

BANanoVuetifyAD3 for Dummies

Table of contents

Introduction	4
Setting up the IDE	4
Back-Ends	12
Facts	12
Tutorials	12
Part 01	12
Part 02	13
Part 03	13
Part 04	13
Part 05	13
Part 06	13
Part 07	13
Part 08	14
Part 09	14
Part 10	14
Part 11	14
Part 12	14
Part 13	15
Part 14	15
Part 15	15
Part 16	15
Part 17	15
Part 18	15
Part 19	16
Part 20	16
Part 21	16
Part 22	16
Part 23	16
Part 24	17
Part 25	17
Part 26	17
Part 27	17
Part 28	17
Part 29	17
Part 30	18
Part 31	18
Part 32	18
Part 33	18
Part 34	18
Part 35	19
Part 36	19
Part 37	19
Part 38	19
Part 39	19
Part 40	20
Part 41	20

Part 42	20
Part 43	20
Part 44	20
Part 45	20

Introduction

Welcome to **BANanoVuetifyAD3 aka BVAD3**

- [BANanoVuetifyAD3](#) is a B4J (Basic4Java) library that helps one create Web Apps / Web Sites using BANano and Vuetify. It is the first VueJS UX based library for BANano. This library is created by Anele Mbanga. It seeks to bring the power of Vuetify to BANano. With BVAD3 one codes their UX to life and then using BANano functionalities, one can build and then publish their web application.
- [B4J](#) is created by Anywhere Software. With it one is able to write Java applications in B4X, a VB (Visual Basic) like syntax code base and it produces native java apps that can run on Windows, Linux and Mac.
- [BANano](#) is created by Alain Bailleul. This helps anyone create websites and or webapps using VB-syntax. It generates pure JavaScript, CSS and HTML for the website/webapp. Apps created with it are SPAs (Single Page Applications) and or PWA(Progressive Web Apps) with the inclusion of web service workers (optional). BANano itself is UX framework independent and this means one can use their own framework of choice.
- [Vuetify](#) is a complete UI framework built on top of Vue.js. The goal of the project is to provide developers with the tools they need to build rich and engaging user experiences. Unlike other frameworks, Vuetify is designed from the ground up to be easy to learn and rewarding to master with hundreds of carefully crafted components from the Material Design specification

Things to remember

- BANanoVuetifyAD3 = **BVAD3**
- Single Page Applications = **SPA**
- Progressive Web App = **PWA**
- Basic4Java = **B4J**
- Visual Basic = **VB**

Things about Anywhere Software

1. There is a video based learning channel. Check it [here](#).
2. With their [B4A](#) (Basic4Android) Now FREE, one can create completely native Android Apps.
3. With their [B4i](#) (Basic4iPhone), one can create completely native iPhone, iPad Apps
4. There are some coding [booklets](#) that have been written that can help you out with the B4X eco-system, thanks to Klaus.

Let us prepare our development environment first

Created with the Personal Edition of HelpNDoc: [Full-featured EBook editor](#)

Setting up the IDE

Developing on Windows PC

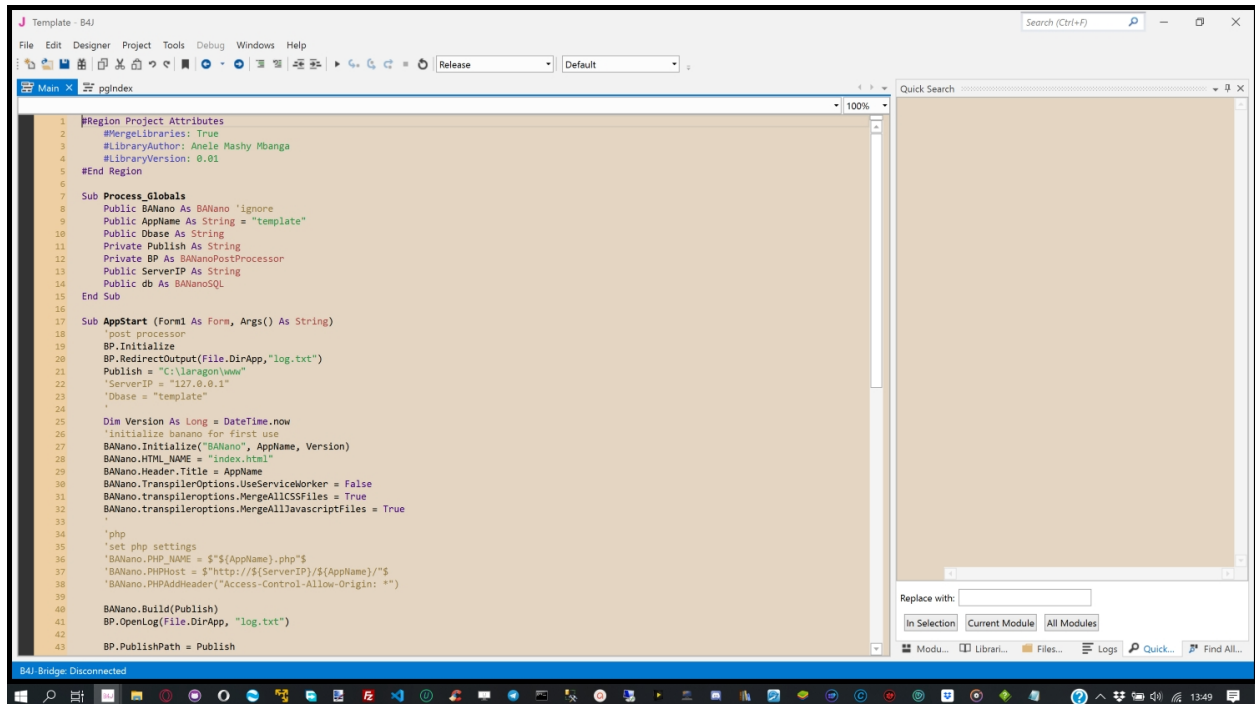
At the time of writing B4J only runs on Windows PC.

