



UNIDAD 15. AMPLIANDO VUE 2. FIREBASE, QUASAR Y NUXT

Desarrollo web en entorno cliente
CFGS DAW

Sergio García Barea
sergio.garcia@ceedcv.es
2018/2019

Versión:181217.1259


Licencia




Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante


ÍNDICE DE CONTENIDO

1. Introducción.....	3
2. Framework Quasar.....	3
2.1 Empezando con Quasar.....	3
2.2 Formato SFC (Single File Component en Vue).....	8
2.3 Escribiendo código Vue con Quasar.....	8
3. AMPLIACIÓN: Vue + Firebase + Vuefire.....	11
4. AMPLIACIÓN: NodeJs + Express + Nuxt (Vue).....	12
4.1 Ejemplo de aplicación.....	12
4.2 Bases del ejemplo visto.....	14
5. Enlaces interesantes para saber más.....	14

UD15. AMPLIANDO VUE 2. FIREBASE, QUASAR Y NUXT

1. INTRODUCCIÓN

En esta unidad vamos a presentar mediante ejemplos prácticos otras alternativas para aprovechar la potencia de Vue.

 Importante: esta unidad trata las herramientas expuestas por encima, ya que su fin es el de difundir su existencia y permitir los primeros pasos con ellas y a efectos académicos solo se valoraran a ese nivel. No obstante, animo a todos a ampliar conocimientos de forma autodidacta.

Entre ellas vamos a ver:

- Ejemplo de aplicación que utiliza Vue y la base de datos de Google Firebase <https://firebase.google.com> uniéndolos mediante la biblioteca “Vuefire” <https://github.com/vuejs/vuefire>
- Uso del Framework de Vue llamado Quasar <https://quasar-framework.org/> para realizar aplicaciones cliente Vue y poder exportarlas directamente a SPA, PWA, Electron y Cordova.
- Ejemplo de aplicación servidor usando NodeJS + Express + Nuxt (Vue).
 - Nuxt <https://nuxtjs.org/> básicamente tiene un funcionamiento similar a Vue en cuanto al desarrollo de la aplicación, pero el cliente solo recibe los cambios en el renderizado, trasladando la carga computacional al servidor (y protegiendo el acceso al código fuente)
 - Express <http://expressjs.com/> incluye un conjunto de funciones y métodos para desarrollar aplicaciones Web en Nodejs.

2. FRAMEWORK QUASAR

2.1 Empezando con Quasar

Quasar Framework <https://quasar-framework.org/> es un framework para crear aplicaciones usando Vue y otras bibliotecas de Nodejs y desplegarla de inmediato en 4 tipos de aplicaciones:

- SPA: Single Page Application (Aplicación de una página)
- PWA: Progressive Web Application (Similar a la anterior, pero con disponibilidad offline).
- Cordova: Aplicaciones móviles, estudiada en unidades anteriores.
- Electron: Aplicaciones de escritorio, estudiadas en unidad anterior.

Esto nos facilita el desarrollo de aplicaciones que queramos desplegar en distintos tipos de lugares (Web, móvil, escritorio...).

Además Quasar incluye componentes propios (los que empizan por q- algo, tipo q-bt, q-ico etc. <https://quasar-framework.org/components/introduction-for-beginners.html#Using-Quasar-Components>

Para utilizar Quasar tenéis dos opciones:

- Utilizar la máquina Docker que os proporcionamos con las herramientas necesarias
 - <https://hub.docker.com/r/sergarb1/dwec-ceedcv-nodejs-electron-quasar/>
- Instalar las herramientas básicas siguiendo los siguientes pasos:
 - <https://quasar-framework.org/guide/index.html>
 - En resumen:
 - apt-get install nodejs npm
 - npm install -g vue-cli
 - npm install -g quasar-cli
 - Para tener lo necesario para desplegar aplicaciones Android, lo más práctico y rápido es instalar Android Studio (que auto configura el Android SDK).

Una vez instalado podemos crear un proyecto Quasar con:

```
quasar init <folder_name>
```

Con dicho comando, os creará un proyecto Quasar con código de ejemplo basado en esta plantilla <https://github.com/quasarframework/quasar-starter-kit>

Tras ello os pedirá varios datos (nombre del proyecto, autor, nombre cordova, si utilizas algunas herramientas extra (por defecto ninguna), etc.).

