IN4MATX 285:
Interactive Technology Studio

Practice: Databases and Storage

Today’s goals

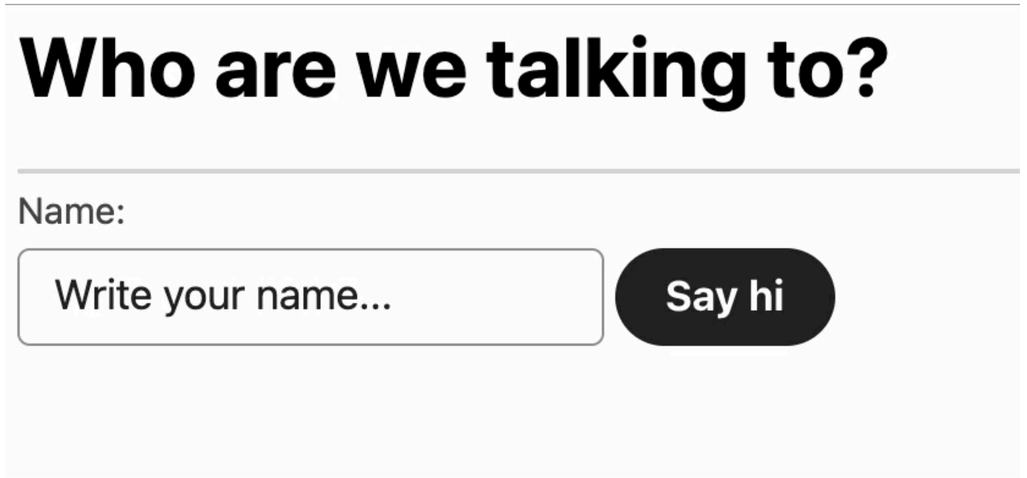
By the end of today, you should be able to…

- Describe how databases and storage support interactive interface design
- Differentiate relational from non-relational databases
- Explain the advantages of each style of database

So we refresh our webpage…

Refreshing

- Often, we lose all the details we entered
  - By default, HTML loses everything when you refresh the page
- Sometimes, key details are remembered
- How is this done? *Storage*



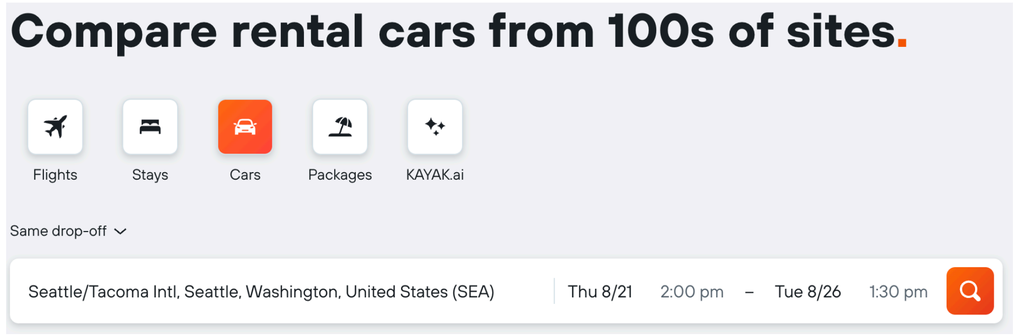
Storage

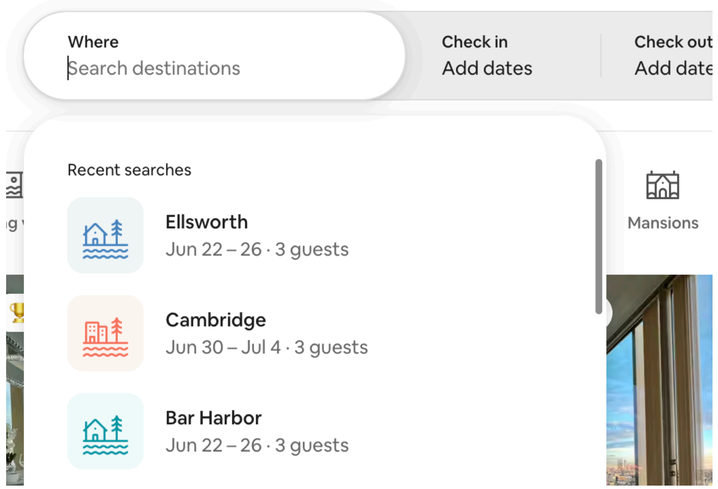
Storage

- Your browser can hold a (small) amount of information about you and your activities on a website
  - Things you search for or type into a text box
  - Some form of login credentials (more on this another week)
  - Types of content you often to engage with
- When you later return to a website, it can tailor the content based on what was stored

Storage

- Example: what you were previously searching for
  - Helps you resume your task more quickly





Storage

- This storage can be either *client side* or *server-side*
  - Client: only visible in your browser
  - Server: shared with the website developers
- Cookies: Server-side storage
  - Often used to track web activity as part of targeting content (ads, recommendations)
- Client-side storage is limited to ~10MB, cookies are 4KB

Storage

- What if you want to store more information about a user?
  - Larger files: profile picture, user-generated images/videos
  - Longitudinal history of messages/emails
- You’re typically looking at server-side storage, such as in a *database*

Storage and regulation

- Subject to GDPR, CCPA (California), and other privacy regulations
- Consent is needed for cookie storage, as these are typically needed for ads/analytics
- Not needed for client storage, as that data remains private

Cookie compliance

To comply with the regulations governing cookies under the GDPR and the ePrivacy Directive you must:

- Receive users' consent before you use any cookies **except** strictly necessary cookies.
- Provide accurate and specific information about the data each cookie tracks and its purpose in plain language before consent is received.
- Document and store consent received from users.
- Allow users to access your service even if they refuse to allow the use of certain cookies
- Make it as easy for users to withdraw their consent as it was for them to give their consent in the first place.

https://gdpr.eu/cookies/

What is a database? If we can store data on devices, why do we need databases?

