Proposed Microservices Architecture

esm_logo.jpg

Nexelus USA  
New York, NEW YORK (NY)  
646-558-1950 ext.128

**2/20/2023**

# Document Information

The following table shows the details for document creation, review, approval, and effective date.

| **Category** | **Information** |
| --- | --- |
| Work Product: | Proposed Microservices Architecture |
| Product Name: | Nexelus |
| Function Name: | SOC 1 Type II Document |
| Version: | 0.1 |
| Status: | Draft |
| Author(s): | Adnan Abbas |
| Reviewer(s): | Asim Jameel, Tauseef Shahzad |
| Approver(s): | Imran Rahman |
| Control Status: | CONTROLLED, PROTECTED |
| Disclaimer: | This document contains confidential information. Do not distribute this document without prior approval from Nexelus. |

# Revision History

The following table is used for revision details of this document.

| **Author(s)** | **Date** | **Version** | **Description of Change** |
| --- | --- | --- | --- |
| Adnan Abbas | Feb 2, 2023 | 0.1 | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

Contents

[Document Information ii](#_Toc127784414)

[Revision History iii](#_Toc127784415)

[Table of Contents i](#_Toc127784416)

[Scope 1](#_Toc127784417)

[Executive Summary 2](#_Toc127784418)

[The Objective 2](#_Toc127784419)

[Technologies 2](#_Toc127784420)

[Research Areas 3](#_Toc127784421)

[Multi-Tanent System 4](#_Toc127784422)

[Messaging with in the MicroService 4](#_Toc127784423)

[Dapper Vs EF Core: 5](#_Toc127784424)

[Logging 5](#_Toc127784425)

## Scope

Nexelus Application has been developed using Microsoft .Net Webforms. Some newer modules have been created using Microsoft .Net Core platform. .Net Core application is created as separate application, however both applications share the session/state information managed in Web Forms application.

The scope of this document is to propose migrating Nexelus to Microservices architecture and to provide high level architecture details.

## Executive Summary

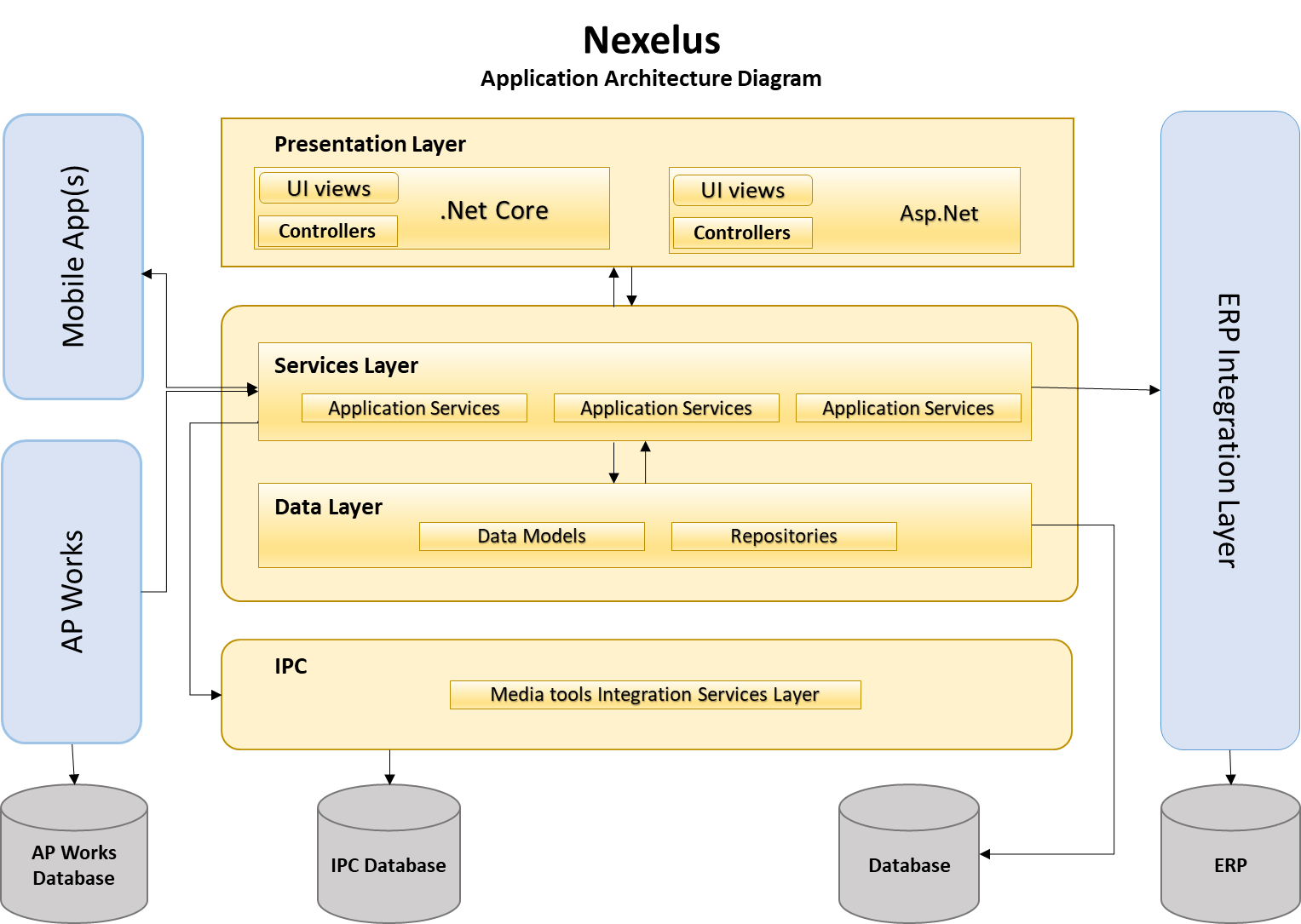
The objectives for the new architecture are:

* Module Wise Micro Front End
* Module Wise Microservice
* Microservices using Gateway
* Need to integrate this Application with Current Application
* Need to Enable Application to Support Multi-Tenant.

## Technologies

#### Existing Architechure

Monolithic front ends have cumbersome codebases that become even more unstructured and difficult to maintain over time. Micro frontends solve this issue. The source code of each individual micro frontend is smaller, simpler, and, thus, much more manageable.

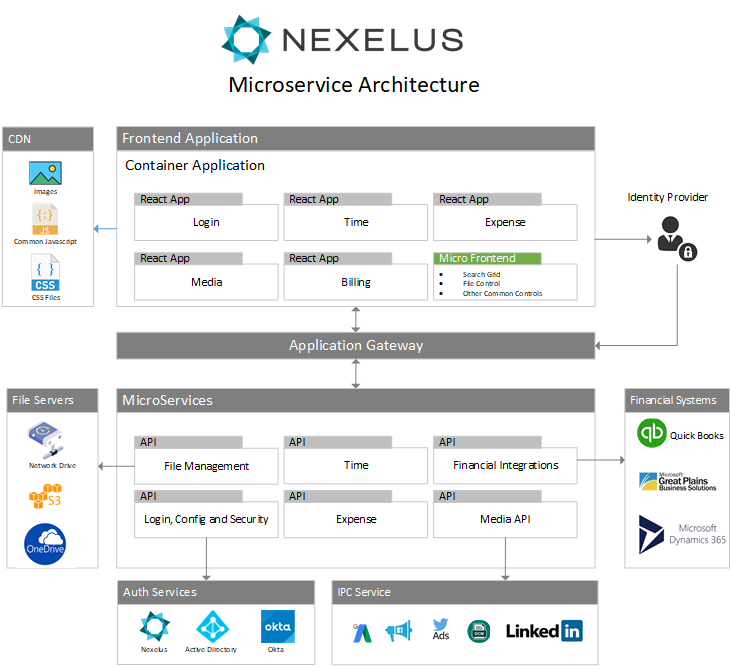


#### Proposed Architecture - (Micro Services with Micro Front Ends)

Microservices are a set of services that act together to make a whole application operate. This architecture utilizes APIs to pass information, such as user queries or a data stream, from one service to another.

How the underlying software works, or which hardware the service is built upon, depends solely on the team who built the service. This makes both communicating between teams and upgrading services very dynamic—even reactive—allowing a software company or team to be more resilient in its development.