Una vez contestadas las preguntas, te creara la estructura necesaria.

Podrás probar tu aplicación en las distintas arquitecturas con comandos como los siguientes ejemplos:

- quasar dev -m spa
 - Lanzar Quasar en modo desarrollo en plataforma SPA.
- quasar dev -m electron
 - Lanzar Quasar en modo desarrollo y para plataforma Electron.
- quasar dev -m cordova -T [android|ios]
 - Lanza Quasar en modo desarrollo y para Android o IOS según elijas.
- quasar build -m pwa -t mat
 - Lanza Quasar en modo producción para una SPA con tema “mat”.

Más ejemplos en <https://quasar-framework.org/guide/quasar-cli.html>

Para cualquier duda relacionada os animo a utilizar los foros del módulo.
De manera gráfica, lanzando en el ejemplo inicial el siguiente comando

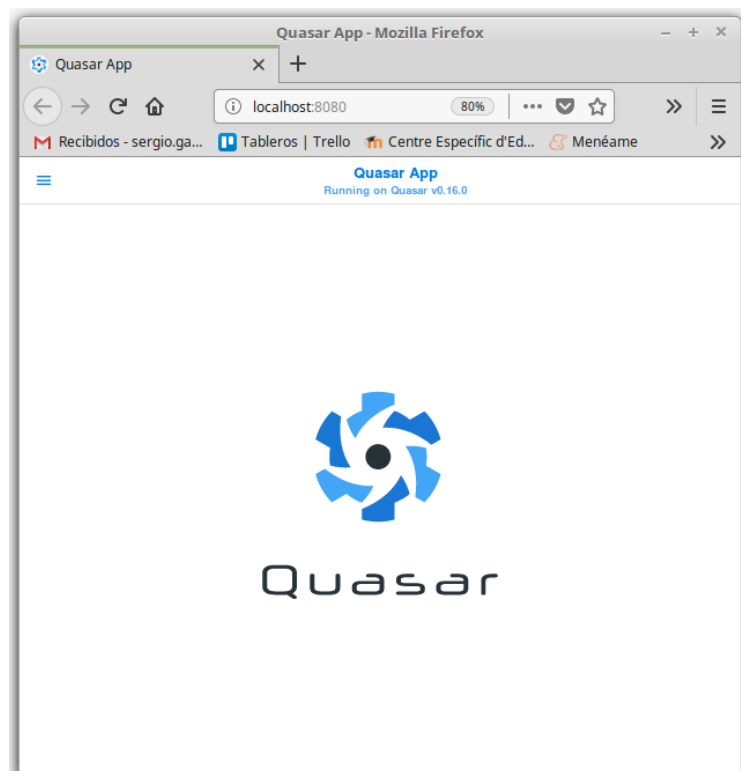
```
quasar dev -m spa -t ios
```

Donde creamos una SPA y el tema es tipo IOS (ojo, eso es solo visual, no tiene nada que ver con la plataforma para la cual se genera código).

Nos aparece la siguiente información en consola

```
sergie@SergiCEEDCV ~ $ cd prueba/  
sergie@SergiCEEDCV ~/prueba $ quasar dev -m spa -t ios  
app:dev Running: Mode [ SPA ] with [ IOS ] theme +0ms  
  
app:quasar-conf Reading quasar.conf.js +953ms  
app:dev Checking listening address availability (0.0.0.0:8080)... +3ms  
app:quasar-conf Generating Webpack config +8ms  
app:quasar-conf Extending Webpack config +32ms  
app:generator Generating Webpack entry point +4ms  
app:dev-server Booting up... +4ms  
  
Build completed in 7.001s  
  
DONE Compiled successfully in 7008ms  
  
I App [SPA with IOS theme] at http://localhost:8080/
```

Y vemos el resultado de la siguiente forma



De manera gráfica, lanzando en el ejemplo inicial el siguiente comando

```
quasar dev -m electron -t ios -T ubuntu
```

Donde es una aplicación Electron, con tema IOS y como objetivo (Target) es una plataforma Linux Ubuntu.