To be able to develop BVAD3 apps, you will need the following: You can click on each to download

[1. B4J](#)

There are instructions on the website on how to install and configure B4J. With it you will need Java JDK

8+. Just follow the instructions on how to set up your IDE and get it ready.



Create a folder structure: You can skip this step if your IDE is already set up

1. Create a folder named B4X in your C: drive, and then create the respective sub folders

C:\B4X\B4J\Shared - we will call this folder "**shared**"

C:\B4X\B4J\Libraries - we will call this folder "**external libraries**"

C:\B4X\B4J\Workspace - we will call this "**workspace**"

- The **shared** folder will store all code modules that have sharable code
- The **external libraries** will store all libraries from others users e.g. BANano & BVAD3 library.
- The **workspace** will store your project folders, e.g. projects we will create with BVM

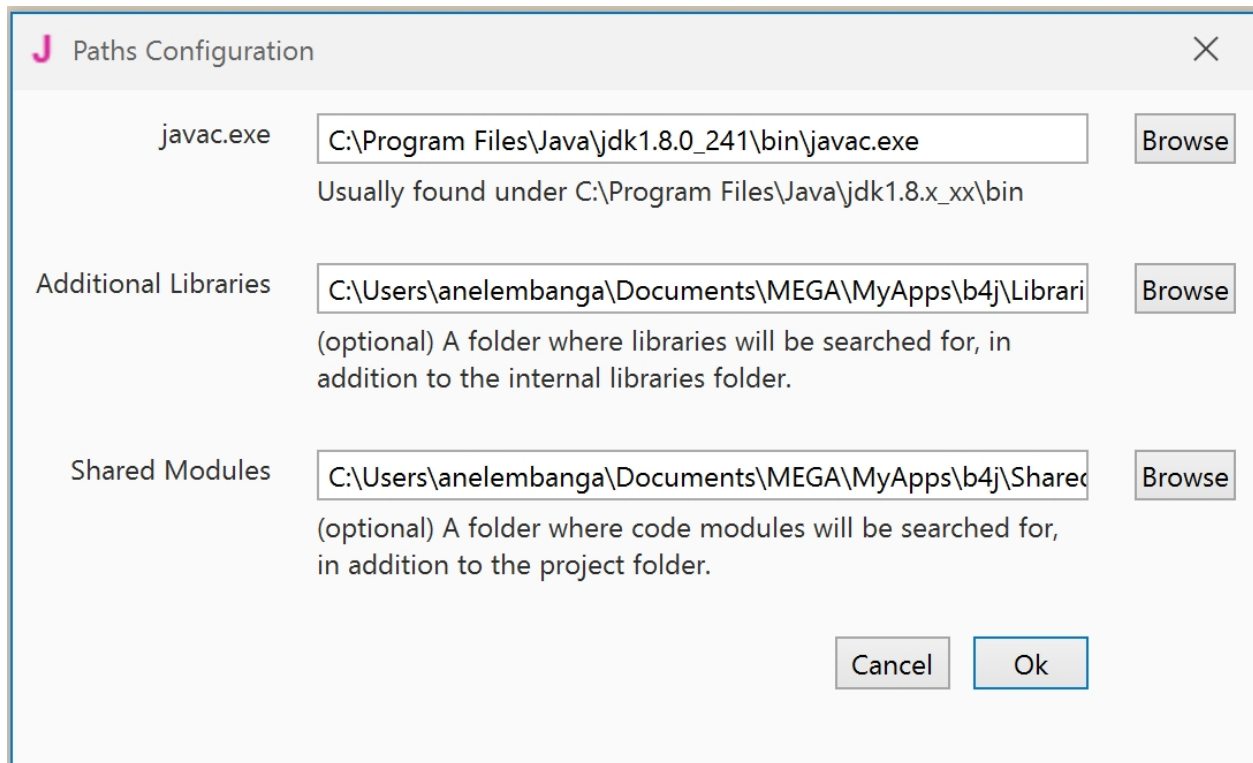
We have B4X master folder because we can create the same structure for **B4A** and or **B4I** IDEs.

Test the readiness of your IDE

To test the readiness of your IDE, we will do 3 things.

1. Start B4J, in the menu click Tools > Configure Paths. A screen like this will appear.

Figure 2



- Ensure that the specified paths point to the correct locations.
- Additional Libraries should point to your **external libraries** path you created before
- Shared Modules should point to the **shared** path you created before. You click Ok to save any changed details.

