11 Making RESTful HTTP requests

In this lab we're going to exercise making fetches at low-level. We'll use the http Dart library's post() method to purchase tickets in Checkout and then use get() to read from the RESTful API server in Ticket.

Purchasing the movie tickets

Our mission for this section is to actually submit a purchase of N seats to our server and get back a list of N ticket numbers.

1. Edit checkout.dart. Review the _checkout() method. This is where we've summarized and JSON encoded the purchase/cart data. It's also where we should be POSTing the purchase to the server.

You have purchaseJson, a serialized JSON string that represents our purchase. Let's send it to the server.

2. Write a HTTP request to POST this JSON string to "http://localhost:3008/api/buyTickets".
   Here are some hints to make your purchase POST successful.
   - Don't forget to import 'package:http/http.dart';
   - You'll need to call post() and pass into it ...
     - the Uri,
     - a body which is the purchaseJson string,
     - and this headers: {'Content-Type': 'application/json'}
   - The post() returns a Future<Response>. This means you'll have to process the response with a .then() or await it.
   - Your response will have a body property. (ie. theResponse.body) which will be a list of the ticket numbers.
   - Here's how to process it:

     ```
     var ticketNumbers =
           (json.decode(res.body) as List).cast<Map<String,dynamic>>();
     ```
   - If your request fails inexplicably, replace the "http://localhost:3008" with getBaseUrl() in the repository.dart file. This library method replaces the first part of the url depending on the platform, solving some of the security problems.
   - Use the IDE's debugger and some print() statements to do problem determination.

3. Run and test this, tuning until you get a successful response that has one ticket number for each seat you purchased.

Getting the tickets into state

We now have one or more ticket numbers. that should be viewed in the tickets scene. So of course, let's save those ticket numbers in rawState before we Navigate away.

4. Put this in your code just before you Navigator.pushNamed():

```
rawState.set("ticketNumbers", ticketNumbers);
```

Next, we'll read those in the Ticket scene.

5. Edit ticket.dart. Just inside the State class, read those values from rawState:

```
List<Map<String, dynamic>> ticketNumbers =
      rawState.get<List<Map<String, dynamic>>>("ticketNumbers");
```

6. For now, just print them out to make sure you got them transferred alright.

Getting the details of each ticket

We're almost finished. We've purchased the tickets and gotten back a list of ticket numbers. We just need to gather data about each ticket from the API server and display the details.

The idea will be for the customer to show up at the theater with their phone, show their phone with the details and this will serve as their ticket. We will be paperless. We want to tell the user what movie they're seeing, which theater, and what time it starts. We're going to use each ticket number to look up that information.

7. You can iterate the list of ticket numbers any way you'd like but for each ticket we're going to want to make a GET request from the server at http://localhost:3008/api/reservations/<ticket_number>

Note that since each request is a simple GET there's no need for a body nor headers. It's still going to be a Future<Response> whose Response.body will need to be json.decoded(). When each response comes back it will have all the info we'll need to display a ticket.

8. Create one widget for each ticket with a Column(). The column's children are Text() widgets:
    - "We're looking forward to seeing you. Show this to your host when you arrive. This is your ticket."
    - The film title,
    - Theater name,
    - Showing date and time,
    - Table number
    - Seat number
    - Ticket number

    Hint: To show the film title, go Text(ticket["film"]["title"] ?? "(No title)").

9. Run and test.

10. You'll almost certainly overflow the screen. Wrap your whole Ticket widget with a SingleChildScrollView in order to scroll with your finger.

Congratulations! You're finished.

11. Bonus! If you have extra time, add a movie poster and price paid to each ticket.