**General guideline to follow while consume the APIs**

# General Guideline

Define the business objectives and the technical challenges involved in the creation of this technology. It is also necessary to decide whether we want a public or private API.

To achieve the best result, these components must be organized in layers:

**Security layer:** APIs may also become a threat to security. That is why it is essential to implement a sound and specific security layer on the periphery of the API architecture.

**Cache storage layer:** delivers cached responses to common queries. This reduces the pressure on implementations and real administrative resources.

**Representation layer:** The presentation must be as simple as possible for developers, so that they can focus on developing a user-friendly access to their APIs.

**Organization layer:** if APIs are to be capable of delivering real value, data from various APIs and various administrative resources must be combined. To carry out these compositions an organization layer has to be implemented together with the interfaces.

**Microservices Architecture:** Microservices Architecture the idea is to split your application into a set of smaller, interconnected services instead of building a single monolithicapplication. Each microservice is a small application that has its own hexagonal architecture consisting of business logic along with various adapters

**Boilerplate code**: boilerplate code or boilerplate refers to sections of code that have to be included in many places with little or no alteration.

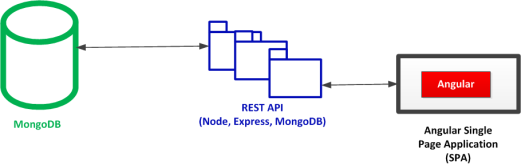
# MEAN Stack Applications

Single Page Applications (SPA)

A browser sends an initial user’s request to the server and server returns AngularJS application with requested data. The subsequent user’s requests are processed most of the time by the AngularJS application running in the browser while data goes back and forth between the browser and server.

SPA also supports two-way data binding where the template and data are sent independently to the browser.

The browser compiles the template into a view and the data into a model. The view is “live” since it is bound to the model. If the model changes the view changes as well and if the view changes then the model also changes.



# MEAN Technologies

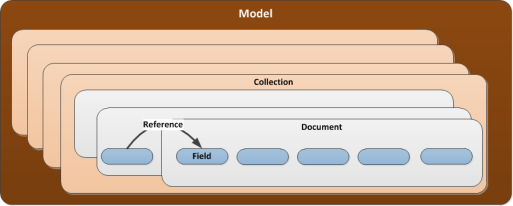
MEAN includes five main technologies:

1. **MongoDB** database and Mongoose object data modeling (ODM) tool
2. **Express** middleware
3. **AngularJS** front-end
4. **Node.js** server platform

# Mongo DB

MongoDB is a NoSQL document-based database management system which data model includes:

* Collections
* Documents
* Fields
* References



Collections

A Collection is a top model element. Each model can have one or more collections. Collections are analogous to tables in a relational database. Each collection contains documents that are analogous to records in the relational database. Collections model one or more concepts (e.g., account, user, order, publisher, book, etc.) the data is based on.

Documents

Documents are JSON-like data structures containing fields that have values of different types (e.g., String, Date,  Number, Boolean, etc.). A value can also belong to another document or an array of documents embedded in a document. Documents can have different structures in a collection. However, in most cases in practice, collections are highly homogeneous

Fields

Fields are analogous to columns in the relational database. The field/value pairs (better known as key/value pairs) construct document’s structure.

# Mongoose

Mongoose is a MongoDB object data modeling (ODM) tool designed to work in an asynchronous environment. Besides the data modeling in Node.js, Mongoose also provides a layer of CRUD features on top of MongoDB. It also makes it easier to manage connections to MongoDB databases and perform data validations.

# Express

Express is a middleware framework for Node.js that abstracts away some common web server functionalities. Some of these functionalities include session management, routing, templating, and others.

# Node.js

Node.js is a foundation of the MEAN stack. Node.js is not a language. It is a software platform based on JavaScript. You will use it to build your own web server and applications that will run on top of it. Node.js applications when codded correctly are fast and they efficiently use system resources. This is supported by the core Node.js feature that it is single-threaded and executes a non-blocking event loop.