React provides state-of-the-art functionality and is an excellent choice for developers looking for an easy-to-use and highly productive JavaScript framework. Using React, you can build complex UI interactions that communicate with the server in record time with JavaScript-driven pages.



## Research Areas

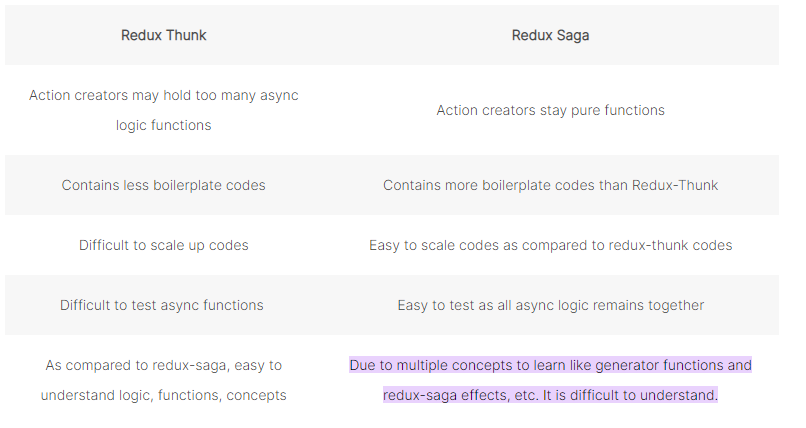
### Communication of Micro Front Ends.

We are using MF apps to communicate with each other using webpack concepts.

#### State Management for Front end Application.

1. In State Management of Front End Application, we are using Redux to handle this. In Redux we have two ways to handle the API requests
   1. Thunk (Mostly handled by Thunk)
   2. Saga (Better controlled Code for Complex/Enterprise Applications)

In our application we need better controlled code so we are going to use Saga for this.

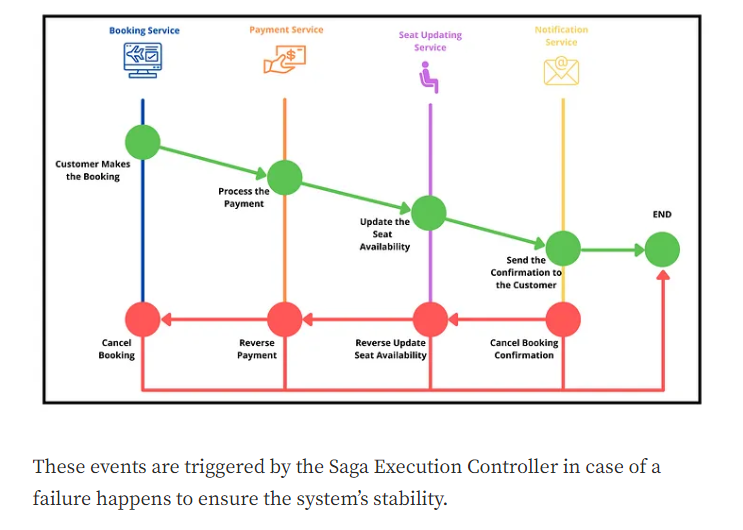


### Multi-Tanent System

We are going to have our MicroService to get request from the Application and decide which DB (enterprise) going to use depending on the URL provided by the Frontend Application.

### Messaging with in the MicroService

We have multiple options to use for Messaging inside MicroService (Rabbit MQ, Azure Bus) , Also we have Saga pattern for microservices Communications as well which handle the intercommunication with in MS App. (Below is Saga Example)



### Database Connectivity

Nexelujs Web Forms is using ADO.net for database connectivity.

It is proposed to use Entity Framework for new Microservices Application

### Authentication Service

JWT Vs OAuth Vs OpenID VS Identity Server [ Write a little details on each]

JWT is proposed for token base authentication.

### API Gateway

(Istio / Ocelet)

Proposed solution is to use Istio as it’s a service mesh. It also provides monitoring services.

### Cache

Radis Cache

### Logging

Logging of .Net Core MicroServices

#### Nlog vs Serilog vs Elmah.

Serilog is preferred to be used for Logging in MicroServices app.

Reference URL : <https://stackify.com/nlog-vs-log4net-vs-serilog/>