**Car Selling Website**

Software Requirement Specification

Version:1.0

**Sep 09, 2022**

# 

**Author: Tam Phan**

**Security Classification: Confidential**

110 Bishopsgate, London EC2N 4AY

Tel: +44 (0)20 7333 0033

Email:



Table of Contents

[Document History 1](#__RefHeading___Toc5975_2283222772)

[1. Overview 1](#__RefHeading___Toc5977_2283222772)

[1.1 Purpose 1](#__RefHeading___Toc5979_2283222772)

[1.2 Business Objectives 1](#__RefHeading___Toc5981_2283222772)

[1.3 Scopes 1](#__RefHeading___Toc5983_2283222772)

[1.4 Constraints 1](#__RefHeading___Toc5985_2283222772)

[2. Architectural Goals 2](#__RefHeading___Toc5987_2283222772)

[2.1 High-Level Architecture 2](#__RefHeading___Toc5989_2283222772)

[2.2 Services communicate via Kafka 4](#__RefHeading___Toc5995_2283222772)

[2.3 Continuous Integration and Continuous Delivery 4](#__RefHeading___Toc5997_2283222772)

[3. Technology Selection 5](#__RefHeading___Toc6003_2283222772)

[3.1 Advantages of NextJS in This Solution 5](#__RefHeading___Toc6005_2283222772)

[3.2 Advantages of .Net 6 in This Solution 5](#__RefHeading___Toc6074_2283222772)

[4. Deployment Model 7](#__RefHeading___Toc6007_2283222772)

[4.1 Development Environment (on promise and cloud) 7](#__RefHeading___Toc6076_2283222772)

[4.2 Staging / Production Environment 7](#__RefHeading___Toc6078_2283222772)

# Document History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Version History** | | | | | |
| **Version** | **Effective Date** | **Author** | **Details** | **Reviewer** | **Approvers** |
| 1.0 | Sep 09, 2022 | Tung Nguyen | Initialize | All team members | Le Cao |

Confidentiality

This document is distributed on a restricted basis, is commercial in confidence to the recipient, and may not be used for any purpose other than that associated with a NashTech project. The contents of this document may not be disclosed to any third parties without the expressed advance written authorisation of NashTech.

# Overview

## Purpose

This document describes the proposed architecture (i.e., High-Level Design) for the project “Car Selling Website” and how various modules work in cohesion with each other.

The current solution utilizes all the benefits of NextJs technologies, ASP.NET 6 and is deployed by cooperating with the CI/CD team to bring the site living on the internet by using Docker and Kubernetes.

## Business Objectives

The NextJs workshop will build a website using the NextJs and .Net 6 frameworks. The website allows users to do the following once it is launched

* Search for a car by brand, model, year, name, or category
* View details information of a car
* Order a car
* Receive an email after ordering
* Register to sell a car

## Scopes

The scope of the project is:

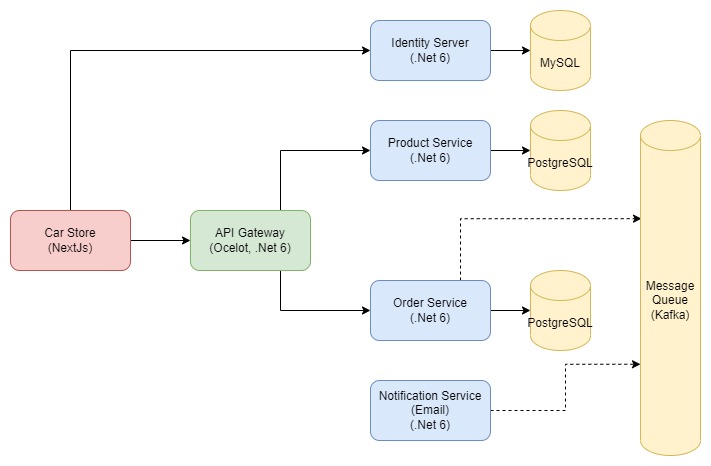
* Build a client website that has the user interface built in NextJs.
* Build microservice backend including some basic features like authentication, authorization, and CRUD.
* Configure variables exposing hosts and ports and other things that are required from CI/CD teams to communicate among services in the systems.
* Set up Azure Storage Account, Blob to store static car images to display in the user interface.

