Nuxt3 mit Vuetify 3 – Teil 1: Installation

Wenn man aus der Microsoft Welt kommt und Webseiten mit JQuery bearbeitet hat, dann sind UI Framworks wie Vue eine wahre Veränderung. In meiner ASP WebApp habe ich Vue2 über CDN eingebunden und Vue anstatt JQuery genutzt. Vue funktioniert aber am besten mit NodeJs. Der Versuch NodeJs zu erlernen verlief aber immer in Frust. Weil es keinen roten Faden gibt, wie etwas umgesetzt werden kann. Es gibt keinen besten Weg. Es gibt sehr viele neue Begriffe und Definitionen, die Stück um Stück erlernt werden müssen. Es ist definitiv ein anderes lernen wie eine erprobte Sprache wie C# oder Java.

Deshalb ist hier **mein** roter Faden. Ziel soll am eine App sein, die folgende Punkte mit sich bringt:

1. Nuxt 3 als Wrapper über Vue3. Nuxt 3 schafft eine Ordnerstruktur und macht die imports z.B. überflüssig.
2. Vuetify 3 Beta. Zwar ist es noch in Beta Phase, aber es wird das beste UI Framework (Meiner Meinung nach)
3. Typescript: Die App soll in Typescript geschrieben sein
4. Pinia wird unser Statemanagement anstatt Vuex
5. Übersetzungen werden über i18n realisiert
6. Daten werden über externe .NET 6 Minimal Api mit Swagger abgerufen
7. Authorisierung erfolgt über eine ASP .NET 6 Proxy Anwendung

Installation

Voraussetzung

NodeJS, Visual Studio Code, Volar Extension. Aktuelle Links

sind hier zu finden:

Nuxt 3 – Quick Start (nuxtjs.org)

In Visual Studio über „Strg + Shift + ö“ das Terminal öffnen

Neue App erstellen:
[crayon-6767e837624c0478943153/]
Die App wird nun im Unterverzeichnis „nuxt-app“ erstellt. Daher müssen wir in das neue Verzeichnis navigieren
[crayon-6767e837624c8594613939/]
packages nachinstallieren:
[crayon-6767e837624c9917489885/]

Vuetify installieren

Referenzen:

The Easy Way to add Vuetify Vue.js 3 Material Components to Your Nuxt 3 / Vue.js 3 Application – YouTube

[Solved] vuetify icon not showing – Local Coder (Solution 5)

vuetify – npm (npmjs.com)

Unter „Version“ die neuste Version raussuchen
[crayon-6767e837624ca370638450/]
Sass muss ebenfalls für Vuetify installiert werden
[crayon-6767e837624e2879264971/]
Pinia als StateManager
[crayon-6767e837624e4360039264/]
[crayon-6767e837624e5869464382/]
und die mdi Icons für Vuetify
[crayon-6767e837624e6017661042/]
Plugins werden in Nuxt im plugins Verzeichnis erstellt. Also muss im Root ein neues Verzeichnis „plugins“ erstellt

In das plugins Verzeichnis wird eine neue Datei „vuetify.ts“ erstellt und folgendes kopiert:

[crayon-6767e837624e8791271586/]
Jetzt muss noch die nuxt.config.ts angepasst werden und das css und der transpiler müssen mit vuetify verheiratet werden. Daher kann der folgende Teil einfach copy/pasted werden. Der Vite Teil wird bei mir als Fehler angezeigt. kompilieren geht trotzdem. Wem das stört, kann den vite Knoten auskommentieren
[crayon-6767e837624e9842170798/]
Jetzt sollte die App starten können:

In der Beta 1 läuft dauert der Start über 1 Minute. Ich hoffe das wird noch gefixt
[crayon-6767e837624ea111333465/]
Weiter geht's mit Teil 2 Layout: https://devandy.de/?p=890&preview=true

---

Template parsen in C#

Ok habe ich mir gedacht. Es wird ja wohl nicht so schwer sein folgenden Code zu parsen und die Templates entsprechend zu ersetzen:
[crayon-6767e83762a13986724241/]
vue.js oder mustache verwendet eine ähnliche Syntax.

Ich könnte an dieser Stelle mustache für C# verwenden und er würde mir das Template ersetzen, allerdings benötige ich nicht nur das Ergebnis als String sondern ich möchte bei jedenm Iterationsaufruf eingreifen können.

Bisher habe ich mir folgende Tools angeschaut:

- Mustache (Stubble): GitHub – StubbleOrg/Stubble: Trimmed down {{mustache}} templates in .NET
- Antlr4: antlr4/index.md at master · antlr/antlr4 · GitHub

- RegEx (Möchte ich aus performance Gründen nicht nutzen)
- Sprache: GitHub – sprache/Sprache: A tiny, friendly, C# parser construction library
- Superpower (Erweiterung von Sprache mit besserer Fehlerausgabe) GitHub – datalust/superpower: A C# parser construction toolkit with high-quality error reporting
- Handlebars: GitHub – Handlebars-Net/Handlebars.Net: A real .NET Handlebars engine

Leider ist Sprache sehr schlecht dokumentiert und es fehlen Beispiele wie man das umsetzen könnte. Dennoch würde ich es gerne bevorzugen.

