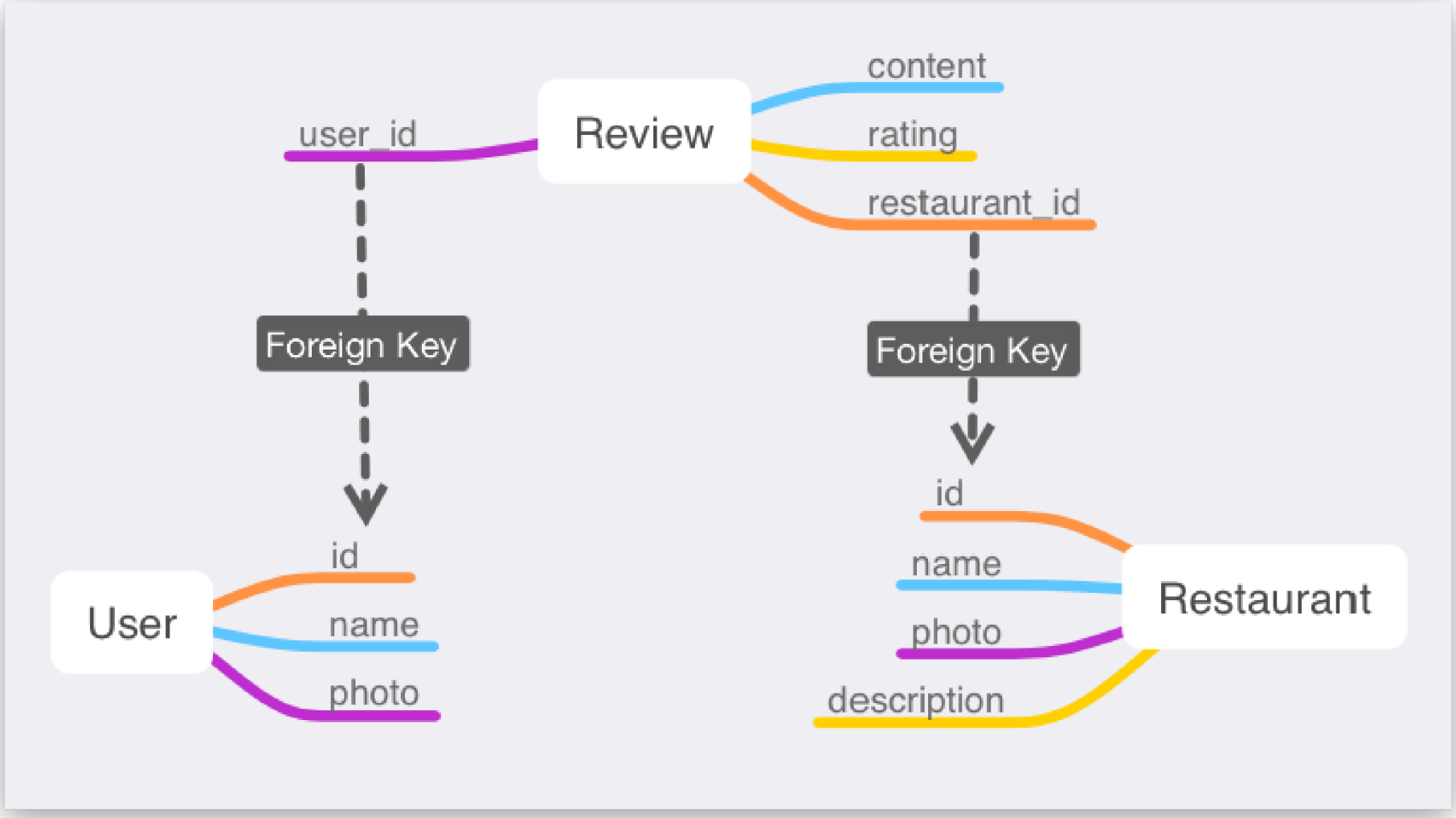
Advanced Schema (continued)