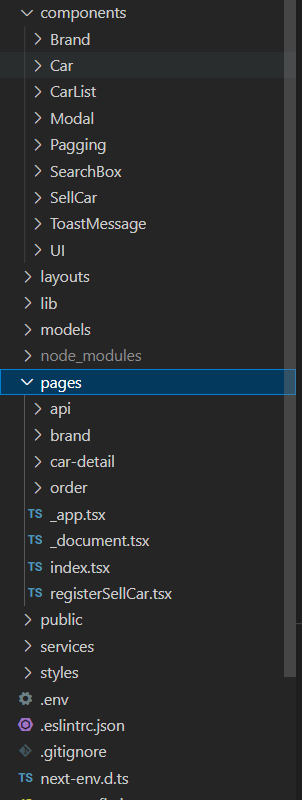
## Constraints

* UI: developed by NextJs.
* Backend:
  + Based on microservices architecture
  + IdentityServer4
  + ASP.NET 6
  + Kafka
* Deployed using Docker and Kubernetes.

# Architectural Goals

## High-Level Architecture



* + 1. **Client Website**
* This client website is structured by HTML, CSS, Typescript and the required JavaScript framework, NextJS.
* The front end is integrated with Material UI that helps us to build responsive pages that make users feel more comfortable with the UX.
* The most important thing when structure a NextJS project is that every under the “pages” folder must be a real page. Things that build up the page should be under the “components” folder.  
  + 1. **API gateway**

In the microservice architecture pattern, regarding communication of the clients with the microservices, we usually try to avoid having direct client-to-microservice communication. That has some drawbacks such as the mismatch between the needs of the client and the fine-grained APIs exposed by each of the microservices or web-unfriendly protocols. That is why we have a much better approach is to use an API Gateway that is a server that is the single-entry point into the system.

In our system, we use Ocelot API Gateway as a gateway in our design. API Gateway will help us to abstract the under system and make it both developers and end-users easily to develop, use the system.

* + 1. **Microservices**
* Every service has its own database
* They can communicate by calling each other directly or via the Kafka
* If there is a request coming from the web browser, the API gateway act as a receiver to receive the request first and then it will look for the corresponding service to query the data and send it back to the web to render the result to the end-user.

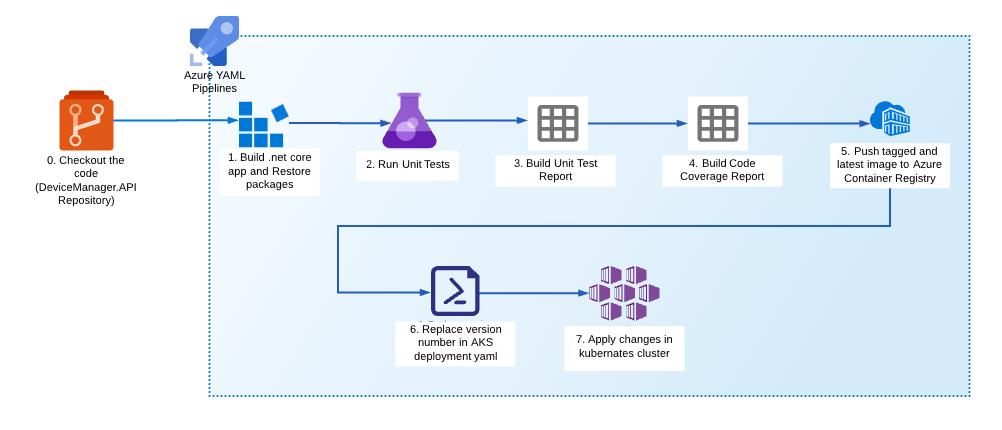
## Services communicate via Kafka

As shown in the overal architecture picture, we use Kafka to handle asynchronous communication between services. That provide us two big benefits:

* Loose coupling: services do not have to know or wait for each other
* Fault isolation: if the receiving service fails for any reason, the sender can still send messages to Kafka without interruption. The receiving service can pick up and process the messages as soon as it's ready to do so.