2. BANano

Once downloaded, copy the contents of the **Library** folder to your external libraries folder e.g.C:\B4X\B4J\Libraries This library comes with some code examples on the usage of BANano. I have also written a nice [tutorial](#) on how one can get started with BANano. That will help you with the basics and also further experience on how to use BANano. As an example, one of the things you will see when writing BVAD3 code is the **BANanoEvent**.

NB: I greatly recommend that you go through this tutorial so that at least you have some understanding of BANano and what it does.

3. BANanoVuetifyAD3

Download the github repo and extract the contents to your working folder, e.g. C:\B4X\B4J\Workspace Open the Library folders inside BVAD3, double click the BANanoVuetifyAD3.b4j file to open b4j. Run the project. This depends on #2 above. Close the project. Your library will be compiled.

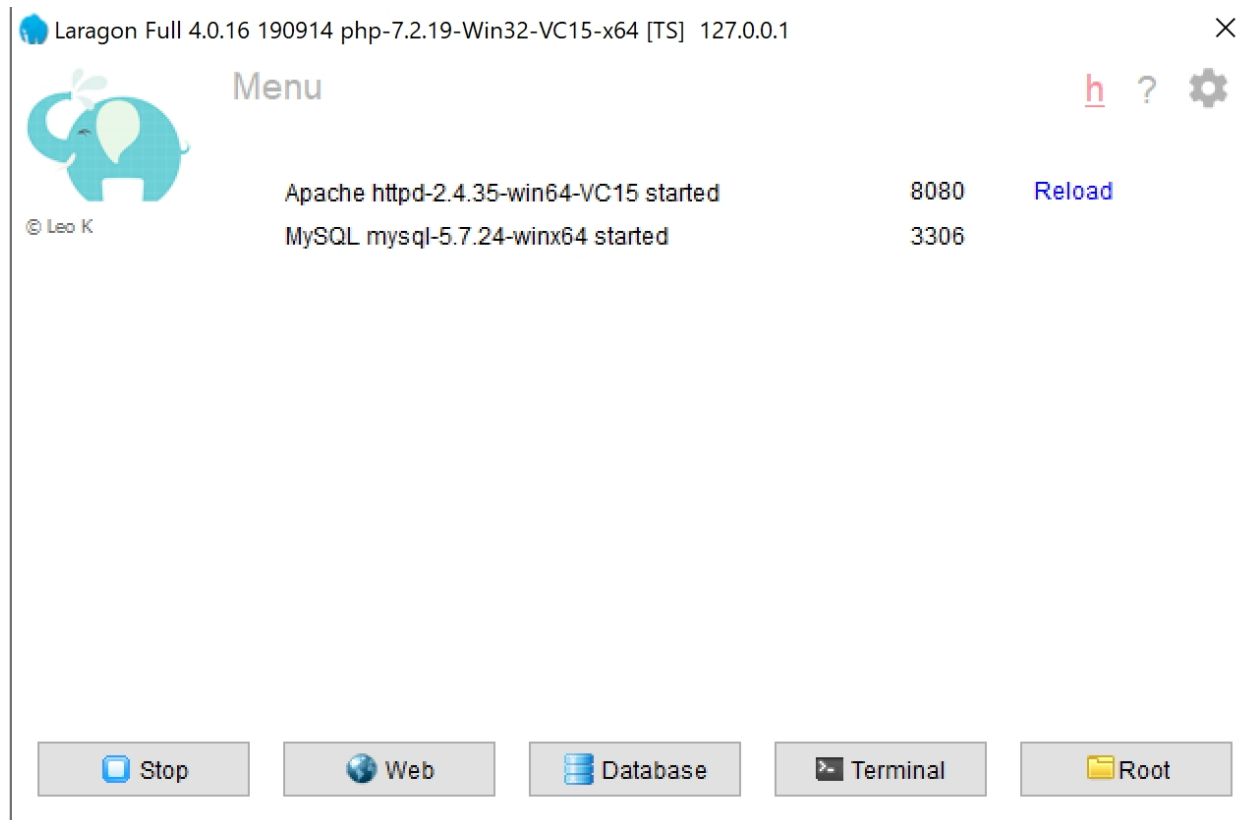
The structure of the BVAD3 github repo.

1. Library - this contains the source code for the BVAD3 b4x library.
2. Demos - a collection of demo projects created with BVAD3
3. Templates - various BVAD3 templates

I am assuming you have briefed yourself about BANano (my tutorial and others) and now you are ready to explore BVAD3 code and its output. Going forward we will use our VB know how to create apps.

