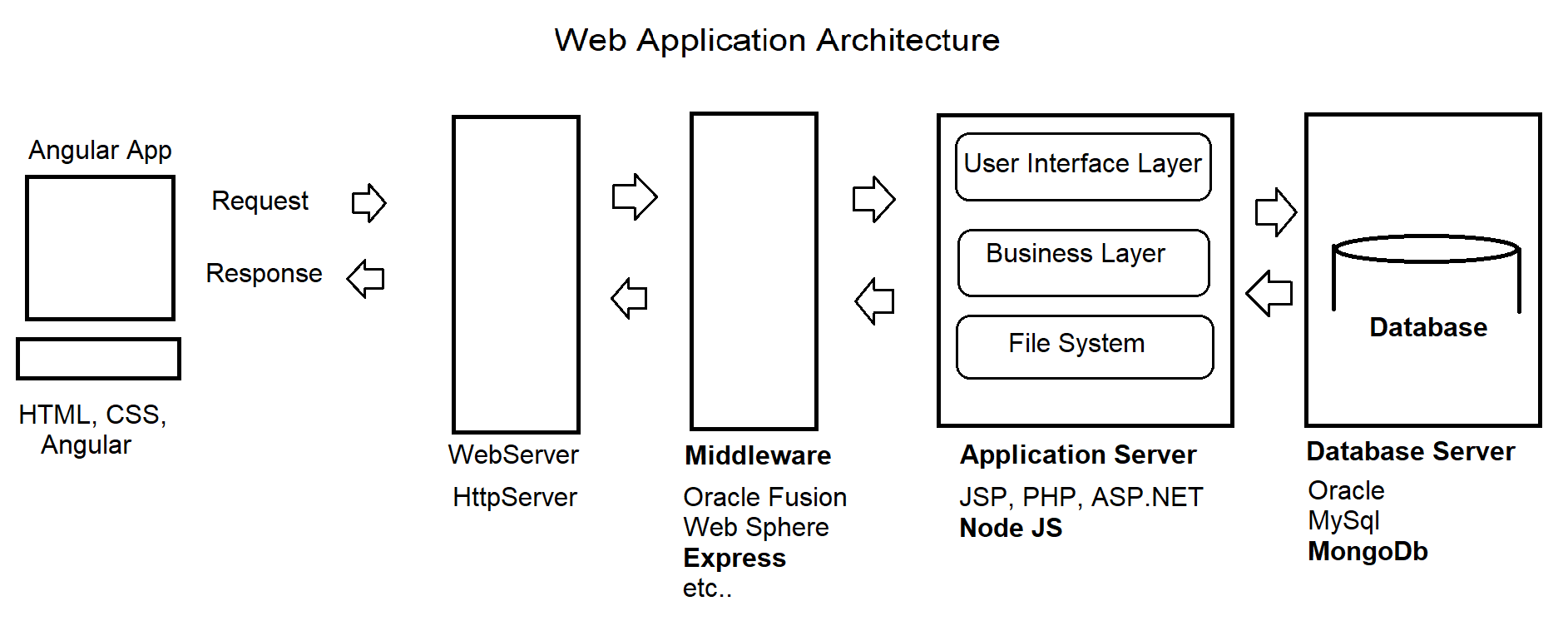
**MEAN Stack – for End to End Application**

****

**Database**

**(MongoDb)**

* Mongo is a cloud service provider.
* MongoDb document-oriented database.
* JavaScript Based
* Non-Sql Database
* Non-RDBMS
* Open Source
* Cross Platform

|  |  |
| --- | --- |
| **RDBMS** | **MongoDb** |
| Database | Database |
| Table | Collection |
| Row | Document |
| Column/Field | Field |
| Join | Embedded Document |

**Setup Environment for MongoDb**

* Download and install MongoDb Database Server.

