ToDo List

VueJS FullStack Project (PART 1)

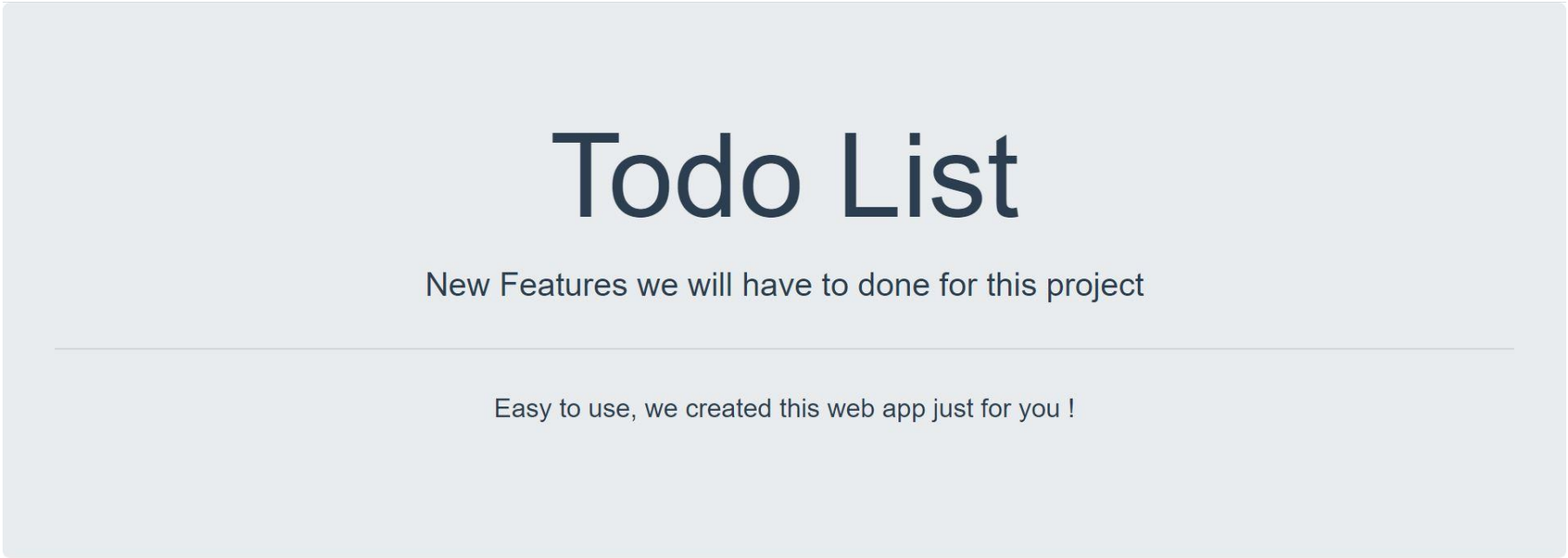
Aim of this project

- Create a little VueJS Project
- Learn how to use Components, Props, Smart & Dump components, $emit, Bootstrap-Vue
- This project will only contain a Front-End part for this first iteration
- You will be alone for this project and for the beginning you will have to only ask to Google or a trainer.
- One of the goal of this project is also to teach you to be autonome when you do not know a technology
- You will need a Git repository of course…
- This is new for everyone, so don’t think to much, try and retry and retry until you succeed ! Don’t miss the train, do the best you can and you will reach the goal !!

First Step : Vue CLI

- 1 : To simply start this project, we want you to generate a new project using Vue CLI.
- 2 : Let’s add Bootstrap Vue
- 3 : Make the view of the First View (see the next Diapo) (Jumbotron)

First Step View



Second Step : Separate

- We now want you to have 2 differents components : App (the principal one) and MyJumbotron (in which you will put the content of your page)