4. A webserver. I am using the [laragon](#) development web server for all my examples here. One can also use [XAMPP](#)

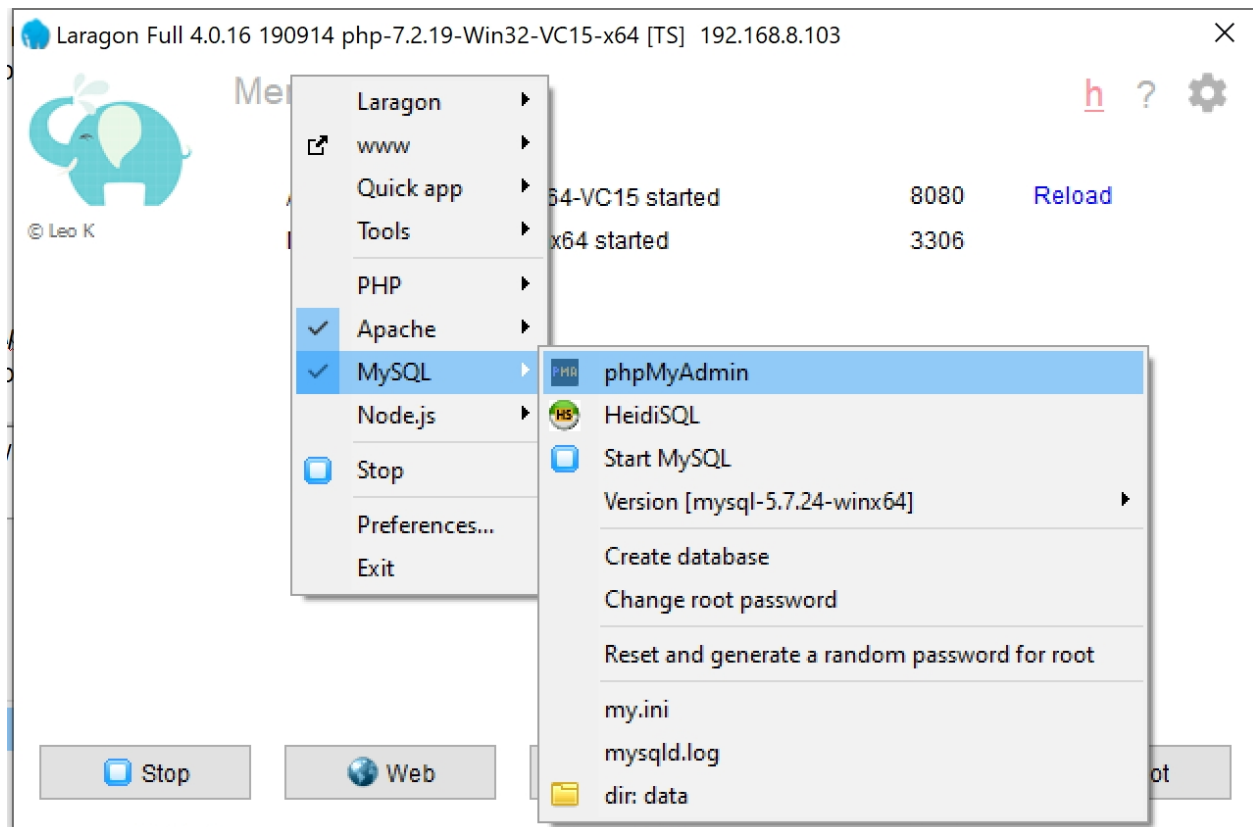
I am yet to test the [USBWebServer](#).



MySQL Usage

- Laragon does not come installed with phpMyAdmin, thus, [download phpMyAdmin](#)
- Extract the folder to c:\laragon\etc\apps**phpMyAdmin**
- The password is **root**.

Check that phpMyAdmin works



MSSQL Usage

[Install PHP drivers for MSSQL](#)

