DATASTAX

IRT GraphQL – Build a Netflix Clone

Housekeeping

- Break(s) will be provided.
- Refreshments will be served.

- Join WiFi: **WeWork Guest**
- Join the Slack Channel - **https://bit.ly/irt-anz-slack**

- Ask Questions !
- Scroll through exercises at your own pace

- Connectivity Issues:
  - disable VPN and/or Firewall

Introduction

Hilton Rosenfeld

- Developer / Architect
- Application Modernization
- Digital Transformation
- IT Operations Management
- CI/CD

Building a NetFlix Clone

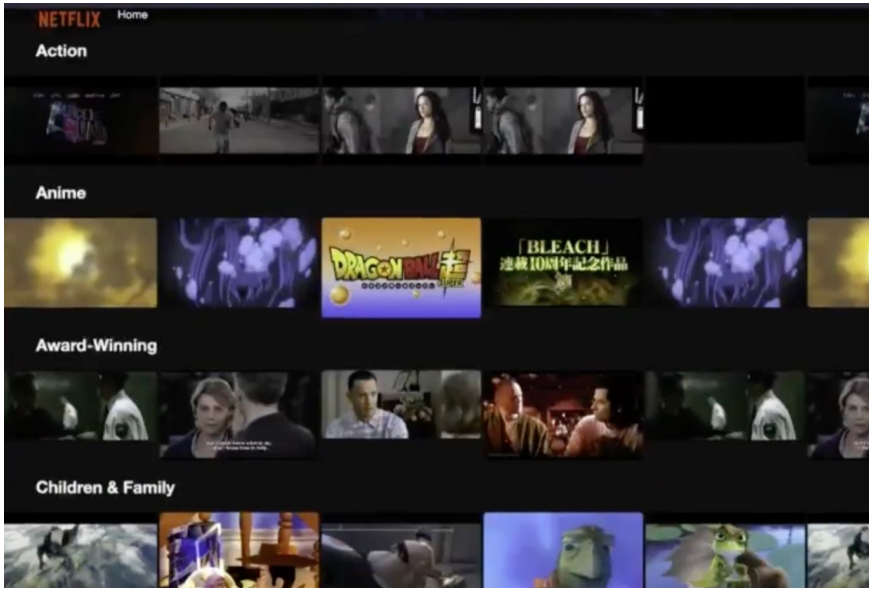
