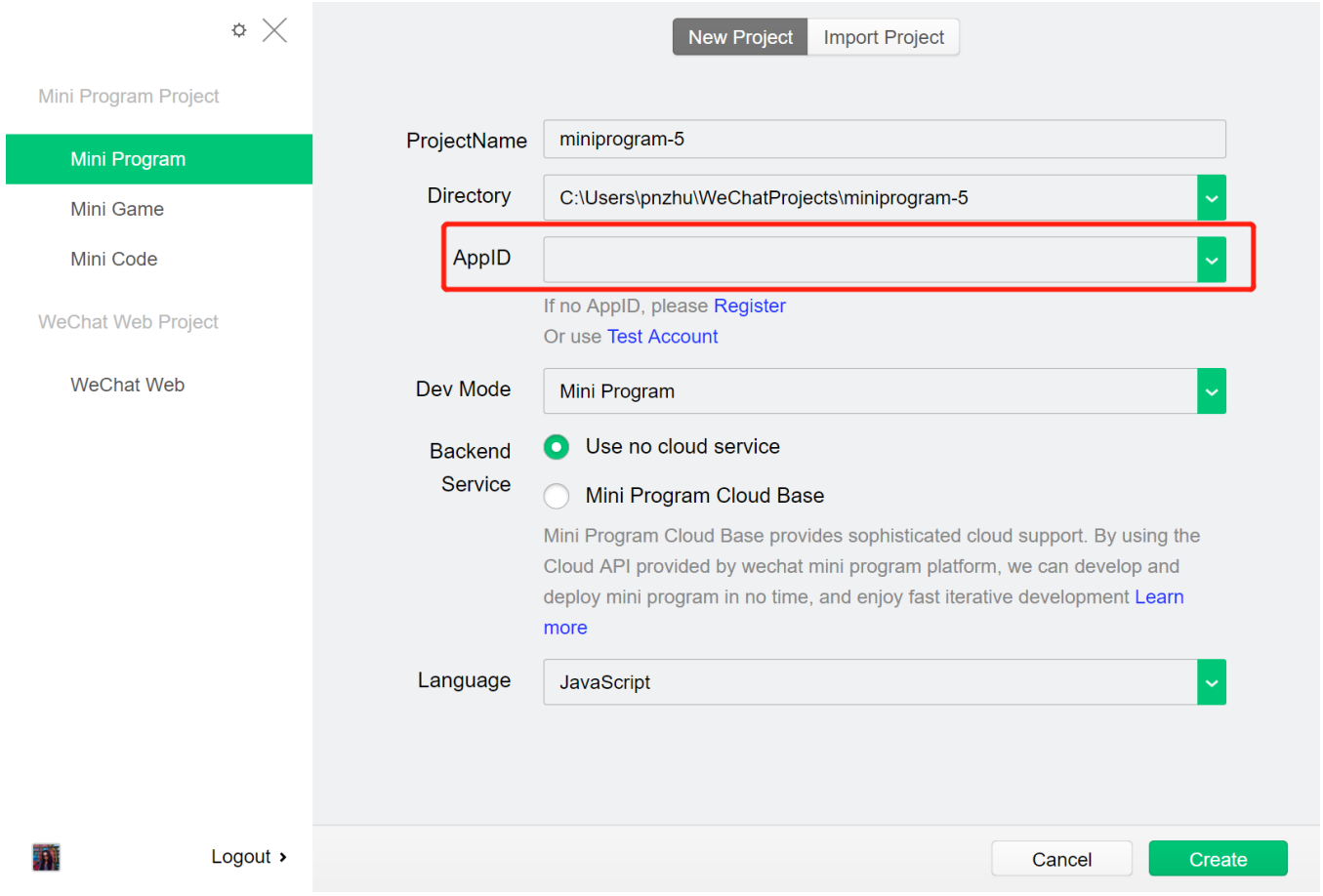



```
"plugins": {
    "sdkPlugin": {
      "version": "2.4.0",
      "provider": "wxc6b86e382a1e3294"
    }
}
```

Installing the SDK

As simple as following a set of instructions

1. Applying AppID



Tip: If you're already using a Test or Tourist AppID, you can modify the AppID in your mini-program's 'Details' tab