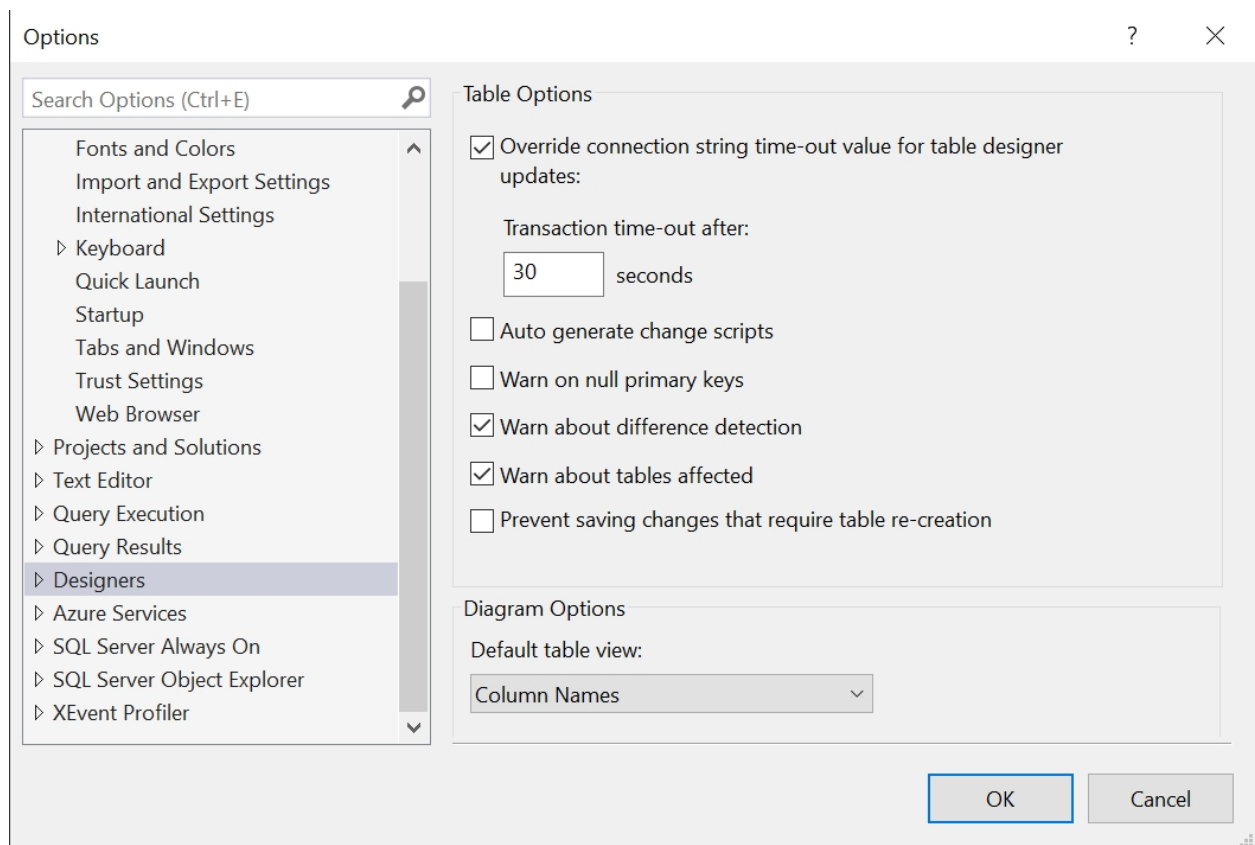
- Extract the files to `C:\laragon\bin\php\php-7.2.19-Win32-VC15-x64\ext`. This is the php extensions folder
- Activate the nts (non-thread-safe option)

This is an example of the config file for MSSQL.

```
<?php
const DB_HOST = '(local)\sqlexpress';
const DB_NAME = 'test';
const DB_USER = 'root';
const DB_PASS = '';
?>
```

Note that the instance name has been specified.
Also ensure that TCP/IP is enabled for SQL server config.

If you use the Management Studio for MSSQL, you need to update the Tools > Options > Designer to be able to make table changes using the designer.



The user for the config.php file should be given read and write access to the database. For example.

Login Properties - root

Select a page

General

Server Roles

User Mapping


Securables

Status


Connection

Server:
DESKTOP-U7U5UJ5\SQLEXPRESS

Connection:
DESKTOP-U7U5UJ5\mbang

 [View connection properties](#)

Progress

 Ready

Script Help

Users mapped to this login:

Map	Database	User	Default Schema
<input type="checkbox"/>	master		
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		
<input checked="" type="checkbox"/>	test	root	dbo