Nos aparece la siguiente información en consola

```
sergi@sergileebtv ~/prueba $ quasar dev -m electron -t ios -T ubuntu
app-dev Running: Mode [ ELECTRON ] with [ IOS ] theme +0ms

app-mode Quasar ELECTRON is missing. Installing it... +42ms
app-mode electron Installing Electron dependencies... +4ms
app:spawn [sync] Running "npm install --save-dev electron@2.0.0 electron-debug@1.5.0 electron-devtools-installer@2.2.4 devtron@1.4.0" +8ms

> electron@2.0.0 postinstall /home/sergi/prueba/node_modules/electron
> node install.js

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
+ electron-devtools-installer@2.2.4
+ electron@2.0.0
+ electron-debug@1.5.0
+ devtron@1.4.0
added 91 packages from 345 contributors and audited 12219 packages in 21.963s
found 0 vulnerabilities

app-mode-electron Creating Electron source folder... +24s
app-mode-electron Electron support was added +52ms
app-quasar-conf Reading quasar.conf.js +903ms
app-dev Checking listening address availability (0.0.0.0:8080)... +4ms
app-quasar-conf Generating Webpack config +5ms
app-quasar-conf Extending renderer process Webpack config +30ms
app-quasar-conf Generating Electron Webpack config +4ms
app-quasar-conf Extending main process Webpack config +21ms
app-generator Generating Webpack entry point +3ms
app-dev-server Booting up... +4ms

Build completed in 6.847s

[DONE] Compiled successfully in 6856ms

app-electron Building main Electron process... +7s
Build completed in 0.645s

app-electron Webpack built Electron main process +647ms

Hash: 6f51f9ede9b5cc4f4ff9
Version: webpack 4.9.1
Time: 646ms
Built at: 2018-07-02 11:33:16
    Asset      Size      Chunks             Chunk Names
electron-main.js 192 KiB  electron-main [emitted]  electron-main
Entrypoint electron-main = electron-main.js
```