2. Referencing the plugin SDK

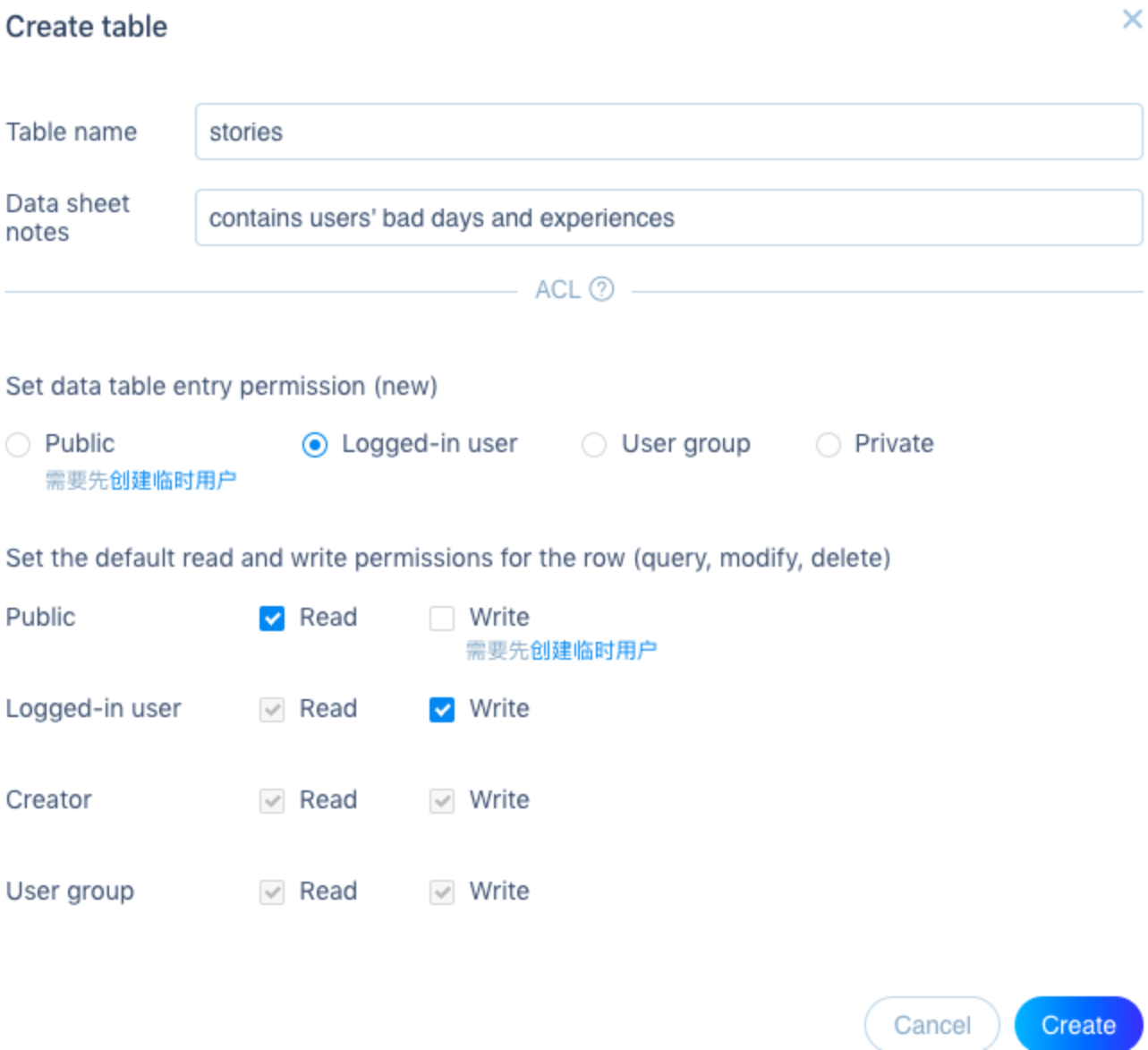
```
// app.json
//... other configurations
"plugins": {
  "sdkPlugin": {
    "version": "2.3.0",
    "provider": "wxc6b86e382a1e3294" //This BaaS id doesn't change
  }
}
```