React

GraphQL

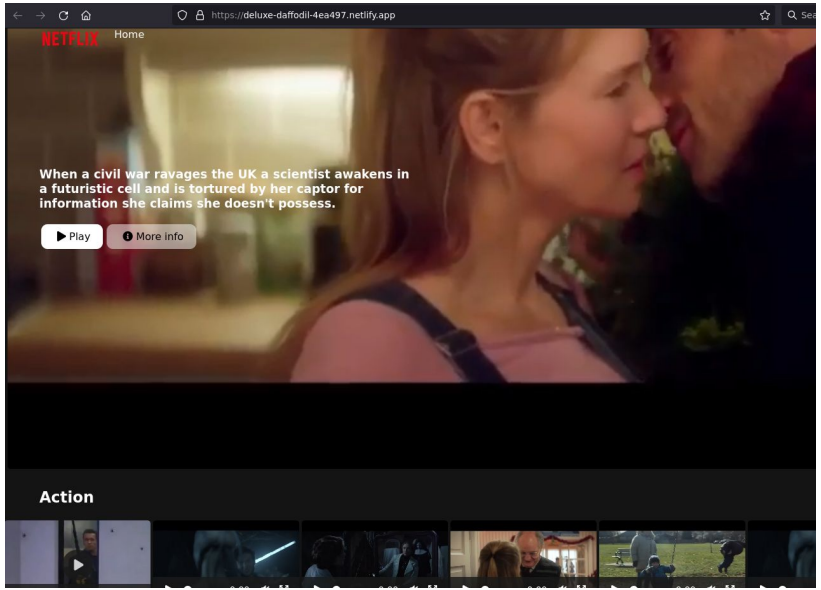
DATASTAX ASTRA DB

What we are building

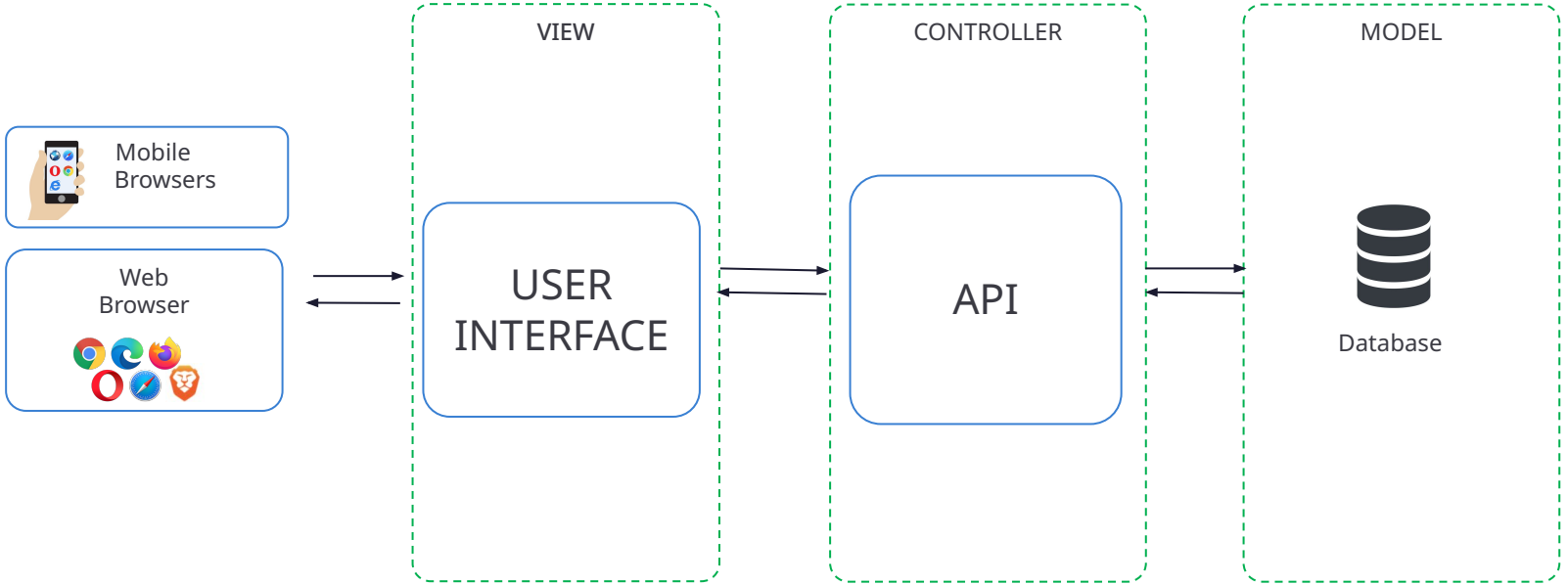
Catalogue of Movies sorted by Genre



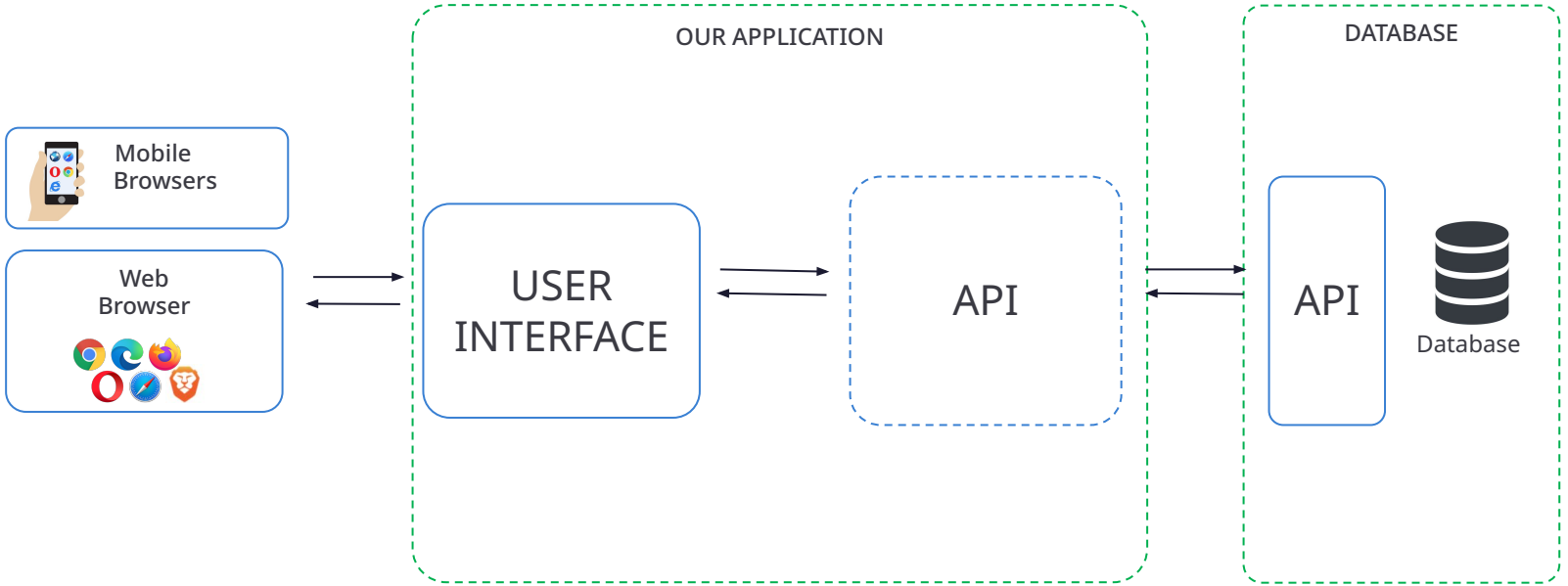
Previews for Movies



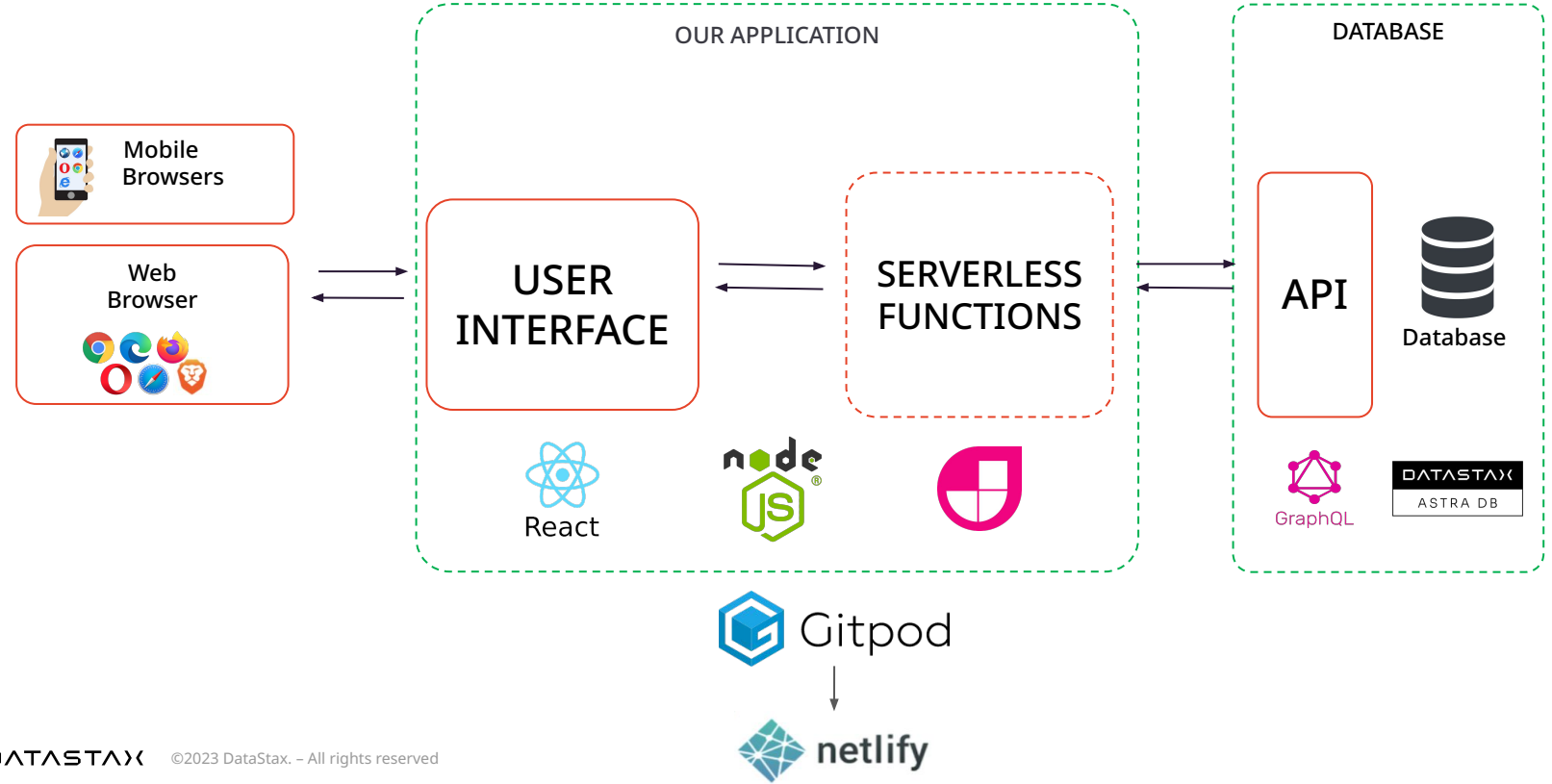
Building a Full Stack Application



Building a Full Stack Application

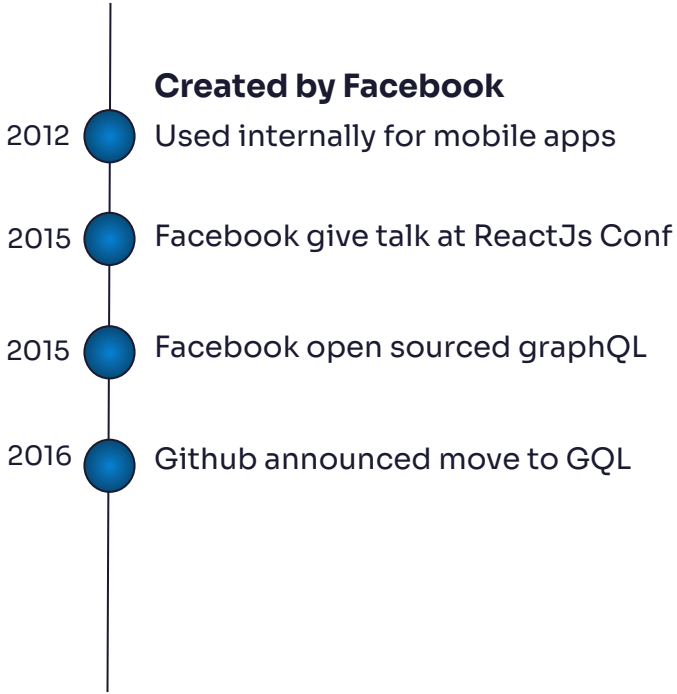


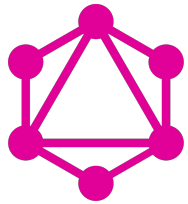
Full Stack App that we will build today



GraphQL

What is GraphQL?

