**Scopul lucrarii** consta in cercetarea Frameworkurilor VueJs si Angular utilizate la crearea unei aplicatii web (sit) in care vor fi incadrate componentele acestor frameworkuri

1. **VueJs Framevork**

Creatorul Vue.js este Evan You, fost angajat al Google și Meteor Dev Group. El a început să dezvolte acest framework în 2013, iar în februarie 2014 a avut loc primul comunicat public. Vue este utilizat pe scară largă în rândul companiilor chinezești, de exemplu: Alibaba, Baidu, Xiaomi, Sina Weibo etc. Este inclus în nucleul Laravel și PageKit. Recent, sistemul de gestionare a depozitelor gratuite GitLab a trecut, de asemenea, la Vue.js.

Vue este un framework Front End progresiv pentru construirea de interfețe utilizator. Spre deosebire de alte cadre monolitice, Vue este proiectat de la început pentru a fi adoptat treptat. Biblioteca de bază se concentrează numai asupra stratului de vizualizare și este ușor de preluat și integrat cu alte biblioteci sau proiecte existente. Pe de altă parte, Vue este, de asemenea, perfect capabil să propulseze aplicații sofisticate cu o singură pagină atunci când este utilizat în combinație cu instrumente moderne și biblioteci de sprijin.Vue indeplineste un lucru enorm executind o legatura reactiva intre date si DOM.

Sa luam in consideratie cel mai simplu exemplu:

app.js  
var demo = new Vue({   
 el: '#demo',   
 data: {   
 message: 'Hello, World!'   
 }   
})

cream o nouă instanță prin new Vue

în el, definim ce element este urmărit

în data avem un obiect de stare

**app.html**

<div id="demo">   
 <p>{{message ? message : "no content"}}</p>   
<input v-model="message"> </div>

în html, firește, ar trebui să avem un element cu selectorul de care avem nevoie

Noi derivăm datele de la stat în sintaxa {{ }}. Aici avem expresia ternară obișnuită

input este asociat cu valoarea message a obiectului de stare folosind directiva v-model.

**1.1. Vue.js concepte**

Principalele concepte ale Vue sunt:

* Constructor
* Componentele
* Directive
* Router

Prima versiune a avut încă filtre, dar, din câte știu, în versiunea curentă sunt considerate depreciate.

**Constructor**

Lucrul cu Vue.js începe cu crearea unei noi instanțe new Vue. În el, avem un element pe care îl urmărește Vue. In template se selecteaza un element în care va fi afișat Vue. In data se stochează starea curentă a instanței și metoda computed ne furnizează proprietăți calculate.