Databases

- Some sort of organized collection of data, plus tools for retrieving data
- Typically *server-side*
  - Accessible across devices
  - Technically you can have a database in your browser/client, but it's typically not as useful to do so

Databases

- Provide reliability
  - You can get your data back if your phone dies or you get a new phone
- Provide cross-device support
  - Allow you to see and modify the same data across a phone and a desktop, for example

Databases

- Are more than files stored in the cloud
  - Can be “queried” efficiently to get subsets of data
- Two main approaches to making databases
  - Relational databases: MySQL, Postgres
  - Non-relational databases: MongoDB, Firebase, IndexedDB
- *Transaction*: any add/delete/update/etc. made to a database

Databases

Relational databases

- Everything is organized into tables
- Tables contain columns with predefined names and data types
- Tables “relate” to one another by having overlapping or similar columns
  - Minimizes redundancy and keeps order
- Every data entry is a row of a table

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Relational databases

Relational

Person:

Pers_ID	First_Name	Last_Name	City
1	Dexter	Lanasa	Vancouver
2	Ava	Crim	Denver
3	Michael	Plumer	New York City
4	Olivia	Conlin	Dallas
5	Sophia	Hassett	Atlanta
6	Mason	Mora	San Francisco

Phone Numbers:

Phone_ID	Phone_Number	Type	Person_ID
75	111-111-1111	Mobile	1
76	222-222-2222	Home	2
77	333-333-3333	Mobile	3
78	444-444-4444	Home	1
79	555-555-5555	Home	4
80	666-666-6666	Mobile	5
81	777-777-7777	Office	1
82	888-888-8888	Mobile	4
83	999-999-9999	Mobile	5
84	111-222-2222	Office	5

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/

https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Relational databases

```
CREATE TABLE IF NOT EXISTS tasks (
    task_id INT AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    start_date DATE,
    due_date DATE,
    status TINYINT NOT NULL,
    priority TINYINT NOT NULL,
    description TEXT,
    PRIMARY KEY (task_id)
)  ENGINE=INNODB;
```

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/

https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Non-relational databases

- Everything is organized into objects
- There are no restrictions on how objects are structured
- Every data entry is an object, or “document”
  - Documents may be structured differently from one another

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Non-relational databases

MongoDB
Document

```
{
   first_name: 'Dexter',
   last_name: 'Lanas'
   city: 'Vancouver'
   location: [45.123,47.232],
   phones: [
      { phone_number: '111-111-1111',
        type: mobile,
        person_id: 1, ... },
      { phone_number: '444-444-4444',
        type: home,
        person_id: 1, ... },
      { phone_number: '777-777-7777',
        type: office,
        person_id: 1, ... },
   ]
}
```

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/

https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Non-relational databases

- There is no well-defined enforced structure
- That said, flatter structures are generally better

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Advantages of relational databases

- Relational databases support better querying
  - Provide *languages* for querying, such as Structured Query Language (SQL)
  - Those languages can be used to ask for specific tables or even join data across tables
  - “Give me the first name of every user whose phone number starts with 949”

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Advantages of relational databases

- Relational databases are more organized
  - Because field types are defined, data reliably follows that structure
- Relational databases are more reliable
  - Structure is enforced when new data is added
  - Transactions are atomic, so it’s easy to “get” the current state of the database

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Advantages of non-relational databases

- Non-relational databases support more flexibility
  - Structure imposes restrictions
  - Adding a new field (column) can mess up a relational database
- Non-relational databases are faster for simple operations
  - It’s much easier to “watch all the files” than to query and index many rows across multiple tables

https://www.neonrain.com/blog/mysql-vs-mongodb-looking-at-relational-and-non-relational-databases/
https://www.mongodb.com/scale/relational-vs-non-relational-database

Databases

Relational vs. Non-relational

- Relational databases tend to be used in Enterprise, large-scale applications
  - It’s important that data conforms to standards
  - It’s important to robustly query large amounts of data
- Non-relational databases tend to be used in smaller applications
  - Data flexibility is valuable
  - Data is small enough to reliably retrieve and parse
- That said, plenty of large apps use non-relational databases and vice versa

Reflecting on databases

- Developers, especially for backend, think a lot about database structure and optimization
  - Relational versus non-relational
  - How to organize tables for efficiency and interpretability
- But, use cases and frontend functionality should inform this structure

Today’s goals

By the end of today, you should be able to…

- Describe how databases and storage support interactive interface design
- Differentiate relational from non-relational databases
- Explain the advantages of each style of database

IN4MATX 285:
Interactive Technology Studio

Practice: Databases and Storage