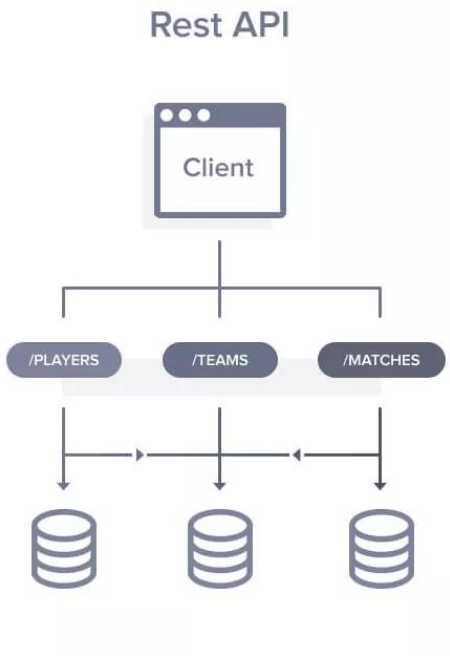


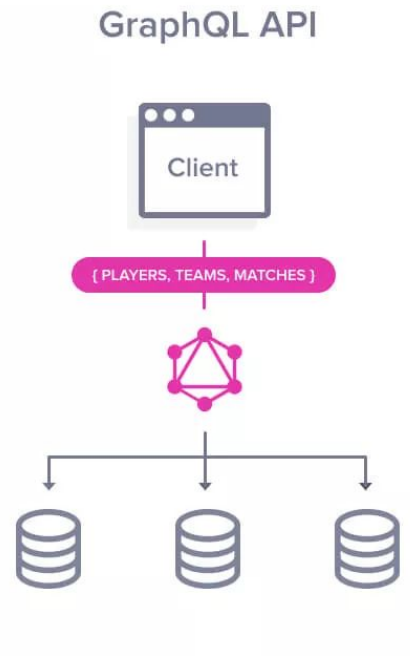


Definition

GraphQL is an API query language and server-side runtime that prioritises giving customers precisely the data they request.

GraphQL is the better REST





Data Fetching with REST

1. GET /users/<id>

```
{
 "user": {
   "id": "er3t8439friw",
   "name": "Mary",
   "birthday": "July 26, 1982"
 }
}
```

2. GET /users/<id>/posts

```
{
 "posts": [{
   "id": "ncwon3ce89hs",
   "title": "Learn GraphQL today",
   "content": "Lorem ipsum",
   "comments": [ ... ]
 }]
}
```

3. GET /users/<id>/followers

```
{
 "followers": [{
   "id": "le083h2doisu",
   "name": "John",
   "birthday": "July 26, 1982"
 },
 ...]
}
```

Data Fetching with GraphQL

1. HTTP POST

```
query {
 User (id: "er3tg439frjw") {
   name
   posts {
     title
   }
   followers(last: 3) {
     name
   }
 }
}
```

```
{
 "data": {
   "User": {
     "name": "Mary",
     "posts": [
       { title: "Learn GraphQL today" }
     ],
     "followers": [
       { name: "John" },
       { name: "Alice" },
       { name: "Sarah" },
     ]
   }
 }
}
```

GraphQL API

Make queries

- Fast
- Flexible
- Client-friendly

Developer can pick exact data the client UI needs

Reduce number of queries by retrieving all relevant data from a single endpoint

GraphQL objects generated for every table

Queries - Read data

Mutations - insert and modify data

Schema API

Create/drop table

Create/delete schema

Query API

Querying and modifying table data using GraphQL fields

DataStax Astra & GraphQL

Astra DB
Zero Lock-In

Astra DB is available to deploy on AWS, GCP and Azure

Retain data sovereignty to meet country-specific regulations

Bulk data loading and integrated stream processing ensure data portability

Maintain API compatibility and consistency of developer tools with Apache Cassandra

Developer Productivity Through Stargate & Astra DB

GraphQL is the better REST

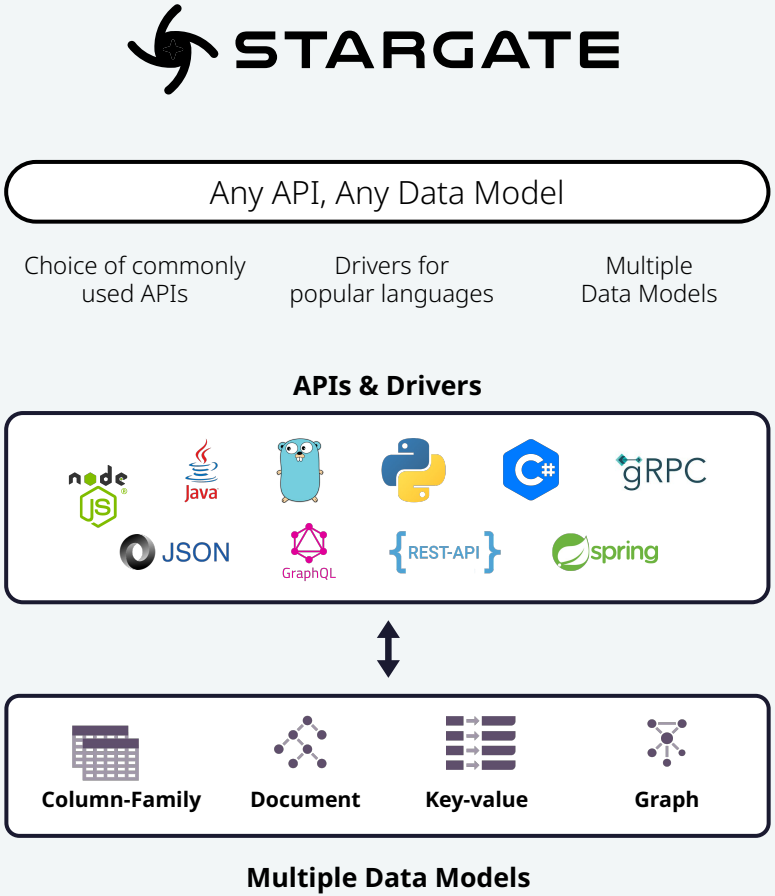
gRPC API is as fast as drivers and easier to learn

DocumentDB: handle JSON with Document API

Prototype via REST/Document API, refactor for speed

Every Astra DB database includes Stargate-as-a-service

Faster Time to Value for applications!



Benefits of our Architecture

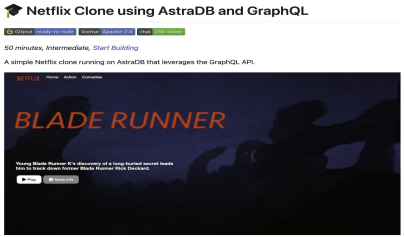
- Cloud native, serverless app and data layer allows you to **scale based on business needs**.
- **Go global** in minutes, with multi-region data layer bringing application data closer to users.
- **No downtime**, always available app and data layer.
- **Fully managed and minimal ops**, across all public cloud deployments.
- Built for performance, for **better user experience**.
- **API enabled**, for ease of consumption and integration.
- **Multiple data models**, to support variety of applications and data.
- Ability to manage all **real-time data**, whether time-based streaming event data or transactional data.