<https://www.mongodb.com/try/download/community>

* Start MongoDB Database Server
  + Go to “services.msc” in window run option
  + Right Click on “MongoDB Database Server” and “start”
* Open MongoDB Client to configure, create and manipulate database.
  + Open Command Prompt
  + Change to the following location and execute the command  
    **C:\Program Files\MongoDB\Server\4.0\bin> mongo.exe**
  + This will connect to MongoDB server  
    **mongodb://127.0.0.1:27017**

**MongoDB Commands [Case Sensitive]**

|  |  |
| --- | --- |
| **Command** | **Description** |
| show dbs | To view the list of all databases. |
| db | To view the active database. |
| use | To switch to existing database or to create a new database.  Syntax:  > **use database\_name**  **>** use angular10db |
| db.createCollection() | To create a new database table.  Syntax:  > **db.createCollection(“tableName”, options)**  > db.createCollection(“tblproducts”)  > db.createCollection(“tblcategories”) |
| show collections | To View the list of tables |
| db.collectionName.insert() | To insert records (documents) into database table.  Syntax:  > db.collectionName.insert({}) one record  > db.collectionName.insert([{}, {}]) multiple  The datatypes for MongoDB are similar to JavaScript datatype.  Ex:  db.tblcategories.insert({CategoryId:1, CategoryName:"Electronics"})  db.tblcategories.insert([{CategoryId:2, CategoryName:"Footwear"}, {CategoryId:3, CategoryName:"Fashion"}])  Ex:  db.tblproducts.insert([{Name: "Samsung TV", Price: 34000.55, InStock:true, Photo: "assets/tv.jpg", CategoryId:1}, {Name: "JBL Speaker", Price: 4000.55, InStock:true, Photo: "assets/speaker.jpg", CategoryId:1}, {Name: "Nike Casuals", Price: 3000.55, InStock:true, Photo: "assets/shoe.jpg", CategoryId:2}, {Name: "Shirt", Price: 1000.55, InStock:true, Photo: "assets/shirt.jpg", CategoryId:3} ]) |
| db.collectionName.find({}) | To view the records in database table.  > db.tblproducts.find()  > db.tblproducts.find().pretty() |

**Server Side – Node JS**

* Server-side scripting is a technique used in web development, where scripts are employed on server in order generate a response customized to every client request.
* Node JS is an open source, cross platform JavaScript based server-side scripting.
* Asynchronous
* No Buffering
* Single Threaded
* Node JS uses “JavaScript Programs” server-side.
* Server-Side Node JS programs will have extension “.js”

**Nodes JS Server Side Program Approach**

* Install the library required.

***Syntax:***

***> npm install packageName***

* Import the library into program by using “require()”.

***Syntax:***

***var ref = require(“moduleName”);***

* Access the library members by creating a reference.
* Implement the functionality.

**Node JS connecting with MongoDB**

* Install MongoDB library for handling communication with MongoDB database.

**C:\projects-workspace> npm install mongodb**

* Create a new folder “server-side” in your workspace.
* Add a new JavaScript file into folder

**testconnection.js**

//Import MongoDB Client library

var mongoClient = require("mongodb").MongoClient;

//MongoDB Connection String

var url = "mongodb://127.0.0.1:27017";

//Connecting with MongoDB server

mongoClient.connect(url, function(err,clientObj){

if(!err){

console.log(`Connected...`);

} else {

console.log(err);

}

})

* Compile and run form terminal or command prompt

**> node testconnection.js**

**Connecting and Reading Documents from MongoDB Collection**

**Ex: testconnection.js**

//Import MongoDB Client library

var mongoClient = require("mongodb").MongoClient;

//MongoDB Connection String

var url = "mongodb://127.0.0.1:27017";

//Connecting with MongoDB server

mongoClient.connect(url, function(err,clientObj){

if(!err){

var database = clientObj.db("angular10db");

database.collection("tblproducts").find().toArray(function(err, documents){

if(!err){

console.log(documents);

} else {

console.log(err);