Let's Recap

Many-to-many relationship



On Tuesday, we learnt the most common action of the C**R**UD - **READ**

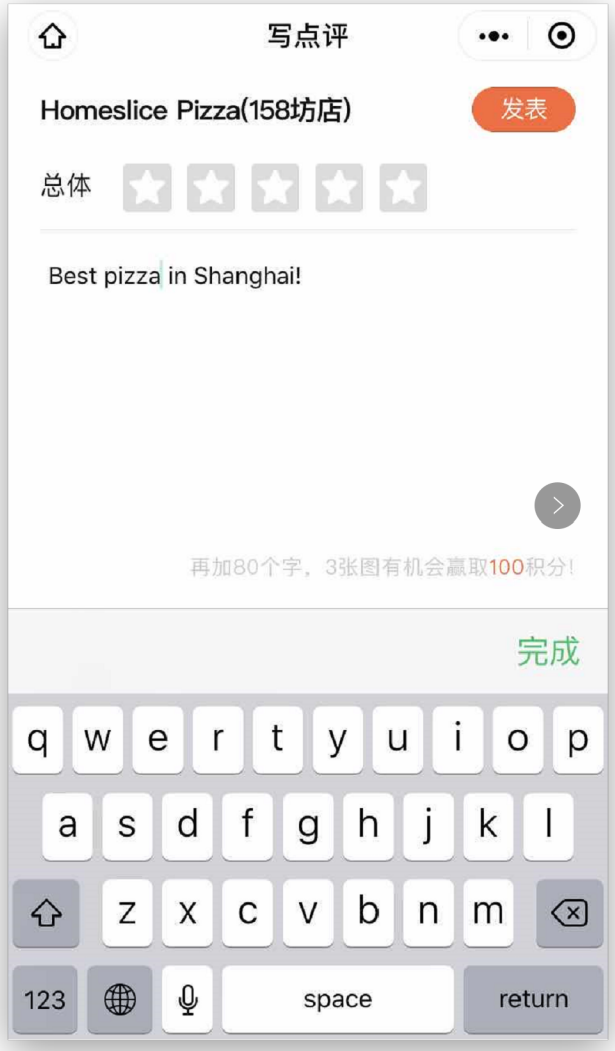
- As a user, I can view all
- As a user, I can view one
- As a user, I can view all for one