Hands On Time

Tools

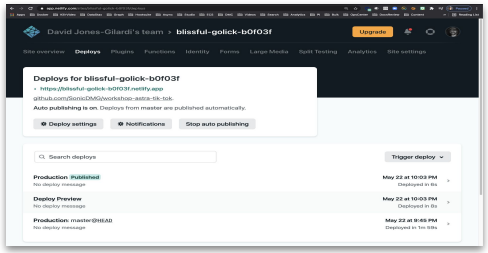
- Nothing to Install!
- Register for your Astra account at: astra.datastax.com
- GitHub repository: github.com/in-realtime/workshop-graphql-netflix

Source code + exercises + slides



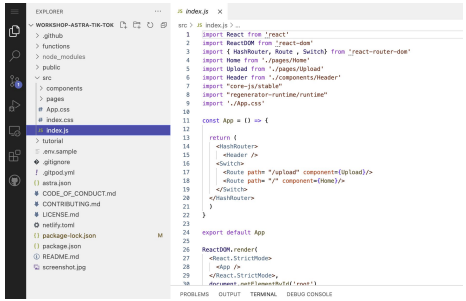
GitHub

Deploy + Run + host



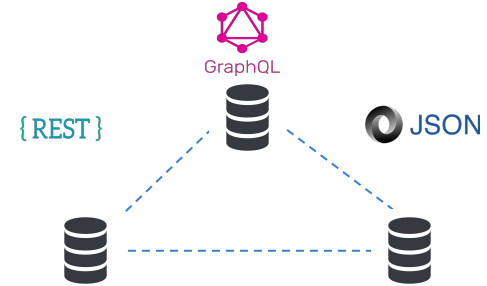
netlify

IDE



Gitpod

Database + GraphQL + PlayGround



DATASTAX ASTRA DB

Lab Activities

Database Setup

- Astra registration
- Create database
- Security token

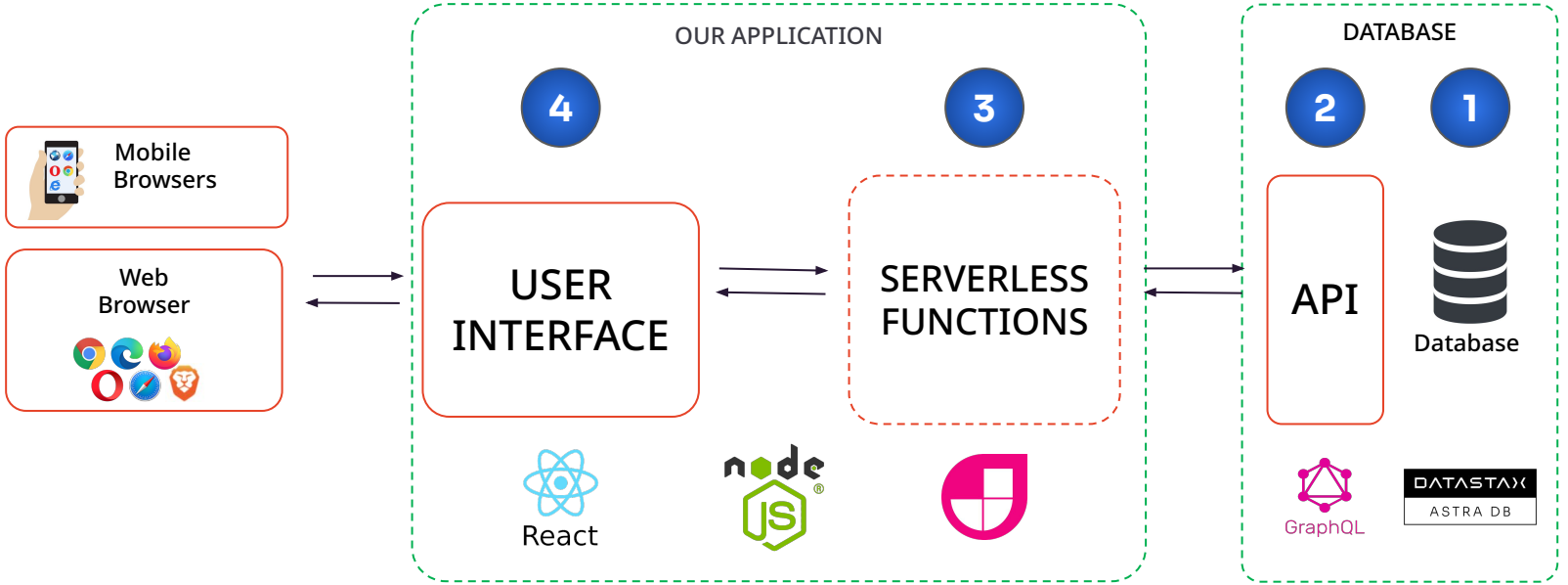
Schema and Data with GraphQL

- GraphQL Playground
- Create schema
- Read & write data
- Bulk import data

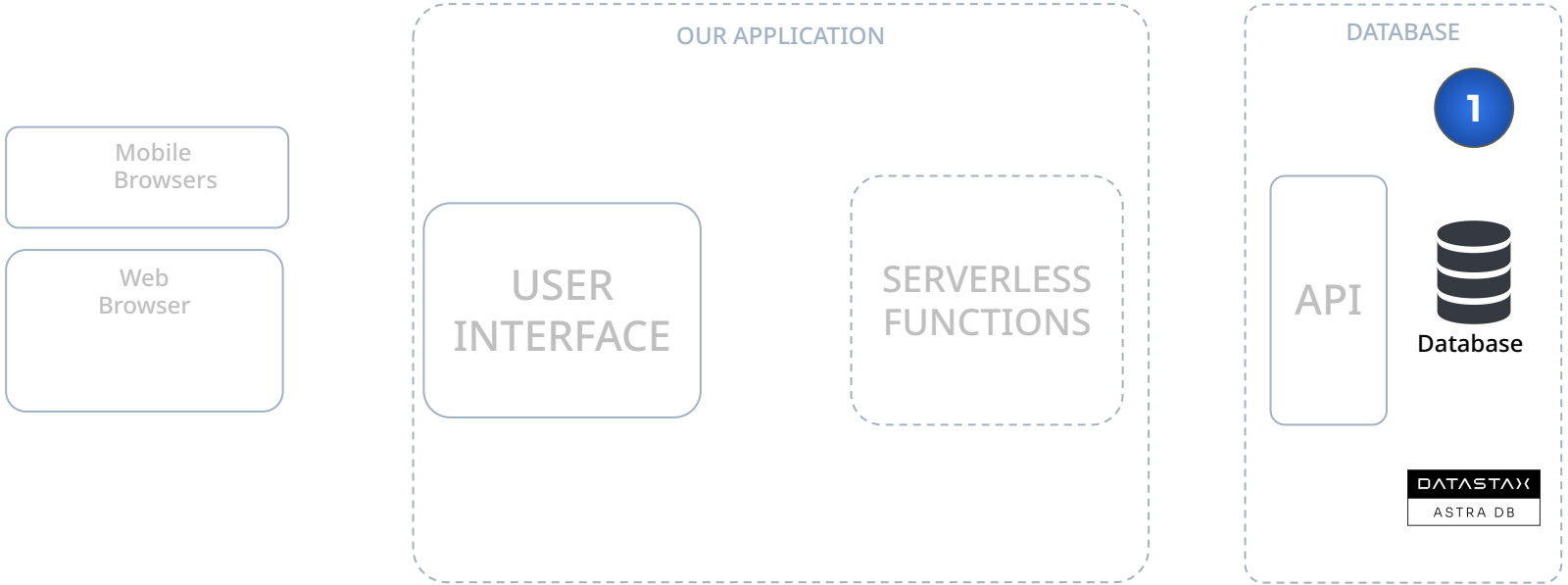
Build & Deploy

- Deploy to Netlify
- Run app in GitPod
- Edit code
- Deploy to Prod
- Run app in Prod