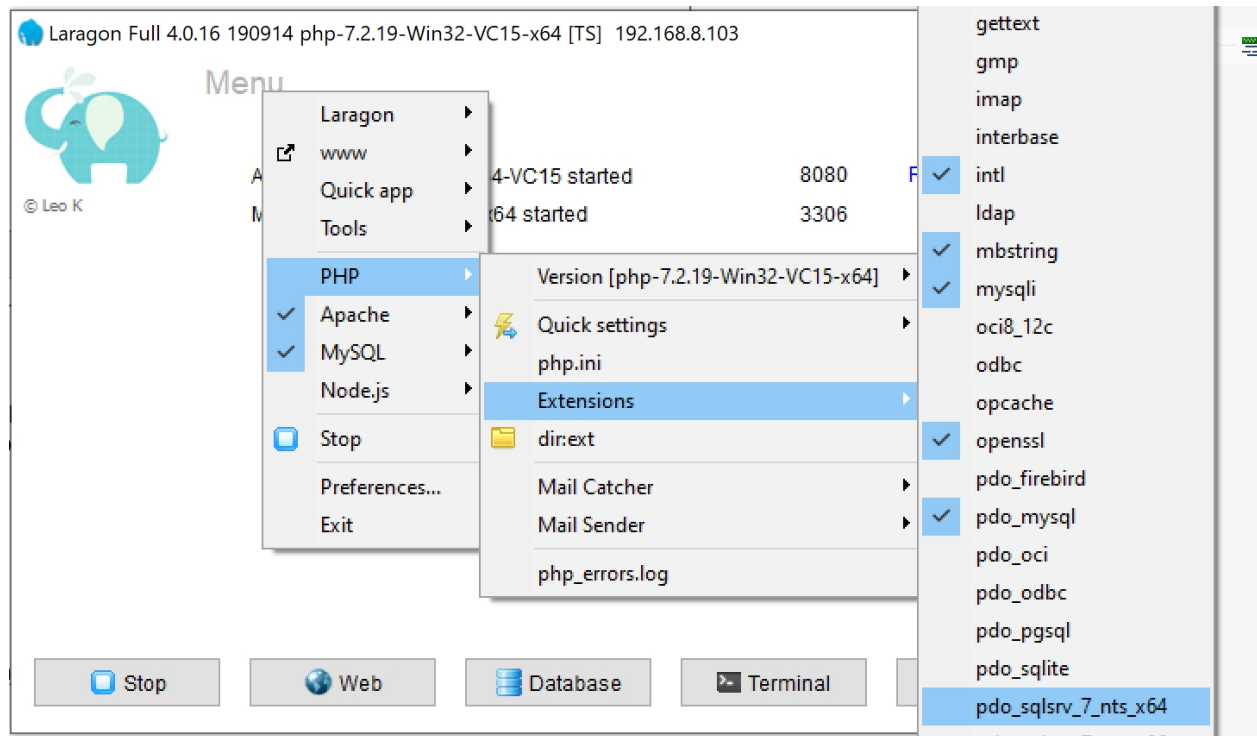
☐ Guest account enabled for: test

Database role membership for: test

<input type="checkbox"/>	db_accessadmin
<input type="checkbox"/>	db_backupoperator
<input checked="" type="checkbox"/>	db_datareader
<input checked="" type="checkbox"/>	db_datawriter
<input type="checkbox"/>	db_ddladmin
<input type="checkbox"/>	db_denydatareader
<input type="checkbox"/>	db_denydatawriter
<input type="checkbox"/>	db_owner
<input type="checkbox"/>	db_securityadmin
<input checked="" type="checkbox"/>	public

OK

Cancel



Internet Information Server

1. [Install Web Platform Installer](#)
2. [Install PHP Manager](#)

Install IIS from WPI.

2.5. Install an FTP tool

I am using [FileZilla](#) to upload my BVM apps to the interweb. The output of your website, will be saved to the folder that you told banano to publish on.

This structure will follow this pattern.

www > template >				
Name	Date modified	Type	Size	
assets	2020/04/25 14:10	File folder		
fonts	2020/04/25 14:10	File folder		
scripts	2020/04/25 14:10	File folder		
styles	2020/04/25 14:10	File folder		
favicon.ico	2020/04/25 14:09	Icon	15 KB	
index.html	2020/04/25 14:10	Opera GX Web Docu...	4 KB	
manifest.json	2020/04/25 14:10	JSON Source File	1 KB	

Explaining the folders

1. assets - this stores all assets for the app e.g. images, json, and other files
2. fonts - (optional for storing font files)
3. scripts - this folder has all your .js files
4. styles - this folder has all your .css files

Back-Ends

By default, when creating apps with the designer, BANanoSQL is the default backend. You can change your backend so that your app works with:

1. BANanoSQL ([IndexedDB](#) via [AlaSQL](#))
2. [SQLite](#)
3. [MySQL](#)
4. [MSSQL](#)
5. [FireBase](#)

For the first 4, we have created a library called BANanoVueConnect and for Firebase we have created a library called [BANanoFirestoreDB](#). You can check the MealPrep demo project on how Firebase storage was used.

Facts

Important things to know:

- Referencing state in BVAD3 should always be in lowercase e.g. {{ anotherone }} and DO NOT use {{ anotherOne }}
- States CANNOT be hyphenated e.g. "my-name" should be "myname"
- States CANNOT have spaces or special characters
- **VModel** should not have spaces.

A good example would be

Set the state...

```
Dim items As List = vuetify.NewList
items.Add("Anele Mbanga (Mashy)")
vuetify.SetData("items", items)
```

Get the state

```
Dim items As List = vuetify.GetData("items")
Log(items)
```

Tutorials

[Our blog](#)

Part 01

[Youtube Link](#)

[Source Code](#)

This is a skeleton project

Part 02

[Youtube Link](#)

[Source Code](#)

Adding navigation bar, hamburger, spacer, and a button. Binding abstract designer components to vuetify app.
Firing events.

Created with the Personal Edition of HelpNDoc: [Create HTML Help, DOC, PDF and print manuals from 1 single source](#)

Part 03

[Youtube Link](#)

[Source Code](#)

Using individual blocks, we build and run our app. We create a dynamic title for our toolbar title and change this on button click by updating its state.

Created with the Personal Edition of HelpNDoc: [Free CHM Help documentation generator](#)

Part 04

[Youtube Link](#)

[Source Code](#)

We start with a blank template and create routers, load layouts to the router components and link these to the vuetify app. For more details of how routers are used, see the MealPrep app in the Demos folder.

Created with the Personal Edition of HelpNDoc: [Easily create EBooks](#)

Part 05

[Youtube Link](#)

[Source Code](#)

Based on our previous example, we create a menu that is activated by a button link. We also set an active class for each menu item being selected.
We apply a transition to the menu and link routers per menu item. We use state binding for the menu items.

Created with the Personal Edition of HelpNDoc: [Full-featured Kindle eBooks generator](#)

Part 06

[Youtube Link](#)

[Source Code](#)

We continue from part 5 and add a logout button and an empty navigation drawer with a background image. We will add navigation items in part 7.

Created with the Personal Edition of HelpNDoc: [iPhone web sites made easy](#)

Part 07

[Youtube Link](#)

[Source Code](#)

We continue from part 6 and add a list to the drawer, use the same links we created the menu items with. We then use a v-for loop and binding to ensure each drawer item can navigate to its page.