Drawback of multi-threaded system:

The web server running on Node.js is different from traditional multi-threaded web servers (e.g., Apache, IIS, etc.). The multi-threaded servers create new thread for each new user session and allocates memory and other computing resources for it. During the peak periods when many users access the server concurrently its resources can get exhausted in which case the system could halt its operations until the load decreases and/or more machines and resources are added.

When Node.js is used, rather than giving each user a separate thread and pool of resources, each user joins the same thread and the interaction between the user and thread exist only when it is needed. In order to ensure that this approach works Node.js supports non-blocking by making blocking operations run asynchronously.

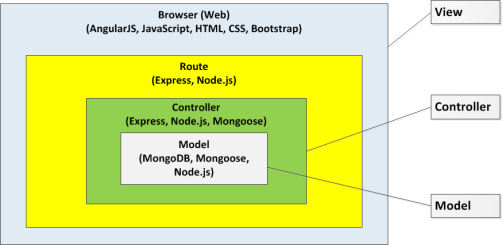
# Angular JS

While you can use Node.js, Express and MongoDB to build data-driven applications, the use of AngularJS will bring more sophisticated features to the interactivity element of the MVC architectural pattern supported by MEAN. AngularJS puts HTML together based on provided data. It also supports two-way data binding by immediately updating the HTML based on changed data and also by updating the data if HTML changes.

# MEAN Architectural Patterns

MEAN architectural patterns are based on the Model-View-Controller (MVC) pattern.

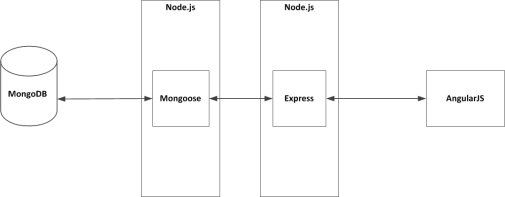
The MVC pattern is data oriented. Model holds data, Controller processes data and view renders data. There is also a route component between the controller and users’ browsers (Web). The route component coordinates interactions with the controller.



# Mongoose

There is also one more important technology component, Mongoose, that is a liaison between the controller and MongoDB

MongoDB communicates with Mongoose only. Mongoose communicates with Node.js and Express and AngularJS communicates with Express only.



# MEAN STACK REST API

The REST API is a common architectural element used in all MEAN architectural patterns.

A common way to architect MEAN stack is to have a REST interface feeding a single page application (SPA).  REST interface is implemented via REST API that is built with MongoDB, Node.js and Express and SPA is built with AngularJS that runs in browser.

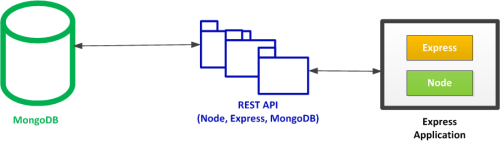
REST API creates a stateless interface to your database. It enables other applications to work with your data.

# Architectural patterns are enabled by the MEAN framework

* Node.js and Express Application (NEA)
* Node.js and Express application with AngularJS addition for better interactivity (NEA2)
* AngularJS Single Page Application (SPA)

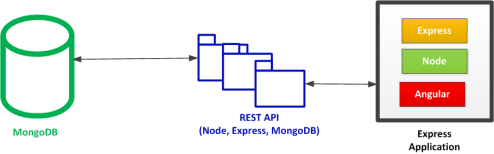
# Node.js and Express application (NEA)

HTML and content are directly delivered from the server. The HTML content requires data that is delivered via REST API. REST API is developed with Node.js, Express, Mongoose and MongoDB.



### Node.js and Express Application with AngularJS Addition for Better Interactivity (NEA2)

If you need a richer interactive experience for your users, you can add AngularJS to your pages.



### AngularJS Single Page Application (SPA)

In order to implement Single Page Applications, AngularJS is needed.