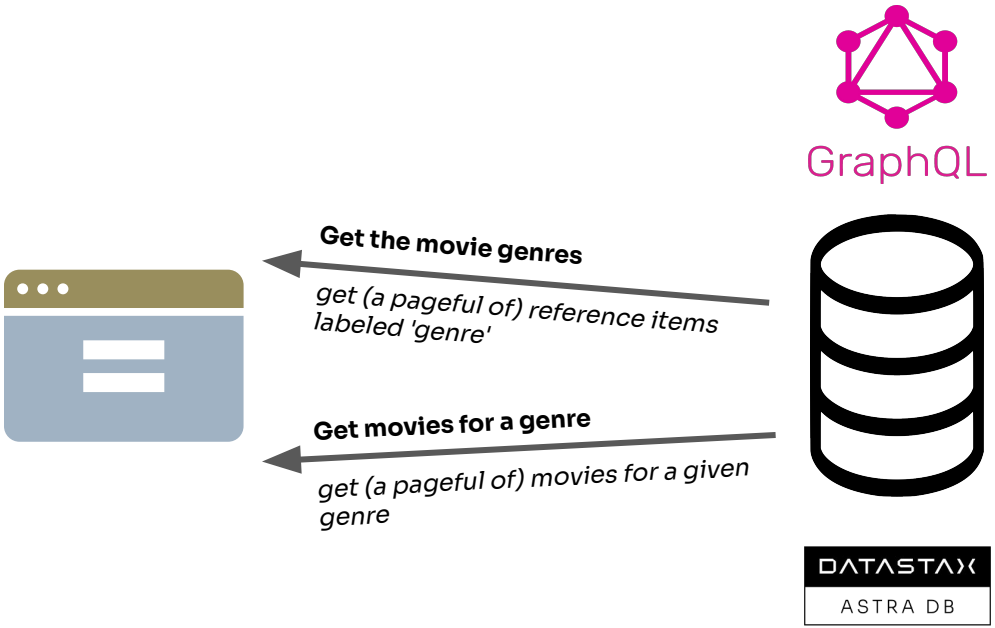
Full Stack App that we will build today



Full Stack App - Database



DB Query patterns



Data modeling, the Astra way:

"design the table after the query"

(also: tables are partitioned!)

Model Data for performance

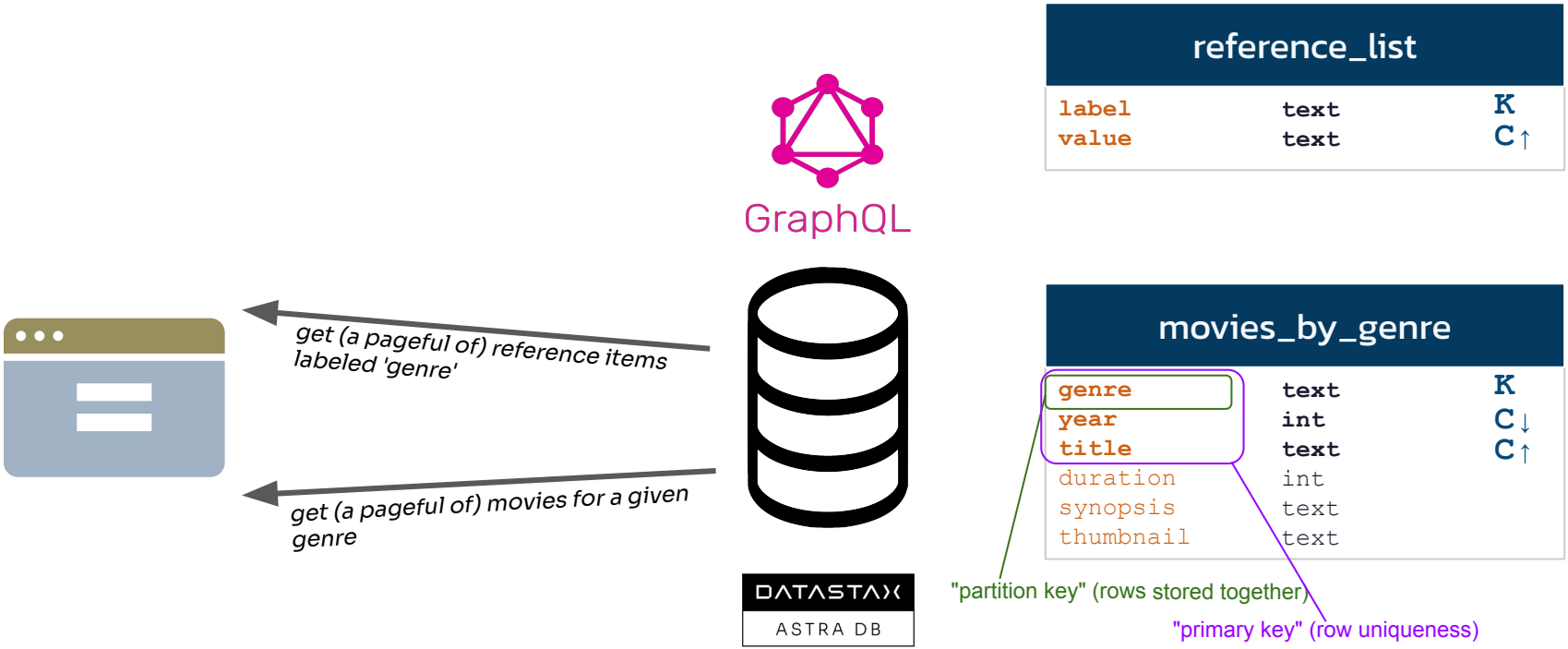
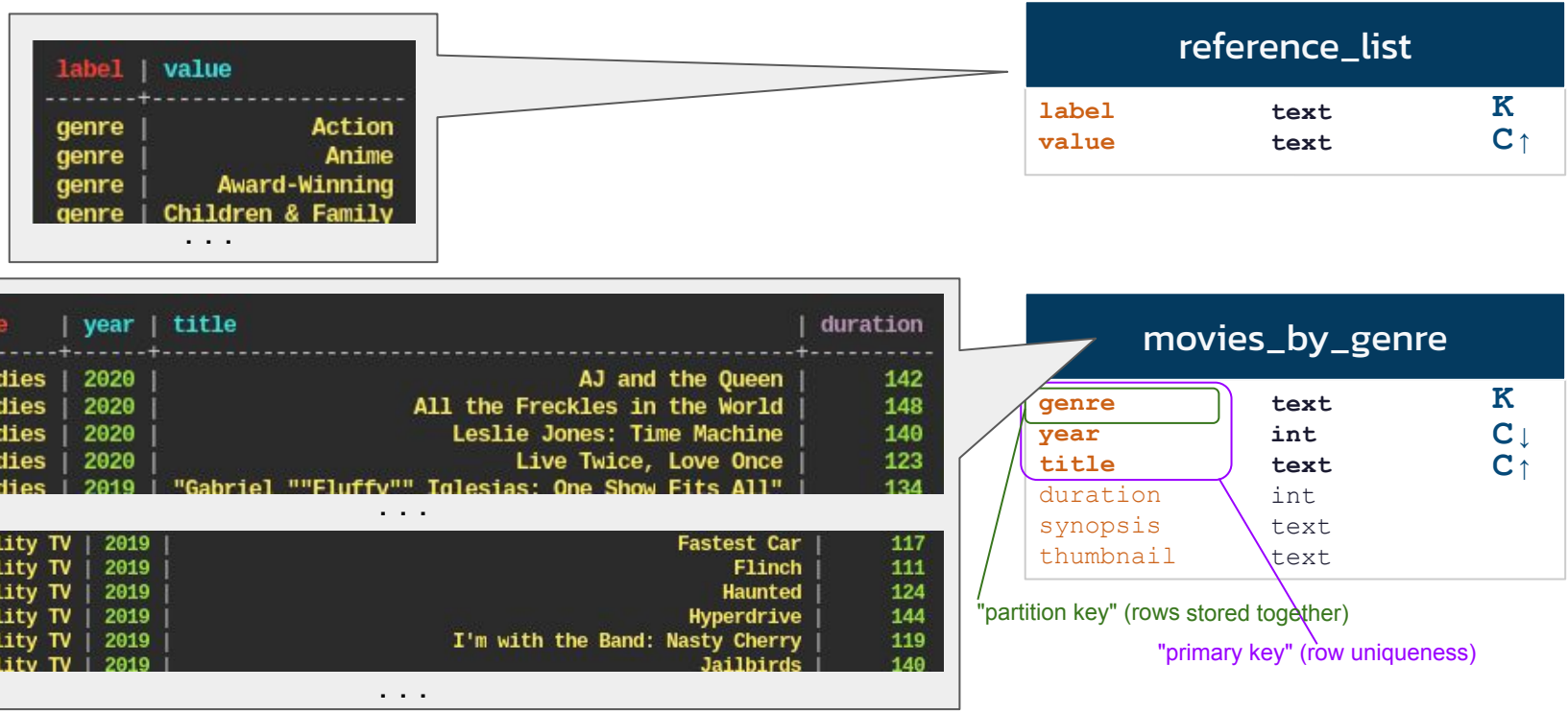
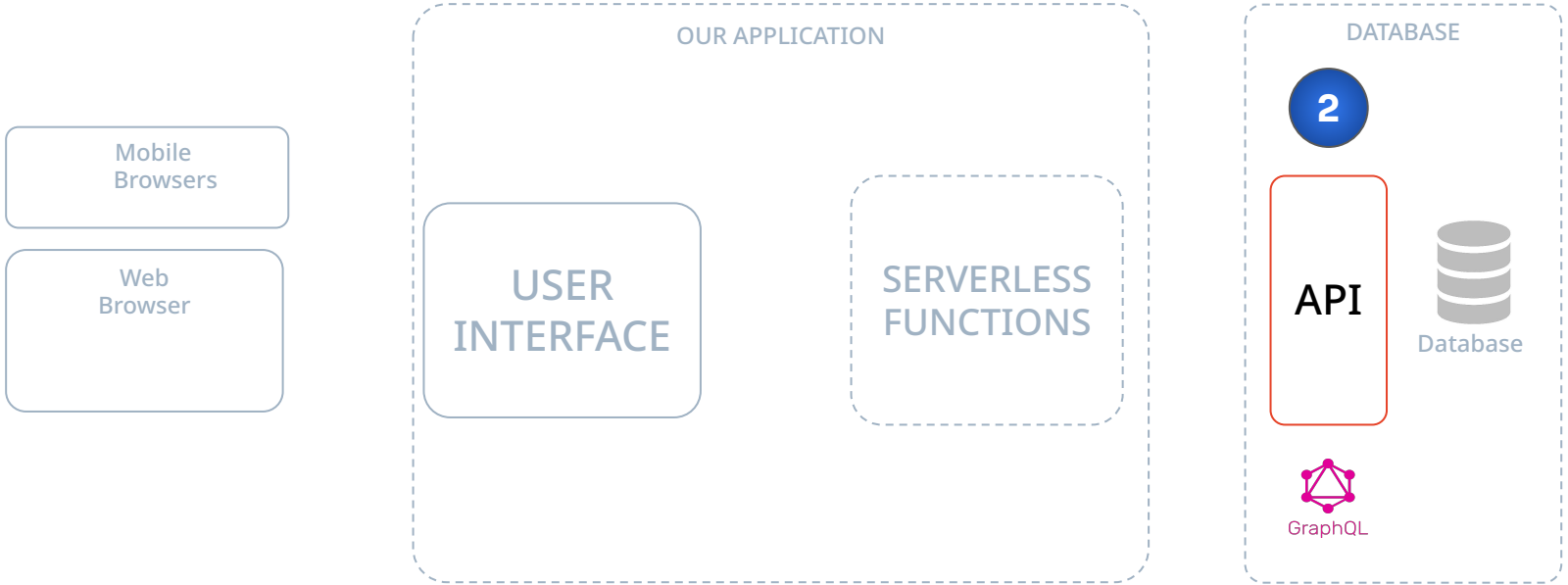


