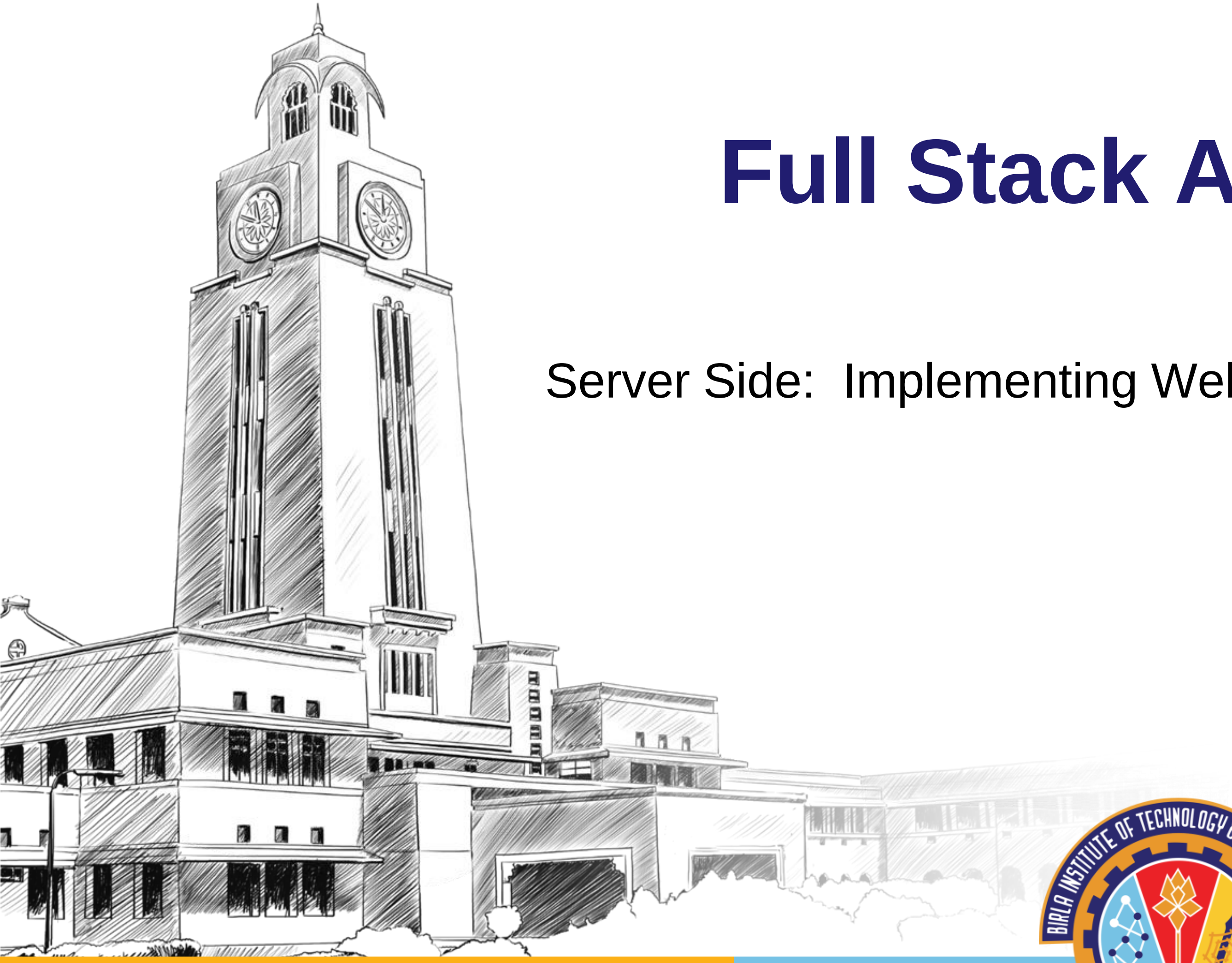

Full Stack Application Development

Server Side: Implementing Web Services

GraphQL

Akshaya Ganesan

GraphQL

- GraphQL is
  - ✓ a query language for your APIs and
  - ✓ a server-side runtime for executing queries.
- A GraphQL service is created by defining types, fields on those types and functions for each field
- A GraphQL query asks only for the data that it needs.