3. Configuring the SDK

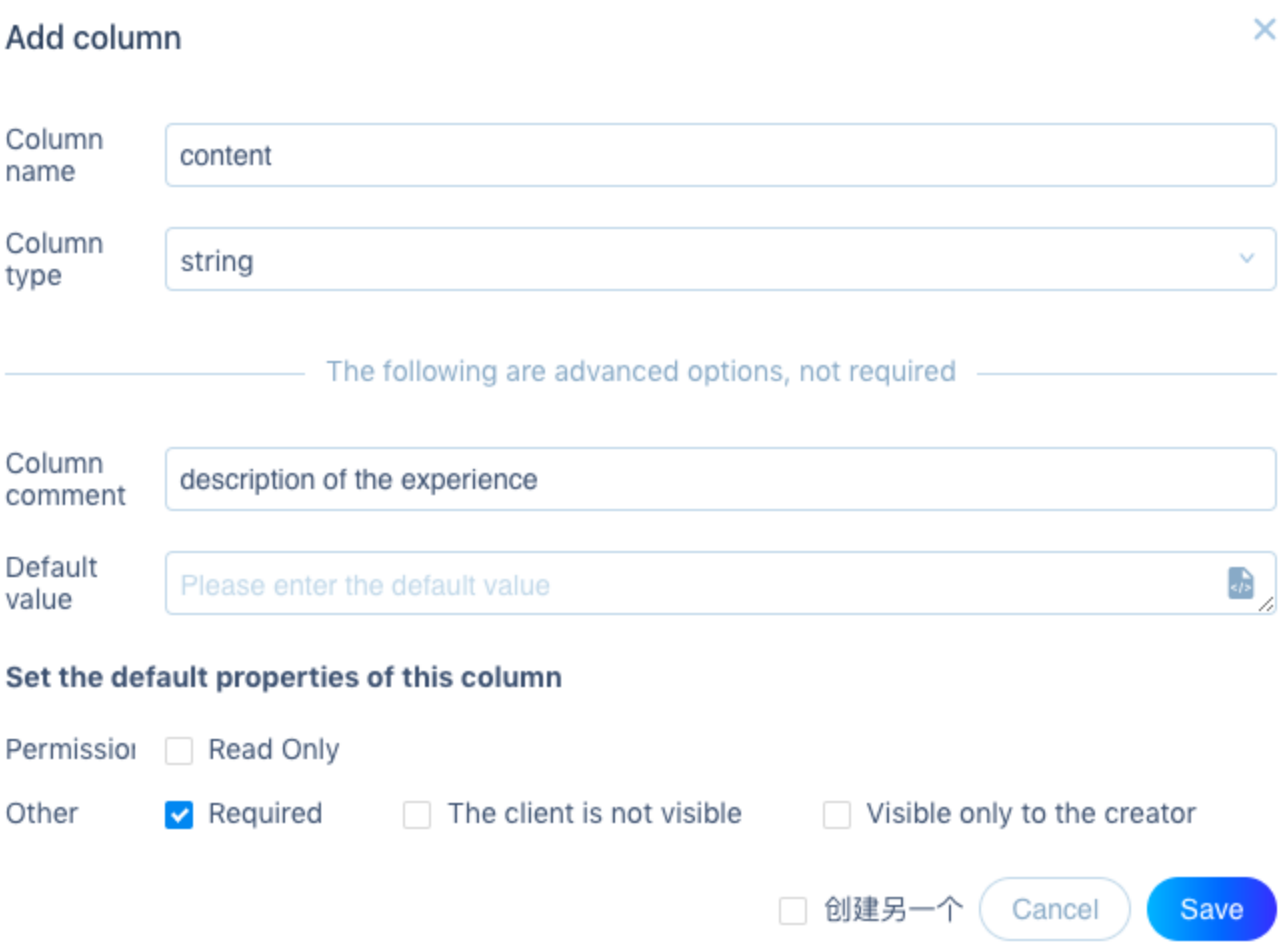
```
// app.js
App({
  onLaunch: function() {
    wx.BaaS = requirePlugin('sdkPlugin')
    // enables login, payment, and other features
    wx.BaaS.wxExtend(wx.login,
     wx.getUserInfo,
     wx.requestPayment)

    const clientID = 'c1e7a280f6d8c8646756' // The ClientID received by the backend
    wx.BaaS.init(clientID)


    // Login as anonymous user
    wx.BaaS.auth.anonymousLogin().then(user => {
      console.log(user)
    }).catch(err => {
      // HError
    })
  },
})
```