Table Row Examples



Full Stack App - GraphQL API



› Create table for **genres** using GraphQL

```
mutation {
 reference_list: createTable(
   keyspaceName:"netflix",
   tableName:"reference_list",
   ifNotExists:true
   partitionKeys: [
     { name: "label", type: {basic: TEXT} }
   ]
   clusteringKeys: [
     { name: "value", type: {basic: TEXT}, order: "ASC" }
   ]
 )
}
```

List **genres** using GraphQL

```
query getAllGenre {
   reference_list (value: {label:"genre"}) {
     values {
       value
     }
   }
}
```

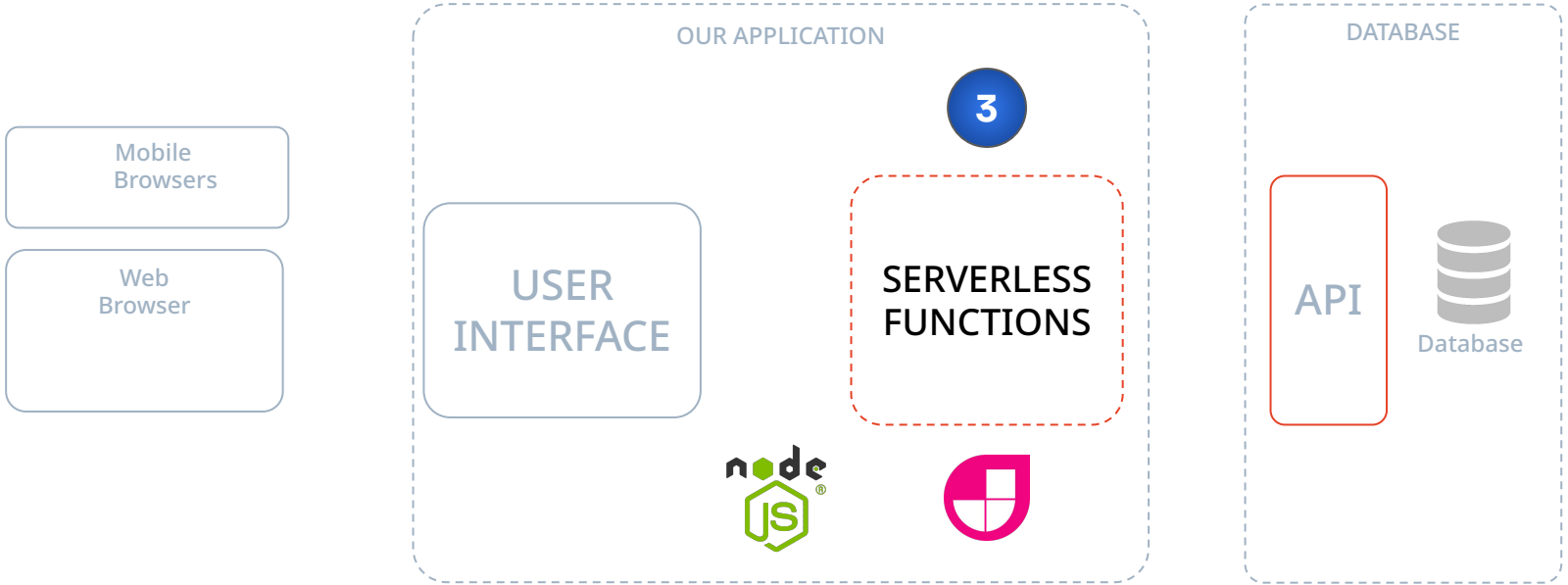
Create table for **movies** using GraphQL

```
mutation {
 movies_by_genre: createTable(
   keyspaceName:"netflix",
   tableName:"movies_by_genre",
   ifNotExists: true,
   partitionKeys: [
     { name: "genre", type: {basic: TEXT} }
   ]
   clusteringKeys: [
     { name: "year", type: {basic: INT}, order: "DESC" },
     { name: "title", type: {basic: TEXT}, order: "ASC" }
   ]
   values: [
     { name: "synopsis", type: {basic: TEXT} },
     { name: "duration", type: {basic: INT} },
     { name: "thumbnail", type: {basic: TEXT} }
   ]
 )
}
```

