The backend

Saving and retrieving data in an app

for UNC COMP 523: Software Engineering Laboratory
on Monday, September 23rd, 2019
by Jeff Terrell
The problem

- All apps need data
- Where does the data live?
  - On-device
  - In a centralized location (e.g. “in the cloud”)
- The centralized location is the authoritative “source of truth”
- This is called “the backend” or “server”
- (“The frontend” or “client” is the app code running on the user’s device)
Backend components

• **The database** stores the data durably
  • “Durably” means “surviving a reboot”, i.e. using disks
• **The “API server”** interfaces between clients and the database
  • Why? Separation of concerns.
  • Handling HTTP requests
  • Doing authentication and authorization checks
  • Limit the types of interactions (untrusted) clients can have with data
  • Easier to develop as a separate program than as part of database
• Backend programmers *use* a database but *create* an API server
The API: application programming interface

- Like a “user interface” (the screens that users interact with), but for programs
- The frontend is a program that uses the API
- The API defines ways that the frontend code can save and retrieve data
- These ways are called “endpoints”; an API is a set of endpoints
- You must define an API for your app and your data
- Modern APIs usually use HTTP
HTTP: the hypertext transfer protocol

- Browsers use HTTP almost exclusively
- Two types of HTTP messages: requests and responses
- Requests have a method, a URL, and maybe parameters or a body
- Responses have a status code (success? error?) and usually a body
- All messages have headers with extra information, e.g. cookies and content types
- Request methods might be GET, POST, PUT, DELETE, etc.
- You can inspect HTTP messages in your browser
App development process

• Design screens that users will see
• Define an API
• In parallel:
  • Develop the backend
  • Develop the frontend
• Deploy
• Profit
Outline

1. Introduce backend concepts
2. Decide what to build
3. Define an API
4. Write backend code
5. Write frontend code
What we’ll build

• A rudimentary shared-canvas drawing app

• Supported actions:
  • Get the current canvas
  • Create a rectangle
Outline

1. Introduce backend concepts
2. Decide what to build
3. Define an API
4. Write backend code
5. Write frontend code
Defining an API

• Our job: define expectations for HTTP requests and HTTP responses, including:
  • request method
  • request path
  • request body, if any, including content type and specific requirements
  • response status code(s)
  • response body, including content type and specific shape

• Remember to consider the frontend's perspective
Defining an API • get the current canvas

• The request:
  • should have a method of GET
  • should have a URL path of `/`
  • should not have a body

• The response:
  • should have a code of 200 (“OK”) with a body whose content type is `image/png` containing the canvas as a PNG image
Defining an API • create a rectangle

• The request:
  • should have a method of POST
  • should have a URL path of /rect
  • should have a content type of EDN (extensible data notation)
  • should have a body like this:
    • [50 100 10 30 [0.95 0.5 0.1]]
    • (i.e. x, y, width, height, and RGB color values 0 <= x <= 1)

• The response:
  • If the expectations aren’t met:
    • should have a code of 400 (“bad request”) and a body that explains why
  • If the expectations are met:
    • should have a code of 204 (“no content”) with an empty body
## API Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Params</th>
<th>Status code(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/</td>
<td>-</td>
<td>200 (&quot;OK&quot;)</td>
</tr>
<tr>
<td>POST</td>
<td>/rect</td>
<td>• x</td>
<td>400 (&quot;bad request&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• y</td>
<td>204 (&quot;no content&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• width</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• height</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RGB color values</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Introduce backend concepts
2. Decide what to build
3. Define an API
4. Write backend code
5. Write frontend code

Find the code here:

https://github.com/kyptin/shared-canvas-backend