On Thursday, we created a fully-functioning **sign-up** and **login** system

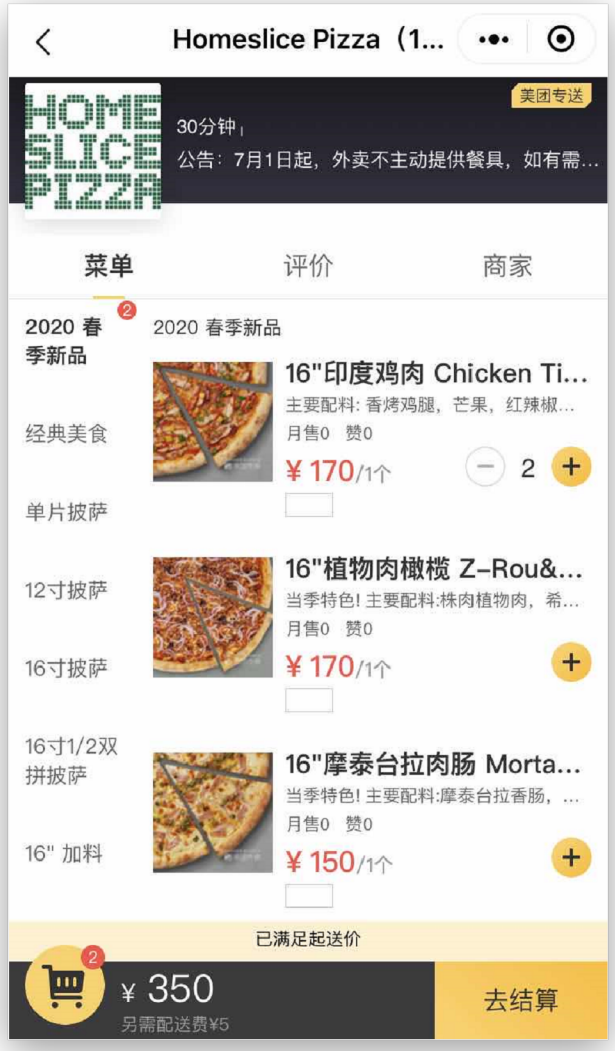
- As a user, I can sign up and log in

This afternoon, we will build a more **functional** mini-program

As a logged-in user, I can now post reviews



As a logged-in user, I can now place an order



CRUD - **CREATE**

A 3-step process with the SDK

1. Create an empty record

```
// define table object
let review = Review.create()
```

2. Give the record some data

```
review.set(data)
```

3. Store data to the backend

```
review.save() //.then(dosomething) if needed
```

We can implement the **CREATE** function for different use cases

Live Code 1: Posting Reviews 👩‍💻

Reviews form

```
<!-- at bottom of show.wxml -->
<form bindsubmit="createReview" wx:if="{{currentUser}}">
  <view class="section">
    <input name="content" placeholder="put down your thoughts"/>
  </view>
  <button formType="submit" class="footer-btn">Submit</button>
</form>

<button bindtap="showUserPage" class="section" wx:else>
  Please Log In to Review
</button>
```

Bonus: Using WeChat Components

Scroll `picker`

```
<!-- in review form of show.wxml -->
<picker mode="selector" range="{{ [1, 2, 3, 4, 5] }}" bindchange="onRate">Rating</picker>
```

`bindchange` event handler

```
// show.js
onRate: function(event) {
    console.log(event)
}
```

Create review

```
// show.js
createReview: function(event) {
    // ...
    let Review = new wx.BaaS.TableObject('reviews')

    let review = Review.create()

    let data = {
        // review's content and rating
    }

    review.set(data).save().then(dosomething)
},
```

Your turn!

EXERCISE 1: POST REVIEWS 💪

Live Code 2: Placing Orders 👩‍💻

Two additional tables: `meals` and `orders`

Which of the two is the **join** table? 🤔

A user can view all `meals` for a restaurant

Sounds familiar? Similar to listing a restaurant's reviews!

Except now, a user can place an order for a meal

```
<!-- in wx:for meals loop of show.wxml -->
<button data-id="{{meal}}" bindtap="submitOrder">Order</button>
```

Create order

```
// show.js
submitOrder: function(event) {
    // ...
    let Order = new wx.BaaS.TableObject('orders')

    let data = {
        // meal_id and current user_id
    }

    Order.create().set(data).save().then(dosomething)
    // relaunch and swtich to user profile to display orders
},
```

Display meals ordered

For this, we need to expand the meals table so we can include the meal's `name`, `price` and `photo` in the order!

```
// user.js
onLoad: function (options) {
    let page = this
    wx.BaaS.auth.getCurrentUser().then(function(res) {
      page.setData({
        currentUser: res
      })
      // ⚠
      let Order = new wx.BaaS.TableObject('orders')
      let query = new wx.BaaS.Query()
      let currentUser = page.data.currentUser.id.toString()
      query.compare('user_id', '=', currentUser)
      Order.setQuery(query).expand(['meal_id']).find().then(function(res) {
        console.log('ressin', res)
        page.setData({
          orders: res.data.objects
        })
      })
    })
  },
```

⚠ Subtle but important: we save the `order` after the currentUser responds **success**, in its handler function

Your turn!

EXERCISE 2: PLACE ORDERS 💪

Happy Weekend!