GraphQL

- GraphQL is typically served over HTTP via a single endpoint which expresses the full set of capabilities of the service.
- GraphQL services typically respond using JSON,

REST- disadvantages

- Drawbacks
  - Overfetching
  - Underfetching
- REST APIs need more flexibility.
- As the client needs change, new endpoints have to be created.
- Eg: to fetch the list of books

GraphQL

- A GraphQL operation is either
  - a query (read),
  - mutation (write), or
  - Subscription (continuous read).
- This GraphQL operation is interpreted against the GraphQL schema at the backend, and resolved with data for the frontend application.
- After a GraphQL service runs , it can receive GraphQL queries to validate and execute.
- The service first checks a query to ensure it only refers to the types and fields defined, and then runs the provided functions to produce a result.

GraphQL- Queries

GraphQL Query:

```
query {
  posts {
    title
     author {
      name
      email
    }
  }
}
```

GraphQL Response:

```
{
  "data": {
    "posts": [
      {
        "title": "Introduction to GraphQL",
        "author": {
          "name": "John Doe",
          "email": "john.doe@example.com"
        }
      },
      {
        "title": "Getting Started with React",
         "author": {
          "name": "Jane Smith",
          "email": "jane.smith@example.com"
        }
      }
      // Additional posts...
    ]
  }
}
```

Queries

- GraphQL queries can traverse related objects and their fields, letting clients fetch lots of related data in one request, instead of making several roundtrips as one would need in a classic REST architecture.

Queries- Arguments and Variables

- Every field and nested object can get its own set of arguments.
- Arguments can be of many different types.
- The arguments to fields can be dynamic
- Replace the static value in the query with $variableName
- Declare $variableName as one of the variables accepted by the query
- Pass variableName: value in the separate, transport-specific (usually JSON) variables dictionary

```
{
  “personid": “1000"
}
```

```
human(id: $personid)
```

Queries- Arguments and Variables

Query:

```
{
 human(id: "1000") {
    name
    height(unit: FOOT)
  }
}
```

Response:

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "height": 5.6430448
    }
  }
}
```

Queries

- Default values
- Directives
- Reusable units called *fragments*

Mutations

- Mutations are used to modify data on the server.
- To write new data, we use *mutations*.
- Mutations are defined like queries.
- The Mutation is a root object type.

```
mutation {
  addUser(name: "Alice", age: 30) {
    id
    name
    age
  }
}
```

Schemas and Types

- Every GraphQL service defines a set of types describing the possible data you can query for that service.
- Then, when queries come in, they are validated and executed against that schema.

Schema

- Schema define the data types that your API will expose.
- GraphQL APIs are defined by a schema, which serves as a contract between the client and the server.
- The schema specifies the types of data that can be queried and the relationships between them.
- GraphQL schema documents are text documents that define the types available in an application

Types

- The core unit of any GraphQL Schema is the type.
- These types represent the structure of the data available in the API.
- A type has *fields* that represent the data associated with each object.
- Each field returns a specific type of data.
- A schema is a collection of type definitions

```
type User {
    id: ID!
    name: String!
    age: Int!
  }
```

Schema and Types

- Two types are special within a schema
- Query and
- Mutation
- Every GraphQL service has a query type and may or may not have a mutation type.
- They are special because they define the *entry point* of every GraphQL query

```
const schema = buildSchema(`
  type User {
    id: ID!
    name: String!
    age: Int!
  }

  type Query {
    getUser(id: ID!): User
    getUsers: [User]
  }

  type Mutation {
    addUser(name: String!, age: Int!): User
  }
`);
```

Resolvers

- Resolvers are functions that determine how data is retrieved or modified.
- They map to the types and fields defined in the schema.
- Resolver functions return data in the type and shape specified by the schema.
- Resolvers can fetch or update data from a REST API, database, or anyother service.

GraphQL Service

- A GraphQL service can be written in any programming language,
- It is conceptually split into two major parts,
  - structure and
  - Behavior
- The structure is defined with a strongly typed schema.
- The behavior is naturally implemented with functions and are called resolver functions.

Graphs

- In GraphQL, the foundation is a graph-based approach to modeling business domain.
- By employing GraphQL, the business domain is constructed as a graph,
- Graphs are used formally to represent a collection of interconnected objects.
- Schema outlines different types of nodes and their connections or relationships.

Examples

- Facebook is an undirected graph.
- Think of your GraphQL schema as an expressive shared language for your development team and end-users.
- Creating a robust schema involves examining the everyday language used to describe your business.

GraphQL language

- At the core of a GraphQL communication is a request object.
- The source text of a GraphQL request is often referred to as a document.
- A document contains text that represents a request through operations like queries, mutations, and subscriptions.
- In addition to the main operations, a GraphQL document text can contain fragments that can be used to compose other operations.

Summary

- GraphQL
- Schemas and Types
- Queries and Mutations
- Resolvers

Thank You!

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

ज्ञानं परमं बलम्