Creating Tables

Creating a **table**

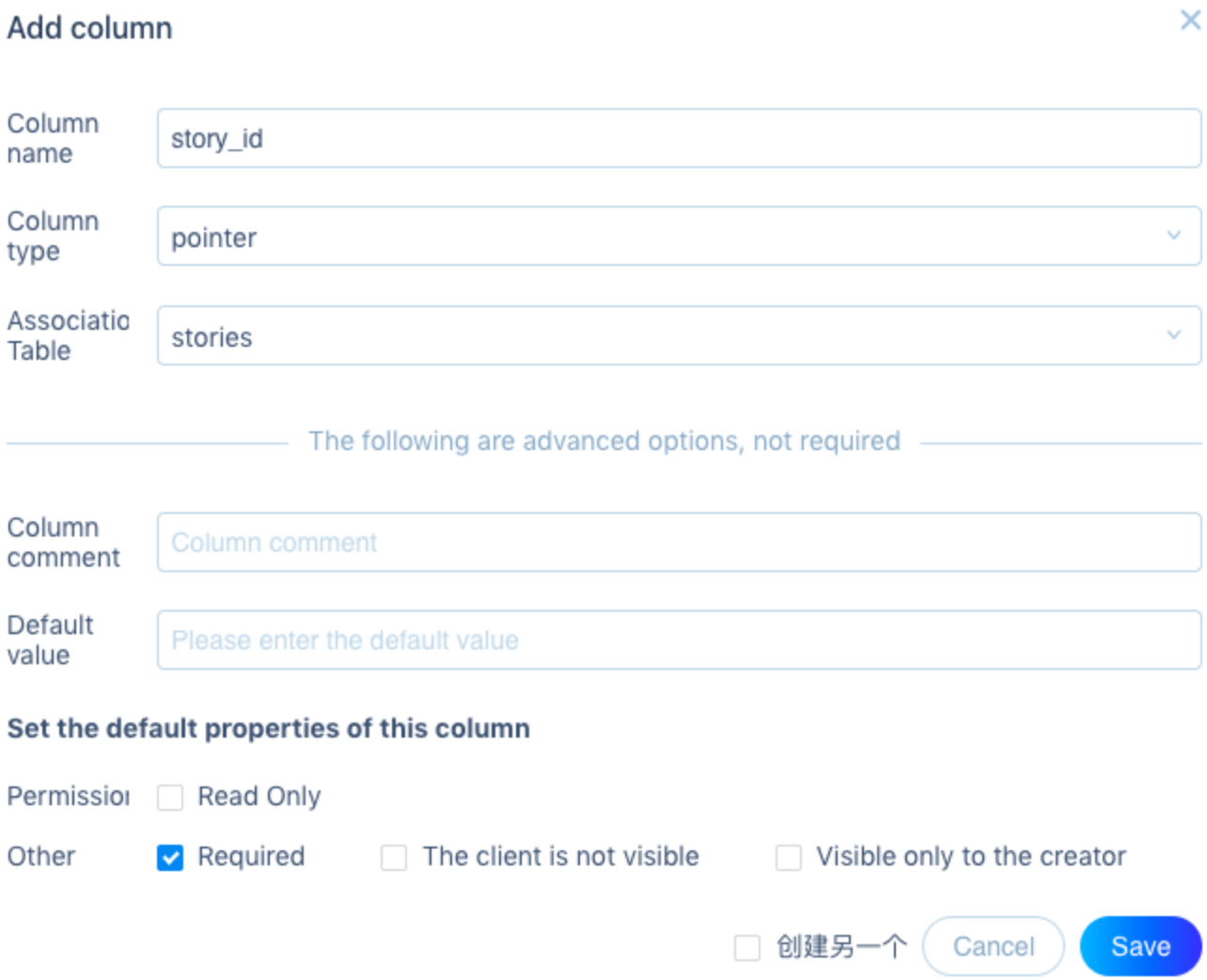


Adding fields or **columns** to a table while setting their **data type** and **properties*