Vue – Konstanten definieren

Konstanten kann man in Vue einfach im Created Hook definieren.

Generell sollte man sich das Arbeiten mit Konstanten angewöhnen.

if(UserTypeId == 7) {…} stellt den Entwickler vor die Frage, was an diesen UserType nun so besonders ist.

Besser: if(UserTypeId == USERTYPE_GUEST) . So weiß jeder, die folgende Logik betrifft den Gastbenutzer
[crayon-6767e83762b83327578549/]

Vue – Directive

Mal angenommen man möchte alle Felder auf einer Seite deaktivieren. Man könnte input für input durchgehen und mit :disabled=“isEditable“ das Feld deaktivieren.
Oder man schreibt eine Custom Directive, die diese Logik abbildet: https://vuejs.org/v2/guide/custom-directive.html

Möchte man das global haben:
[crayon-6767e83762d2b090116915/]
Oder per Instanz:
[crayon-6767e83762d2f534083194/]
Eingebunden wird das nun mit.
[crayon-6767e83762d31369413958/]
oder
[crayon-6767e83762d32848159592/]
oder ohne Bedingung
[crayon-6767e83762d33678905958/]

Cmd Pfad autocomplete funktioniert nicht

Cmd öffnen
[crayon-6767e83762ec1461224761/]
Befehle eingeben
Cmd neustarten und mit cd Tab den Pfad autocompleten lassen

Applikationsdesign: Datensätze abarbeiten nach Punktesystem

Mir kam mal folgende Problemstellung in die Gedanken:

Mal angenommen ich muss unterschiedliche Aufgaben abarbeiten. Diese befinden sich in einer Warteschleife. So dass sie nacheinander oder auch asynchron parallel abgearbeitet werden sollen.

Jede Aufgabe besitzt eine Priorität. Sagen wie 1 (unwichtig) – 10 (sehr wichtig). In die Warteschleife kommen immer wieder Aufgaben nach. Das Problem ist, es werden die sehr wichtigen immer zuerst abgearbeitet. Die weniger wichtigen kommen daher nie dran und werden nie verarbeitet. Die Lösung ist ein Punktesystem mit Zeitangabe. Die Priorität ist dient dann als Faktor und wird mit der Anzahl Minuten multipliziert. Das ergibt die Summe der Punkte pro Datensatz

Beispiel: Es ist Momentan 12:00 Uhr Mittags.

1. Datensatz 1: Prio 4, Hinzugefügt um: 11 Uhr, Punkte: 4 * 60 = 240
2. Datensatz 2: Prio 9, Hinzugefügt um: 8 Uhr, Punkte: 9 * 240 = 2160
3. Datensatz 3: Prio 2, Hinzugefügt um: 4 Uhr, Punkte: 2 * 480 = 960

Sortiert man diese Liste nun nach Punkten ergibt sich eine neue Reihenfolge

1. Datensatz 2: Prio 9, Hinzugefügt um: 8 Uhr, Punkte: 9 * 240 = 2160
2. Datensatz 3: Prio 2, Hinzugefügt um: 4 Uhr, Punkte: 2 * 480 = 960

3. Datensatz 1: Prio 4, Hinzugefügt um: 11 Uhr, Punkte: 4 * 60 = 240

So hat jeder Datensatz eine Chance drangenommen zu werden. Je länger er in der Warteschleife verweilt, desto höher wird sein Punktesatz.

---

system-versioned temporal tables oder automatische Historie auf Tabellenbasis

funktioniert ab SQL Server 2016

[crayon-6767e83763026602023178/]

SQL Column löschen inkl. aller Constraints

[crayon-6767e837631b1153596539/]

Windows Dienst installieren / löschen

sc.exe create MyServiceName binpath= „C:\inetpub\wwwroot\….exe“
sc.exe delete MyServiceName

Bibliotheken für Authorisierung

Das Thema Authorisierungen ist sehr komplex und es gibt dazu viele Ansätze. Hier ist eine Liste von Bibliotheken:

Casbin

Vergleich Casbin mit OPA: OPA vs Casbin (github.com)

https://www.openpolicyagent.org/

oso Documentation – oso Documentation (osohq.com) (Nur Python und Java)

ory/ladon: A SDK for access control policies: authorization for the microservice and IoT age. Inspired by AWS IAM policies. Written for Go. (github.com) (Veraltet)

teramoby/speedle-plus: Speedle+ is an open source project for access management. It is based on Speedle open source project and maintained by previous Speedle maintainers. (github.com)