Created with the Personal Edition of HelpNDoc: [Single source CHM, PDF, DOC and HTML Help creation](#)

Part 08

[Youtube Link](#)

[Source Code](#)

We upload our app to a webserver and run google light-speed on it.

Created with the Personal Edition of HelpNDoc: [Free PDF documentation generator](#)

Part 09

[Youtube Link](#)

[Source Code](#)

In this part we create 3 types of avatars, text, icon and image. We also create a grid layout to set these at row 1, column 1 to 3 respectively.

We also add a user profile just above the list in the navigation drawer

Created with the Personal Edition of HelpNDoc: [iPhone web sites made easy](#)

Part 10

[Youtube Link](#)

[Source Code](#)

We create different alerts and toggle visibility

Created with the Personal Edition of HelpNDoc: [Easily create iPhone documentation](#)

Part 11

[Youtube Link](#)

[Source Code](#)

We create dynamic dialogs and dynamic snackbar controls.

Created with the Personal Edition of HelpNDoc: [Create HTML Help, DOC, PDF and print manuals from 1 single source](#)

Part 12

[Youtube Link](#)

[Source Code](#)

We create an input dialog prompt.

Created with the Personal Edition of HelpNDoc: [Free help authoring tool](#)

Part 13

[Youtube Link](#)

[Source Code](#)

We create text-field layouts and feed these to the grid layout we have created. As we set v-models for each of the text-fields we call `.GetData` to read the values of the text fields.

Created with the Personal Edition of HelpNDoc: [Easily create EPub books](#)

Part 14

[Youtube Link](#)

[Source Code](#)

We create badges and increment and decrement these also changing their color. We also create a user status indicator.

Created with the Personal Edition of HelpNDoc: [Easily create EBooks](#)

Part 15

[Youtube Link](#)

[Source Code](#)

Instead of creating multiple layouts, we re-use our layouts and use `BANanoLoadLayoutArray` to load, extract and update them via code.

Created with the Personal Edition of HelpNDoc: [Produce electronic books easily](#)

Part 16

[Youtube Link](#)

[Source Code](#)

In this example, we have created date and time pickers for input. Both are placed inside a menu so that they are activated when a text field is active.

Created with the Personal Edition of HelpNDoc: [Easy CHM and documentation editor](#)

Part 17

[Youtube Link](#)

[Source Code](#)

We start our journey with `v-data-table`

Created with the Personal Edition of HelpNDoc: [Easily create Help documents](#)

Part 18

[Youtube Link](#)

[Source Code](#)

We extend our tables and add color coded chips and color coded action buttons. We link these action buttons to events, events that are passed the row being processed.

Created with the Personal Edition of HelpNDoc: [Easily create EPub books](#)

Part 19

[Youtube Link](#)

[Source Code](#)

Here we add interactive user input components to the v-data-table, these are switches, rating, progress indicators and are able to display an avatar.

Created with the Personal Edition of HelpNDoc: [Free help authoring environment](#)

Part 20

[Youtube Link](#)

[Source Code](#)

We add lazy loaded images to the table, mail to links, colored icons, and format dates and numbers with day.js and numeral.js.

Created with the Personal Edition of HelpNDoc: [Free Kindle producer](#)

Part 21

[Youtube Link](#)

[Source Code](#)

We also obfuscate our javascript files for the project for protection (run in release mode)

Created with the Personal Edition of HelpNDoc: [Free CHM Help documentation generator](#)

Part 22

[Youtube Link](#)

[Source Code](#)

We build the grid using the abstract designer during debug to experience BANano #LiveCodeSwapping. We add elements to the grid matrix using re-usable layouts, this includes a button with a click event.

Created with the Personal Edition of HelpNDoc: [Write EPub books for the iPad](#)

Part 23

[Youtube Link](#)

[Source Code](#)

Lets begin creating awesome grids using the abstract designer. We use v-row, v-col, v-sheets to explain concepts with #LiveCodeSwapping.

<https://github.com/Mashiane/BANanoVuetifyAD3/tree/main/Tutorials/Part23>

Created with the Personal Edition of HelpNDoc: [Free PDF documentation generator](#)

Part 24

[Youtube Link](#)

[Source Code](#)

We create a v-container, add some v-rows and v-cols. Inside the v-cols we add some v-text-fields, we use .SetData & .GetData to set and get states. We fire a click event on a v-btn and show a snackbar.

Created with the Personal Edition of HelpNDoc: [Free Web Help generator](#)

Part 25

[Youtube Link](#)

[Source Code](#)

This had to be a dash to address an important issue about component page based dialog boxes for alerts and confirm dialogs.

Created with the Personal Edition of HelpNDoc: [Create help files for the Qt Help Framework](#)

Part 26

[Youtube Link](#)

[Source Code](#)

Work review. Adding radio groups, switches, checkboxes, chips, text-area, file inputs, sliders and lists

Created with the Personal Edition of HelpNDoc: [Easily create CHM Help documents](#)

Part 27

[Youtube Link](#)

[Source Code](#)

Understanding the BANanoElement by creating a Vuetify Wireframe. We look IN DETAIL what the BANanoElement is and how you can use it to create a wireframe.