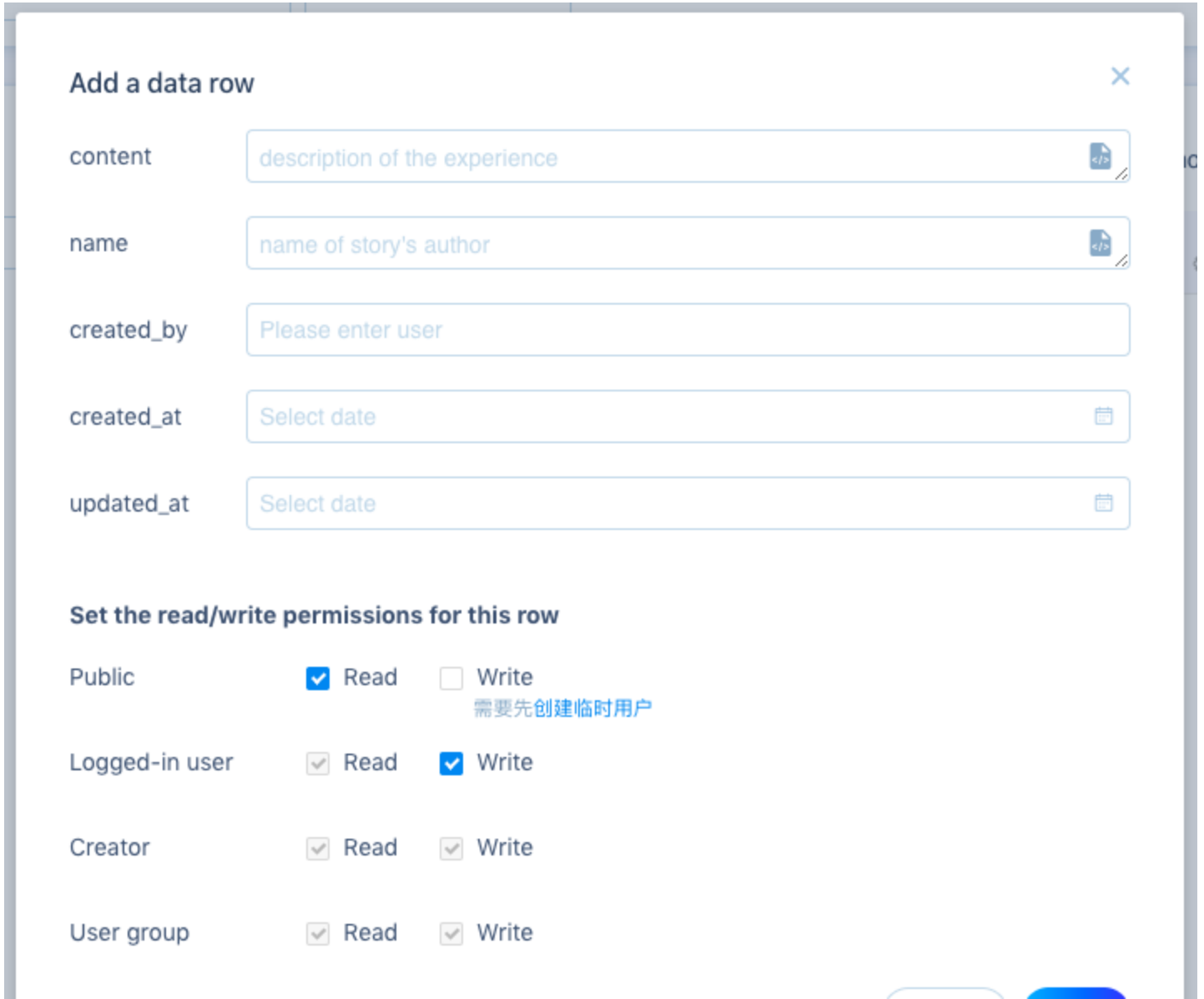
Adding a **foreign key** (called **pointer** in minapp) field to the **child table**