}

})

} else {

console.log(err);

}

})

**Connecting and Inserting Documents into MongoDB Collection**

Ex:

var mongoClient = require("mongodb").MongoClient;

var url = "mongodb://127.0.0.1:27017";

mongoClient.connect(url, function(err, clientObj){

if(!err){

var database = clientObj.db("angular10db");

var data = {

Name: "Fossil Watch",

Price: 45000.55,

InStock: true,

Photo: 'assets/fossil.jpg',

CategoryId:1

};

database.collection("tblproducts").insertOne(data, function(err, result){

if(!err) {

console.log("Record Inserted");

}

})

} else {

console.log(err);

}

})

**API and Service**

* Distributed computing Architecture
* A distributed computing Architecture allows two applications running or two different machines to share information. (or) Two applications running on same machine but on different process can share information.
* Various Technologies
  + CORBA
  + DCOM
  + RMI
  + EJB
  + Web Services
  + Remoting etc.
* Create an application that can reach broad range of devices i.e from a browser to mobile.
* Enable communication between client and server application for sharing information.
* API and Web Service Specifications
  + SOAP
  + REST
  + JSON
* SOAP [Service Oriented Architecture Protocol]
  + Client Request will be in XML
  + Server Response will be in XML
* REST [Representational State Transfer]
  + Client Request will be a simple query [Query String]
  + Response will be in XML or JSON
* JSON [JavaScript Object Notation]
  + Client Request will be in JSON
  + Server Response will be JSON
* XML & JSON
  + Work Offline
  + Transport data without COM Marshalling [Converting Binary to Object -Object to Binary]
  + XML cross platform
  + JSON is cross platform, Native to browser.

**Middleware**

* Express.js or Express
* It is a back-end web application framework for Node.js.
* Open Source, Cross platform.
* Allows communication between client and server-side application.
* It is responsible for parsing the data, locating the static files requested by client and serving the files to client, handling CORS etc.
* **Express listens** from server and **process requests** made by client.
* Client request can be
  + GET – To fetch resources from server
  + POST- To submit resource to server
  + PUT – To modify the resources on server
  + DELETE – To remove the resource from server
* Server-Script scripting handling “Server-side Objects”
  + **Request Object:**   
    It provides a set of properties and methods that are used by server that accept request from client and fetch the resource data from server.  
    Server uses request object to fetch the client details, like query string, cookie, form body.
  + **Response Object:**   
    It provides a set of properties and methods that are used by server in order to send response to client. [HTML, File, JSON, XML etc]
  + Application Object
  + Session Object
  + Cookie Object

**Creating Web API using Express**

* **Install “express” library for your project.**  
  > npm install express
* ***Add a new JavaScript file*** **“api.js”**

var express = require("express");

var app = express();

app.get("/", function(request, response){

response.send("Welcome to API");

});

app.get("/getproducts", function(req, res){

res.send([{"Name":"TV"},{"Name":"Mobile"}]);

});

app.get("/getdetails/tv", function(req, res){

res.send({"Name":"TV", Price:23500.44});

});

app.post("/addproduct", function(req, res){

res.send("POST - This is request for data submit");

});

app.put("/updateproduct", function(res, res){

res.send("PUT - This is request to modify data");

});

app.delete("/deleteproduct", function(res, res){

res.send("DELETE - This is request to Delete data");

});

app.listen(8080);

console.log("Server Started and Listening on : <http://127.0.0.1:8080>");

* **Test from browser or any web debugger like: fiddler, postman etc**.  
  <http://127.0.0.1:8080/getproducts>

Ex:

**Api.js**

const { response } = require("express");

var express = require("express");

var mongoClient = require("mongodb").MongoClient;

var url = "mongodb://127.0.0.1:27017";

var app = express();

app.get("/", function(request, response){

response.send("Welcome to API");