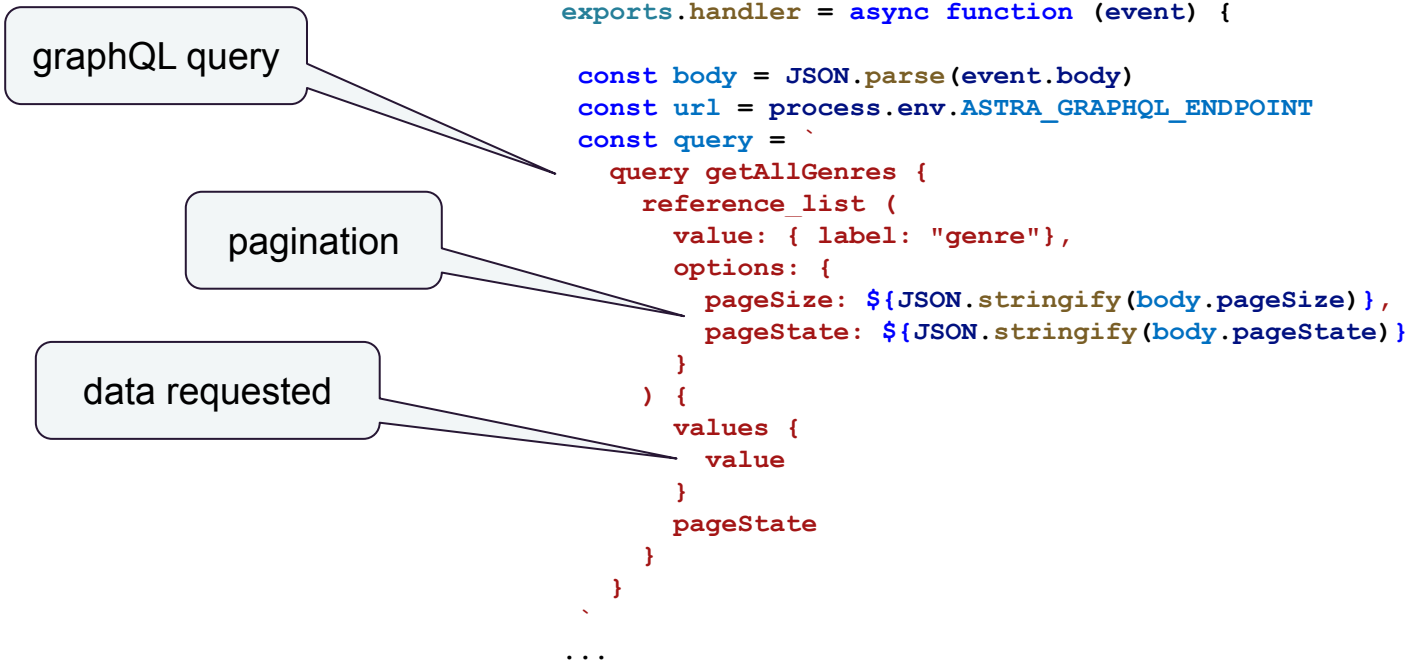
List **movies** using GraphQL

```
query getMovieAction {
   movies_by_genre (
     value: {genre:"Sci-Fi"},
      options: {pageSize: 2, pageState: "YOUR_PAGE_STATE"},
      orderBy: [year_DESC]) {
     values {
       year,
       title,
       duration,
       synopsis,
       thumbnail
     }
   pageState
   }
}
```