*e.g. A story can have multiple comments

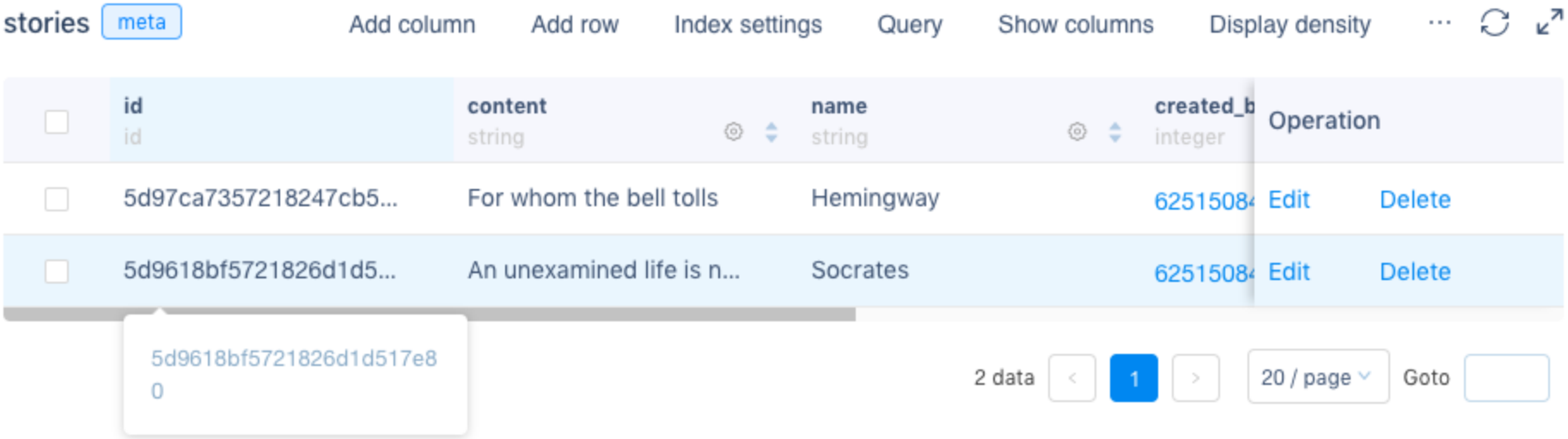


Adding Data

Adding **rows**


Add a data row
content
description of the experience
name
name of story's author
created_by
Please enter user
created_at
Select date
updated_at
Select date
Set the read/write permissions for this row
Public
Read
Write
需要先创建临时用户
Logged-in user
Read
Write
Creator
Read
Write
User group
Read
Write

Adding data to a **child row**



Your turn!

EXERCISE 1: CREATE THE BACKEND 💪

Applying the backend!

Reading Data

Implementing the **Read** feature of CRUD with the BaaS **SDK** instead of an API

Live Code 2: Read (all) 👩‍💻

With the `find` function in the SDK, fetching data is just these few lines:

```
// index.js, in onLoad function
let tableName = 'stories'

let Story = new wx.BaaS.TableObject(tableName)

Story.find().then(dosomething)
```

`dosomething` should be replaced by the function that handles the **request response**

Your turn!

EXERCISE 2: READ ALL STORIES 💪

Live Code 3: Read (one) 👩‍💻

Use the `get` function in the SDK

```
// show.js, in onLoad function
onLoad: function (options) {

  let tableName = 'stories'

  let Story = new wx.BaaS.TableObject(tableName)

  let recordID = options.id // e.g. '59897882ff650c0477f00485'

  Story.get(recordID).then(dosomething)
},
```

Your turn!

EXERCISE 3: READ ONE STORY 💪

Not challenging enough?

Live Code 4: Read (all for one) 👩‍💻

Define the **child table**

```
// show.js, in onLoad function
let tableComments = 'comments'
let Comment = new wx.BaaS.TableObject(tableComments)
```

How do we get specific data? This brings us to **queries**

Querying

What is a query? Think of it as a search function in your table

You can search text, numbers, booleans... any type with exact matching or operations like >, <=, != ...

Chances are you've been using that without knowing. Anytime you use a search bar! You are querying ;-)

Define a `query`

```
//... Define Comment in onLoad function as above

// Instantiate the query object
let query = new wx.BaaS.Query()

// Get the story you retrieved earlier from page's data
let story = this.data.story

// Set conditions to a query...
query.compare('story_id', "=", story.id)

// Run the query
Comment.setQuery(query).find().then(dosomething)
```

Lots of queries are possible, including chaining them!

Your turn!

EXERCISE 4: READ COMMENTS FOR ONE STORY 💪

Happy 1st day of BaaS!