## Continuous Integration and Continuous Delivery

In this process, the CI/CD team is responsible for checking out the code hosting in the GitHub repository and then setting up a pipeline that can build a .NET app and restore packages. Then it will run unites and ensure everything in green. After that, it will push tagged and latest image to Azure Container Registry replacing version number in AKS deployment YAML. Lastly, all changes are applied to the Kubernetes cluster.



# Technology Selection

The following section examines the reasons behind the selection of technologies put forward in the proposal.

## Advantages of NextJS in This Solution

Next.js is a robust JavaScript framework builds upon React, a JavaScript library that makes building frontend user interfaces (UI) quick and efficient.

* + 1. **Pre-Rendering Options**
* **Static page generation**
  + Static page generation pre-renders a page's HTML at build time, which means the pages have already made before users access the site. The static pages do not need a backend or database to function. Uses just requests the page and get it. The pages can also be cached on a CDN.
  + So, the advantage of static pages is that they load faster to the users and more secure because there’s no dynamic code running on a server that can be exploited.
* **Server-side rendering**
  + For pages that display frequently updated data or personalize content for specific users, we will use server-side rendering instead. With this option, the server will generate that page's static HTML on each request made by the client.
    1. **SEO advantages**

Pre-rendering a page, and its corresponding data, help the web application's SEO rankings. This is because web crawlers can access the HTML, the data contained on a pre-rendered page, and because the load time is faster.

* + 1. **File-based routing**

This is for developers. The built-in file based page routing offered by Next.js eliminates the need to import a routing library typically required when working with a React application. The file-based page structure is great for developer experience as it is intuitive, and eliminates the need for explicitly defining routes within the code itself.

## Advantages of .Net 6 in This Solution

* + 1. **Cross-Platform**

.NET 6 is a free and open-source framework for developing cross-platform applications targeting Windows, Linux and macOS.

* + 1. **Performance**

“.NET 6 is the fastest full stack web framework” – Microsoft Docs.

* + 1. **Ease of maintenance**

Standardising on .Net 6 allows us to minimise the number of languages in the solution which in turn makes it easy to support. It is a widely used technology that also makes it easy for us to research and find out solutions for the problems facing in the coding process.

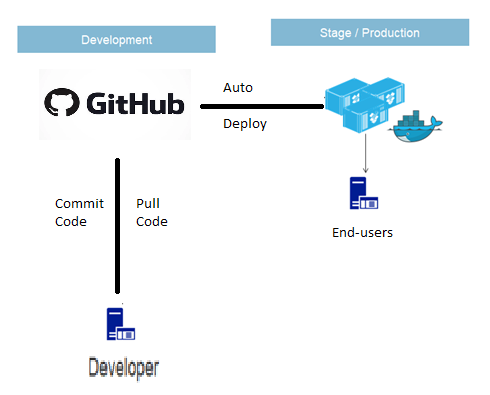
* + 1. **Longevity**

Microsoft sees this as it’s the language of choice for the future. The long roadmap and steady investment by Microsoft along with a vibrant open source community around the technology means will result in the possibility of Kaplan taking advantage of these developments over the lifecycle of the product.

* + 1. **Secure**

Websites need security features to handle the increased demand, and Microsoft's .Net platform emphasizes these aspects. The common language runtime and .NET provide many useful classes and services that enable developers to write secure code, use cryptography, and implement role-based security.

# Deployment Model

Describe the deployment model for production, Development environments. Also if CI/CD is applicable then mention it here to bring benefits to the development team and client as well.

## Development Environment (on promise and cloud)

The team will develop their local environment by interacting with the GitHub repository to commit or pull code and then make modifications to the source code.

## Staging / Production Environment

The staging environment is where users can test the features that are deployed from the development environment. The staging environment is hosted by CI/CD servers.

A close up of a logo

Description automatically generated

**Author**

**Author**

For more details, please send your enquiry to [info@nashtechglobal.com](mailto:info@nashtechglobal.com)

or visit our website [www.nashtechglobal.com](https://nashtechglobal.com/)

[@NashTech](https://twitter.com/NashTechHN)HN

[NashTech Limited](https://www.linkedin.com/company/nashtech-global/)

We are experts in technology, delivering smart solutions that solve business challenges and create value. Our award-winning teams apply deep expertise and passion to deliver complex IT projects globally.