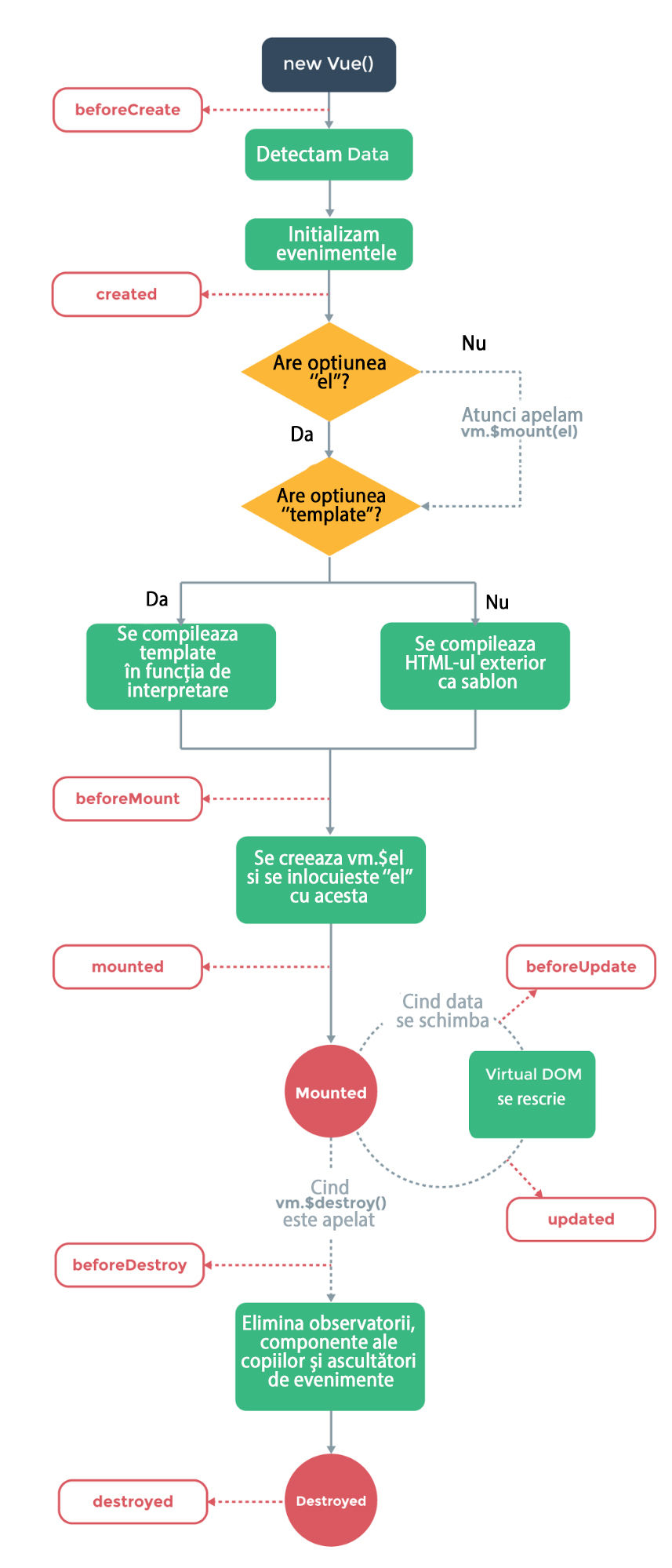
În methods se pot distinge următoarele metode și metode personalizate ale ciclului de viață Vue:

methods: {  
 beforeCreate: function(){},  
 created: function(){},  
 beforeMount: function(){},  
 mounted: function(){},  
 beforeUpdate: function(){},  
 updated: function(){},  
 beforeDestroy: function(){},  
 destroyed: function(){},  
 }

* beforeCreate - urmărește datele și inițiază evenimente
* created - caută el sau șablon. Dacă este așa, se face în ele; dacă nu, caută metoda render.
* beforeMount - creează vm.$el și le înlocuiește cu el
* mounted - elementul este redat

Când se schimbă starea:

* beforeUpdate - redă VDOM din nou și se compară cu DOM real, aplică modificările
* updated - modificări redate
* beforeDestroy - dezmembrarea completă a observatorilor, a componentelor interne și a ascultătorilor de evenimente
* destroyed – se apeleaza când operația se oprește



**Directive**

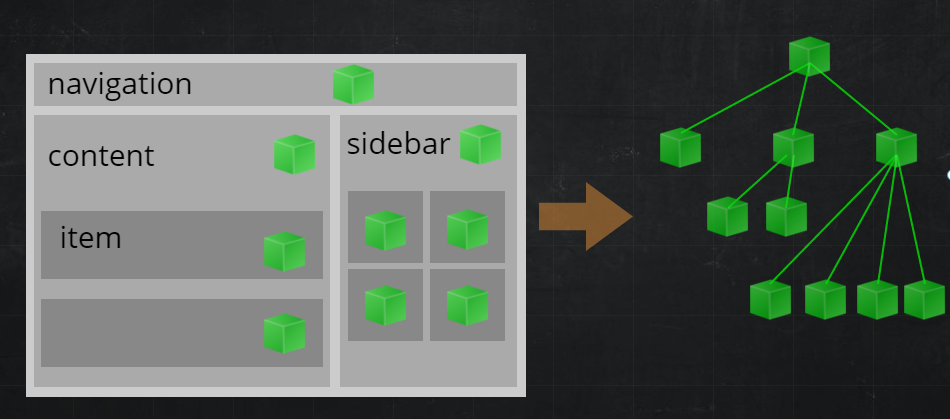
Directivele – atribute speciale pentru adăugarea elementelor html functionalitati noi.   
Iata exemple de directive care sunt similare cu cele din Angular:

* V-bind — Dinamic face legatura cu unul sau mai multe atribute
* v-if — conditie de modificare a elementului
* V-else — este blocul “else” pentru “if”
* V-for — trece masivul de obiecte ciclic
* V-model — leaga starea cu elementul input
* V-on — leaga ascultatorul de evenimente cu elementul
* V-once — deneseaza elementul doar la inceput si nu il mai urmareste
* V-pre — nu compileaza elementul si copii lui
* V-show — schimba afisarea elementului, schimbind proprietatea CSS display
* V-text — reinnoieste elementul textContent

Toate directivele Vue au prefixul "v-". In directivă este transmisă valoarea unei anumite valori de stare, iar argumentii pot fi atribute sau evenimente html.