Created with the Personal Edition of HelpNDoc: [Easily create Help documents](#)

Part 28

[Youtube Link](#)

[Source Code](#)

We create a parallax, a rating and an image. This contains the latest source code and methodologies.

Created with the Personal Edition of HelpNDoc: [Easy to use tool to create HTML Help files and Help web sites](#)

Part 29

[Youtube Link](#)

[Source Code](#)

We create progress indicators e.g. linear and circular. We add click events to drawer items. We also add page transitions.
On file elements we add change event to trap file changes and display file properties.

Created with the Personal Edition of HelpNDoc: [Benefits of a Help Authoring Tool](#)

Part 30

[Youtube Link](#)

[Source Code](#)

We code: The grid and its structure using AddRows and AddColumns. We add element using AddButton, AddCheckBox, AddRadioGroup, AddImage at grid positions. We bind events to the buttons and checkbox and get state values.

Created with the Personal Edition of HelpNDoc: [Easy to use tool to create HTML Help files and Help web sites](#)

Part 31

[Youtube Link](#)

[Source Code](#)

We code a universal list that shows different kinds of content, avatars, icons, titles and sub-titles and it uses the cool v-if directive.

Created with the Personal Edition of HelpNDoc: [Full-featured EPub generator](#)

Part 32

[Youtube Link](#)

Source Code (refer to the latest source code)

We create a dialog with input components, e.g. textfield, select and autocomplete. We fire a trigger to show the dialog.

Created with the Personal Edition of HelpNDoc: [Produce online help for Qt applications](#)

Part 33

[Youtube Link](#)

[Source Code](#)

Creating steppers, preference dialog lists, tasks list, list group items

Created with the Personal Edition of HelpNDoc: [Free Kindle producer](#)

Part 34

[Source Code](#)

[Youtube Link](#)

We create a SQLite backend database and perform some crud functions, thanks to PHP.

Created with the Personal Edition of HelpNDoc: [Create HTML Help, DOC, PDF and print manuals from 1 single source](#)

Part 35

[Source Code](#)

[Youtube Link](#)

We create a MySQL backend database and perform some crud functions, thanks to PHP. Refer to Part 34 for more details.

Created with the Personal Edition of HelpNDoc: [Generate Kindle eBooks with ease](#)

Part 36

[Source Code](#)

[Youtube Link](#)

We create a MSSQL backend database and perform some crud functions, thanks to PHP. Refer to Part 34 for more details about explanation of the source code.

Created with the Personal Edition of HelpNDoc: [Easily create CHM Help documents](#)

Part 37

[Youtube Link](#)

[Source Code](#)

We create a BANanoSQL (IndexedDB using AlaSQL) backend database and perform some crud functions. Source code has been adopted from tutorial 34 with some minor adjustments. 1. The auto-increment is calculated via code and the .ExecuteWait code is updated.

Created with the Personal Edition of HelpNDoc: [News and information about help authoring tools and software](#)

Part 38

[Youtube Link](#)

[Source Code](#)

Data-table updates: Added add new button, clear sort, apply filter and clear filter. Also added v-chip-group for column visibility filter.

Created with the Personal Edition of HelpNDoc: [Free Qt Help documentation generator](#)

Part 39

[Youtube Link](#)

[Source Code](#)

We add switches to our lists. We can also make the lists insets.

Created with the Personal Edition of HelpNDoc: [Free Web Help generator](#)

Part 40

[Youtube Link](#)

[Source Code](#)

We create expansion panels.

Created with the Personal Edition of HelpNDoc: [Easily create Help documents](#)

Part 41

[Youtube Link](#)

[Source Code](#)

We explore firebase messaging with service workers.

Created with the Personal Edition of HelpNDoc: [Free CHM Help documentation generator](#)

Part 42

[Youtube Link](#)

[Source Code](#)

We get a firebase topic, add a topic for subscription, subscribe to the topic and then send a message to the topic

Created with the Personal Edition of HelpNDoc: [Produce Kindle eBooks easily](#)

Part 43

[Youtube Link](#)

[Source Code](#)

This 43rd tutorial deals with how we can get started with BANanoVuetifyAD3 to create web apps: We do things differently this time. We have been developing and testing with laragon, the apps also work well with IIS. So explore for the firsttime with Vuetify the BANanoServer (Java Jetty), web-server. We run our app on port 55056, it uses web-sockets, and use an IndexedDB backend. This DB we have used before.

Created with the Personal Edition of HelpNDoc: [Easily create EPub books](#)

Part 44

[Youtube Link](#)

[Source Code](#)

This 44th tutorial deals with how we can get started with BANanoVuetifyAD3 to create web apps: We create a CRUD functionality using SQLite DB as a backend with the java jetty web server.

Created with the Personal Edition of HelpNDoc: [Full-featured multi-format Help generator](#)

Part 45

[Youtube Link](#)

[Source Code](#)

This 45th tutorial deals with how we can get started with BANanoVuetifyAD3 to create web apps:
We create a CRUD functionality using MySQL DB as a backend with the java jetty web server.

Created with the Personal Edition of HelpNDoc: [Easy EBook and documentation generator](#)