});

app.get("/getproducts", function(req, res){

mongoClient.connect(url, function(err,clientObj){

if(!err){

var database = clientObj.db("angular10db");

database.collection("tblproducts").find().toArray(function(err, documents){

if(!err){

res.send(documents);

} else {

console.log(err);

}

})

}else {

console.log(err);

}

})

});

app.get("/getdetails/tv", function(req, res){

res.send({"Name":"TV", Price:23500.44});

});

app.post("/addproduct", function(req, res){

res.send("POST - This is request for data submit");

});

app.put("/updateproduct", function(res, res){

res.send("PUT - This is request to modify data");

});

app.delete("/deleteproduct", function(res, res){

res.send("DELETE - This is request to Delete data");

});

app.listen(8080);

console.log("Server Started and Listening on : <http://127.0.0.1:8080>");

* Request for accessing data from MongoDB <http://127.0.0.1:8080/getproducts>

**Body Parser**

* It is a library required to parse the data into JSON.
* It uses URL encoding and converts the form data into JSON in order to submit to the server.
* Download and Install body parser for your server-side application.  
  **> npm install body-parser**

Syntax:

var bodyParser = require(“body-parser”);

app.use(bodyParser.urlencoded({

extendend:true

}))

app.use(bodyParser.json());

**CORS Configuration**

* Cross Origin Resource Sharing
* Server Application and Client Application are running of different port numbers.
* Sharing information across the origin will be blocked.
* You have to allow sharing across origin for different types of requests.
* You can manage by configure **CORS** in your middleware.
* Express CORS configuration comprises of following details

Syntax:

app.use(function(req, res, next){

res.header(“Access-Control-All-Origin”, “\*”);

res.header(“Access-Control-Allow-Headers”, “Origin, x-Requested-With, Content-Type, Accept”);

res.header(“Access-Control-Allow-Method”, “GET”, “POST”, “PUT”, “DELETE”);

next();

})

**Create Business Layer to Handle Communication Between Client Side and Server-Side Application**

* You have to install the following library
  + **MongoDB**> npm install mongodb
  + **Express**  
    > npm install express
  + **Body Parser**> npm install body-parser
* Create a server-side Business Logic to handle interaction with database in backend and client-side application.

**RESTapi.js**

const { RequiredValidator } = require("@angular/forms");

//import library

var mongoClient = require("mongodb").MongoClient;

var express = require("express");

var bodyParser = require("body-parser");

//Configure MongoDB Connection String

var url = "mongodb://127.0.0.1:27017";

//Configure Middleware

var app = express();

//Configure Body Parser

app.use(bodyParser.urlencoded({

extended:true

}))

app.use(bodyParser.json());

//Configure CORS

app.use(function(req, res, next){

res.header("Access-Control-All-Origin","\*");

res.header("Access-Control-Allow-Headers","Origin, x-Requested-With, Content-Type, Accept");

res.header("Access-Control-Allow-Methods","GET","POST","PUT","DELETE");

next();

});

app.get("/getproducts", function(req, res){

mongoClient.connect(url, function(err, clientObj){

if(!err){

var database = clientObj.db("angular10db");

database.collection("tblproducts").find().toArray(function(err, documents){

if(!err){

res.send(documents);

} else {

console.log(err);

}

})

}else {

console.log(err);

}

})

});

app.post("/addproducts", function(req, res){

mongoClient.connect(url, function(err, clientObj){

if(!err) {

var database = clientObj.db("angular10db");

var data = {

Name:req.body.Name,

Price:req.body.Price,

InStock:req.body.InStock,

Photo: req.body.Photo,

CategoryId: req.body.CategoryId

}

database.collection("tblproducts").insertOne(data, function(err, result){

if(!err){

console.log("Record Inserted");

} else {

console.log(err);

}

})

} else {

console.log(err);

}

})

});

app.listen(8080);

console.log("Server Listening : http://127.0.0.1:8080");

**Consume in Angular**

* RxJS library
* HttpClient