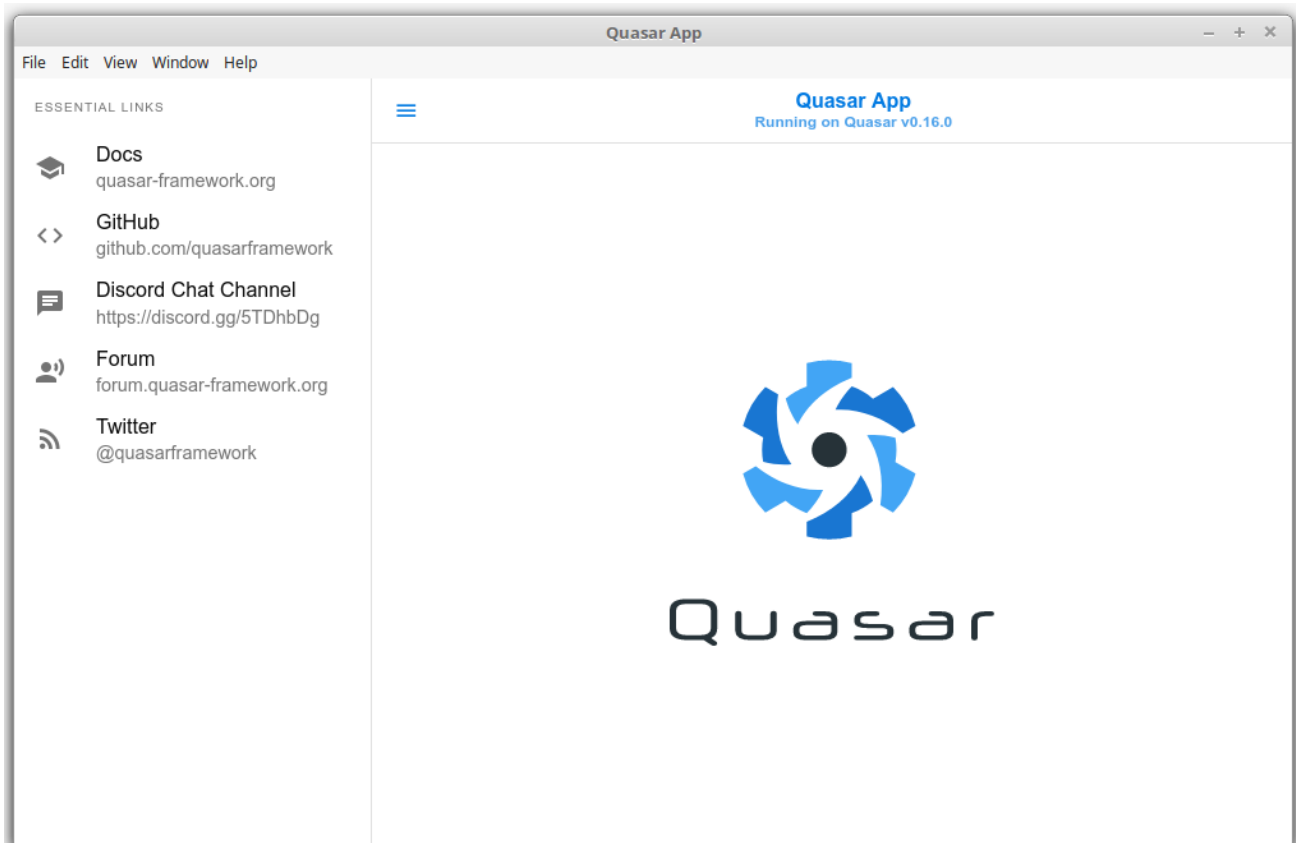
```
WARNING in ./node_modules/electron-debug/index.js
81:45-58 Critical dependency: the request of a dependency is an expression
 @ ./node_modules/electron-debug/index.js
 @ ./src-electron/main-process/electron-main.dev.js
 @ multi ./src-electron/main-process/electron-main.dev.js

WARNING in ./node_modules/electron-debug/index.js
84:85-106 Critical dependency: the request of a dependency is an expression
 @ ./node_modules/electron-debug/index.js
 @ ./src-electron/main-process/electron-main.dev.js
 @ multi ./src-electron/main-process/electron-main.dev.js

app-electron Booting up Electron process... +4ms
app:spawn Running "/home/sergi/prueba/node_modules/electron/dist/electron --inspect=5858 /home/sergi/prueba/.quasar/electron/electron-main.js" +1ms

Debugger listening on ws://127.0.0.1:5858/e96ac0fc-0818-4c30-944d-42c287974b61
For help see https://nodejs.org/en/docs/inspector
[5964:0702/113318.360351:ERROR:CONSOLE(7574)] "Extension server error: Object not found: <top>", source: chrome-devtools://devtools/bundled/inspector.js (7574)
```

Y vemos la aplicación así.



2.2 Formato SFC (Single File Component en Vue)

Antes de empezar, vamos a repasar la estructura general de los componentes “.vue”. Esto no es algo propio de Quasar, ya que es un estilo de desarrollo de componentes Vue para que tengan todo lo necesario en un fichero.

```
<template>
| <!-- you define your Vue template here -->
|
|</template>
|
|<script>
|
|// This is where your Javascript goes
|// to define your Vue component, which
|// can be a Layout, a Page or your own
|// component used throughout the app.
|export default {
|  //
|}
|</script>
|
|<style>
|
|/* This is where your CSS goes */
|
|</style>
```

Donde:

- “template” es una plantilla HTML 5 (nos permite poner código HTML pero que no se cargue para poder usarlo cuando queramos). https://www.w3schools.com/tags/tag_template.asp donde cargamos nuestra aplicación Vue.
- “script” es donde va el código Javascript que define el componente Vue (generalmente puede ser un layout, una página o tu propio componente).
- “style” donde se define el CSS a usar.

2.3 Escribiendo código Vue con Quasar

Aquí tenéis el detalle de la estructura de directorios que presenta Quasar

<https://quasar-framework.org/guide/app-directory-structure.html>

Existe la carpeta “src” (la más importante, con vuestro código).

También conforme vayamos generando código para distintas plataformas se crearán las carpetas con el código asociado “src-electron”, “src-cordova”, etc. por si queremos exportarlo como proyectos independientes, aunque a la hora de

trabajar con Quasar deberemos hacerlo siempre sobre “src”.