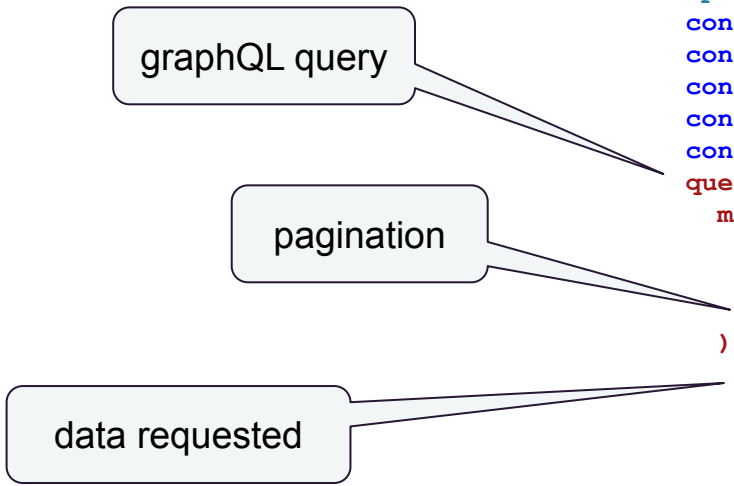
Full Stack App - Serverless Functions



Serverless Function - getGenres.js

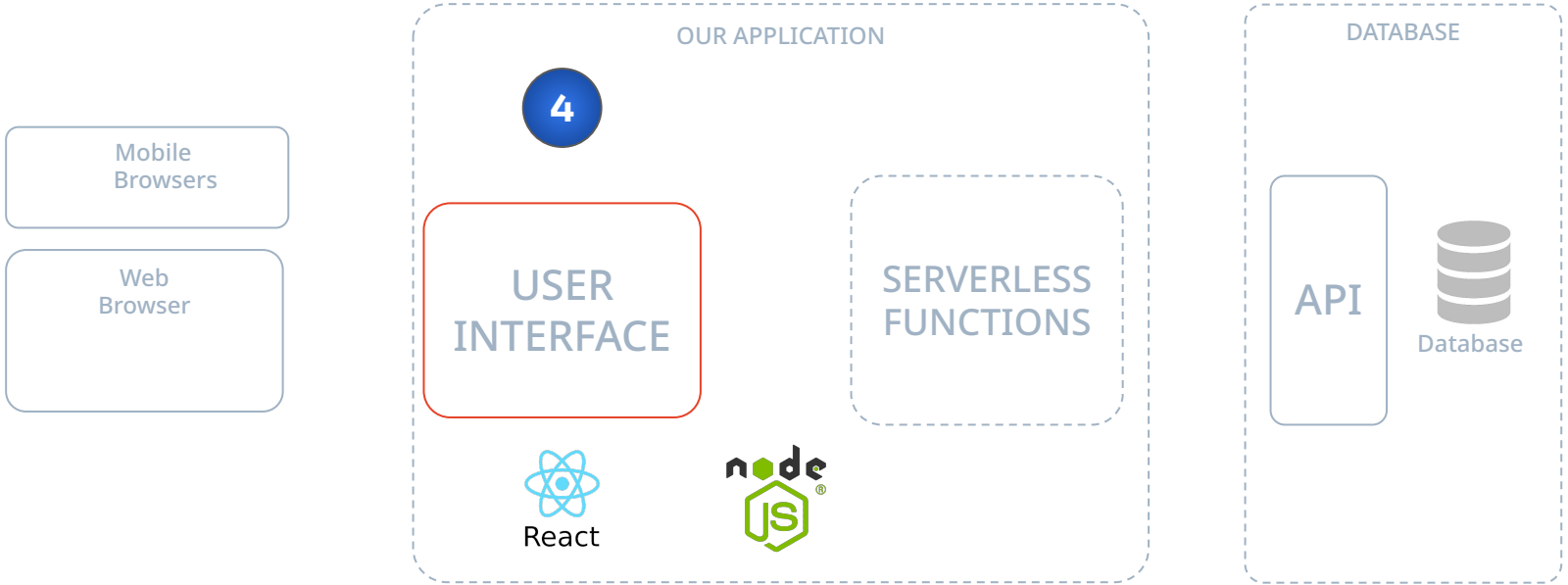


Serverless Function - getMovies.js



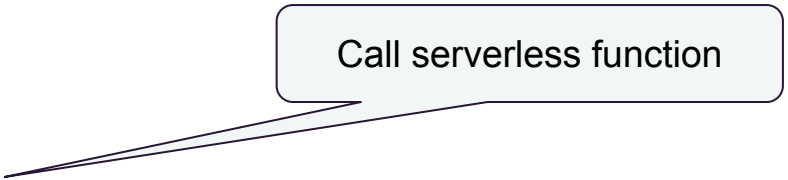

```
exports.handler = async function (event) {
 const body = JSON.parse(event.body)
 const genre = body.genre
 const pageState = body.pageState
 const url = process.env.ASTRA_GRAPHQL_ENDPOINT
 const query = `
 query {
   movies_by_genre (
     value: { genre: ${JSON.stringify(genre)}},
     orderBy: [year_DESC],
     options: { pageSize: 6, pageState: ${JSON.pageState} }
   ) {
     values {
       year,
       title,
       duration,
       synopsis,
       thumbnail
     }
     pageState
   }
 }
 `

 ...
```

Full Stack App - User Interface



Fetching from the frontend

Retrieve a page of genres by calling the getGenres serverless function




```
const fetchData = async () => {
   if (! isFetching)  {
     setIsFetching(true)
     const response = await fetch("/.netlify/functions/getGenres", {
       method: "POST",
       body: JSON.stringify({pageState, pageSize}),
     })
     const responseBody = await response.json()
     setPageState(responseBody.data.reference_list.pageState)
     setGenres(gs => (gs || []).concat(responseBody.data.reference_list.values))
     setIsFetching(false)
   }
 }
```

Fetching from the frontend

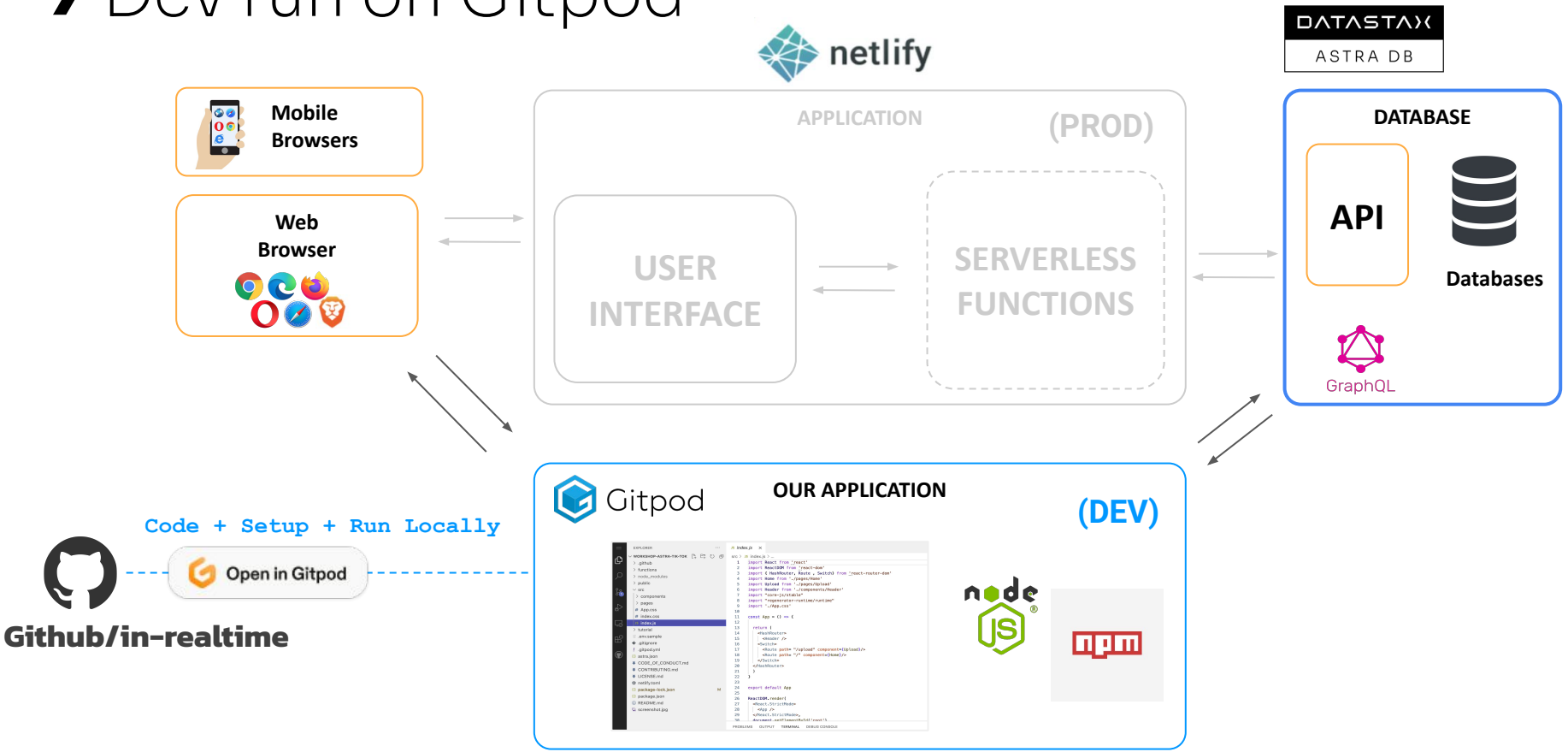
Generate a <Section> component for each genre

```
<>
    <NavBar />
    <HeroSection />
    {genres && (
      <div className="container">
        {Object.values(genres).map((genre) => (
          <Section key={genre.value} genre={genre.value} />
        ))}
      </div>
    )}
    <div
      className="page-end"
      onMouseEnter={() => {
        setRequestedPage( np => np + 1 )
      }}
    />
  </>
```

Iterate on genres

 to catch user activity

Dev run on Gitpod



Help us improve!

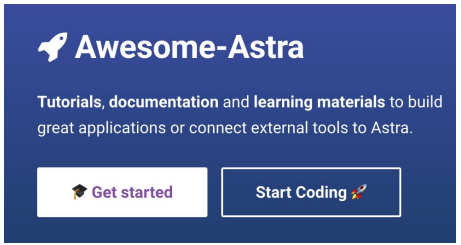
Please use our feedback form:

https://bit.ly/irt-syd-feedback

Key Takeaways and References

1. Why GraphQL is a better REST?
2. How to build a full stack app that scales like Netflix?
3. What is DataStax Astra?

Be an Awesome Astra!



https://bit.ly/irt-awesome-astra

Do your Homework and Earn a Badge!

All info on Github

`!homework`

Give me more!

Another Hands-on GraphQL workshop

https://bit.ly/irt-graphql-workshop

Thank You


DATASTAX