**Componente**

**C**omponentele ajută la extinderea elementelor de bază html și la implementarea codului reutilizabil. În esență, componentele sunt părți reutilizabile ale User Interface. La etapa de proiectare, am împărțit aplicația noastră în părți independente și am obținut structura copac a componentelor.



Comunicarea între componentele vue se efectuează în conformitate cu principiul "Props in, Events out". Adică, de la elementul părinte la copil, informațiile sunt transmise prin props și înapoi - evenimentele sunt declanșate.

De asemenea, în Vue.js există așa-numitele componente cu un singur fișier. Creați un fișier cu extensia .vue și scrieți stiluri, modele și logică acolo. Și puteți scrie pe orice preprocesor convenabil (SASS, Stylus, PostCSS, Jade, ...) și limba compilată în JS (CoffeeScript, TypeScript).

**Tranzitii**

Vue oferă diferite modalități de a aplica efecte de animație atunci când elemente sunt desenate, actualizate sau șterse din DOM. Acestea includ instrumente pentru:

* aplicarea automată a claselor pentru tranziții și animații CSS
* integrarea bibliotecilor straine pentru animațiile CSS, cum ar fi Animate.css
* utilizarea JavaScript pentru a manipula DOM
* integrarea bibliotecilor JavaScript de la terți pentru animații, cum ar fi Velocity.js

Asta se întâmplă atunci când elementul este înfășurat într-o componentă de tranziție: Acesta va mișca în mod automat tranzițiile sau animațiile CSS aplicate. În caz contrar, CSS va fi adăugat / eliminat la cronometrele corespunzătoare.  
Există șase clase aplicate pentru tranzițiile de intrare / ieșire.

* v-enter: Starea de pornire pentru introducere. Adăugat înainte de introducerea elementului, a fost eliminat un cadru după introducerea elementului.
* v-enter-active: Stare activă pentru intrare. Aplicată pe toată faza de intrare. Adăugat înainte ca elementul să fie inserat, eliminat atunci când tranziția / animația se termină. Această clasă poate fi utilizată pentru a defini curba de durată, întârziere și relaxare pentru introducerea tranziției.
* v-enter-to: Disponibil numai în versiunile 2.1.8+. Stare de sfârșit pentru introducere. A fost adăugat un cadru după introducerea elementului (în același timp, v-enter este eliminat), eliminat atunci când tranziția / animația se termină.
* v-leave: Starea de plecare. Adăugat imediat când este declanșată o tranziție de plecare, eliminată după un cadru.
* v-leave-active: stare activă pentru concediu. Aplicată pe toată faza de plecare. Adăugat imediat când declanșarea tranziției este declanșată, se elimină atunci când tranziția / animația se termină. Această clasă poate fi utilizată pentru a defini curba de durată, întârziere și relaxare pentru tranziția de plecare.
* v-leave-to: Disponibil numai în versiunile 2.1.8+. Starea de încheiere pentru concediu. Adăugat un cadru după declanșarea unei tranziții de plecare (în același timp, eliminarea v-leave), eliminată atunci când tranziția / animația se termină.

**Routing**

În Vue.js, un pachet separat vue-router este responsabil pentru rutare. Crearea unei aplicații cu o singură pagină cu Router Vue + Vue este simplă. Cu Vue.js, deja compunem aplicația noastră cu componente. Acesta acceptă rute imbricate pentru componentele imbricate, oferă un API simplificat pentru cârligele de navigare, comportamentul de defilare gestionat și comenzile de navigare avansate. Atunci când adăugăm Vue Router la mix, tot ce trebuie să facem este să cartografiem componentele noastre pe rute și să lăsăm Vue Router să știe unde să le facă.