De manera sencilla, la forma más típica de escribir código sencillo dentro de la carpeta “src”:

- Modificar “router/routes.js” para incluir las rutas nuevas y asociarlas a componentes Vue y ficheros “.vue” que se requieran
- Modificar directorio “layouts” donde se incluirán los layout creados que iran cargando las distintas páginas Vue.
- Modificar directorio “pages” donde se incluirán los “.vue” adecuados.
- Otros directorios interesantes: assests para recursos (imagenes, sonidos), “components” para componentes Vue, “css” para ficheros css, etc.

La página inicial Vue (En el raíz de “src” llamada “App.vue”) tiene la siguiente estructura:

```
<template>
  <div id="q-app">
    <router-view />
  </div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>
</style>
```

Donde hace referencia que en el div con id=“q-app” (definido en el fichero “index.template.html” de la raíz de “src”).

En el div muestra lo que le indique el componente <router-view>

<https://router.vuejs.org/api/#router>

Donde:

- template es una plantilla HTML 5 (nos permite poner código HTML pero que no se cargue para poder usarlo cuando queramos).
https://www.w3schools.com/tags/tag_template.asp
- q-page es un componente de Quasar para indicar páginas
<https://quasar-framework.org/components/layout-page.html>

Con esto conseguimos el ejemplo de que al acceder a "/" nos cargue en el <router-view> el ".vue" de "pages/index.vue" y en el caso de ser otra página que no encuentre, cargue el ".vue" de "pages/404".

El funcionamiento del componente está definido en el fichero "router/routes.js"

```
export default [  
  {  
    path: '/',  
    component: () => import('layouts/default'),  
    children: [  
      { path: '', component: () => import('pages/index') }  
    ]  
  },  
  { // Always leave this as last one  
    path: '*',  
    component: () => import('pages/404')  
  }  
]
```

Un ejemplo del fichero ".vue" con nombre "index.vue"

```
<template>  
  <q-page class="flex flex-center">  
      
  </q-page>  
</template>  
  
<style>  
</style>  
  
<script>
```

```
export default {  
  name: 'PageIndex'  
}  
</script>
```

Donde:

- template es una plantilla HTML 5 (nos permite poner código HTML pero que no se cargue para poder usarlo cuando queramos).
https://www.w3schools.com/tags/tag_template.asp
- q-page es un componente de Quasar para indicar páginas
<https://quasar-framework.org/components/layout-page.html>

Con esto indicamos que cuando se cargue este componente Vue, se sobrescriba el q-page con el código Vue correspondiente.

🔊 Para más información sobre el desarrollo de aplicaciones con Quasar te recomendamos el enlace <https://quasar-framework.org/components/introduction-for-beginners.html>

3. AMPLIACIÓN: VUE + FIREBASE + VUEFIRE

Firebase es una plataforma de apoyo al desarrollo Web en la nube creada por Google <https://firebase.google.com/>.

Entre otras funciones facilita el alojamiento de base de datos no relacional (que es lo que usaremos en nuestra aplicación) y cuya característica más importante es que permite la **sincronización en tiempo real**.

Además Firebase ofrece otras funciones (login, estadísticas, etc.).

Más información en wikipedia <https://es.wikipedia.org/wiki/Firebase>

Aquí tenéis información de como añadir Firebase al código de vuestro proyecto Web <https://firebase.google.com/docs/web/setup?hl=es-419>

Para facilitar el uso de Firebase con Vue, se utiliza el plugin “Vuefire” que está disponible en <https://github.com/vuejs/vuefire>

Para incluirlo en la parte Web en concreto tenemos el “.js”

<https://github.com/vuejs/vuefire/blob/master/src/vuefire.js>

Para aprender los fundamentos de Firebase + Vue + Vuefire, hacemos referencia al ejemplo de la lista de tareas que almacena (y sincroniza) las tareas con Firebase.

Podéis verlo en marcha en <http://hispabyte.net/ListaTareasFirebase/>

El código para implementar dicho programa esta disponible en

<https://github.com/sergarb1/vuejs2/tree/master/lista-de-tareas-firebase>

Como método de aprendizaje de uso de Vue + Firebase + Vuefire se plantea la lectura de este ejemplo y sus abundantes comentarios.

Para implementar el ejemplo debéis seguir estos pasos:

1. Obtener una cuenta Firebase.
2. Descargar el código fuente y adaptar los datos a vuestra base de datos Firebase creada.
3. Lanzad la aplicación.

Para cualquier duda que surja, os invito a plantearla en el foro del módulo.

4. AMPLIACIÓN: NODEJS + EXPRESS + NUXT (VUE)

En este punto veremos un ejemplo de aplicación servidor usando NodeJS + Express + Nuxt (Vue).

- Nuxt <https://nuxtjs.org/> básicamente tiene un funcionamiento similar a Vue en cuanto al desarrollo de la aplicación, pero el cliente solo recibe los cambios en el renderizado, trasladando la carga computacional al servidor (y protegiendo el acceso al código fuente)
- Express <http://expressjs.com/> incluye un conjunto de funciones y métodos para desarrollar aplicaciones Web en Nodejs. Aquí lo utiliza como apoyo.

4.1 Ejemplo de aplicación

Para comprender los fundamentos básicos de Express y Nuxt, veremos el siguiente ejemplo: <https://github.com/ndarilek/express>, forkeado en mi cuenta

En primer lugar instalamos vue-clie

```
npm install -g vue-cli
```

Con ello ejecutamos

```
vue init nuxt/express miPrimerNuxt
```

Con lo que crea un proyecto a partir de la plantilla “nuxt/express” en el directorio “miPrimerNuxt”.

Para instalar las dependencias y luego ejecutarlo, dentro del directorio “miPrimerNuxt” podéis usar los comandos.

```
npm install
```

```
npm run dev #Si queremos lanzarlo y testearlo en desarrollo
```

```
npm run build #Si lo queremos construir solamente
```

```
npm start #Para lanzarlo en produccion
```

Para seguir el ejemplo lo lanzaremos con “npm run dev”. Si todo es correcto, se lanzará el proyecto para visualizarlo en <http://localhost:3000>

Aquí un ejemplo de lo que veremos.



USERS

[Alexandre](#)

[Joanne](#)

[Sébastien](#)

Visit our website for more documentation : nuxtjs.org

🔊 Observad el código HTML que recibe el cliente. No hay rastro de Vue ya que las operaciones de la programación reactiva son realizadas en el servidor y el cliente únicamente recibe que renderizar. Esto traslada la carga computacional al servidor, liberandola del cliente.

Para más información sobre Nuxt <https://github.com/i62navpm/Tutorial-Nuxt>

4.2 Bases del ejemplo visto

Dado la complejidad del ejemplo y que queremos solo explicar que es Nuxt de manera divulgativa, vemos de manera **MUY BÁSICA** que se realiza en el código de ejemplo.

En “api/index.js” se define la aplicación Vue.

En el fichero “api/routes/users.js” se definen las constantes de los usernames que aparecen en la aplicación y se establecen las acciones a realizar según las rutas solicitadas.

Por ejemplo:

```
/* GET user by ID. */
router.get('/users/:id', function (req, res, next) {
  const id = parseInt(req.params.id)
  if (id >= 0 && id < users.length) {
    res.json(users[id])
  } else {
    res.sendStatus(404)
  })
})
```

Si la ruta obtenida por get tiene el formato /users/0, coge “0” como valor del campo “id” en la orden “const id = parseInt(req.params.id)”.

La orden res (con sus funciones send, json, etc.) es para mostrar la respuesta HTML. En este caso con res.json, lo que hace es enviar una respuesta en formato JSON.

5. ENLACES INTERESANTES PARA SABER MÁS

Vuefire tutorial: Crud application

https://www.youtube.com/watch?v=831zOI02Q_0

Quasar tutorial

<https://www.youtube.com/watch?v=hGn8GqZF9yY>

Beginners guide Quasar

<https://quasar-framework.org/components/introduction-for-beginners.html>

Nuxt Introduction by project

<https://www.youtube.com/watch?v=nteDXuqBfn0&t=158s>

Tutorial Nuxt

<https://github.com/i62navpm/Tutorial-Nuxt>