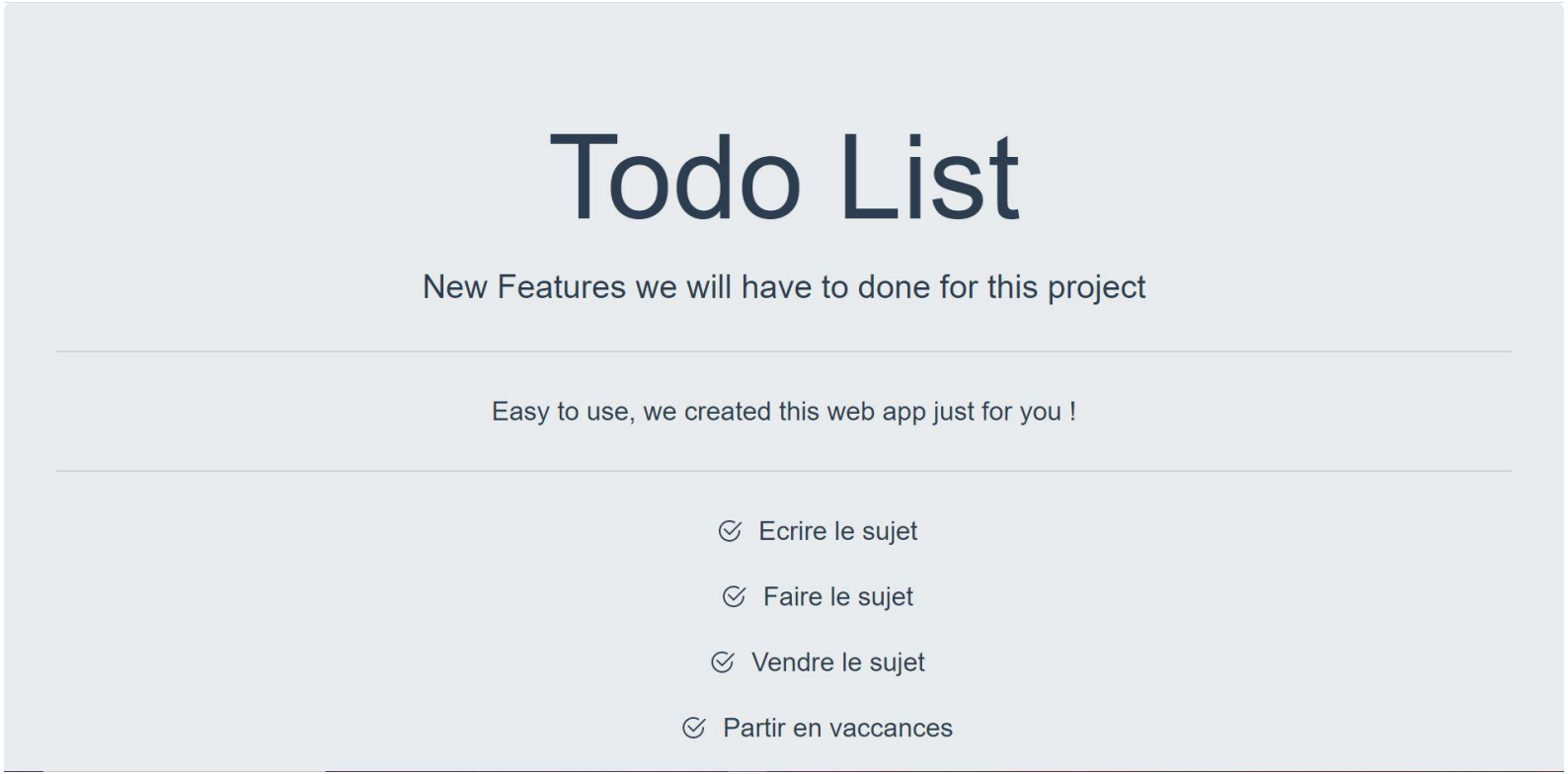
Third Step : List

- Now that you successfully call a component from another one, you will have to create a data called list in the App component.
- Then you will pass it to the MyJumbotron component as a Props.
-
```
      list: [
```
-
```
        {id: 0, name: "Ecrire le sujet", todo: true},
```
-
```
        {id: 1, name: "Faire le sujet", todo: true},
```
-
```
        {id: 2, name: "Vendre le sujet", todo: true},
```
-
```
        {id: 3, name: "Partir en vaccances", todo: true},
```
-
```
      ]
```
-

Fourth Step : Display List

- The MyJumbotron component will call a ListTodo component and give to it the List he received (as Props as well).
- The ListTodo component will iterate through the Array he received and will call SingleTodo component for each item of the List to looks like the picture below.

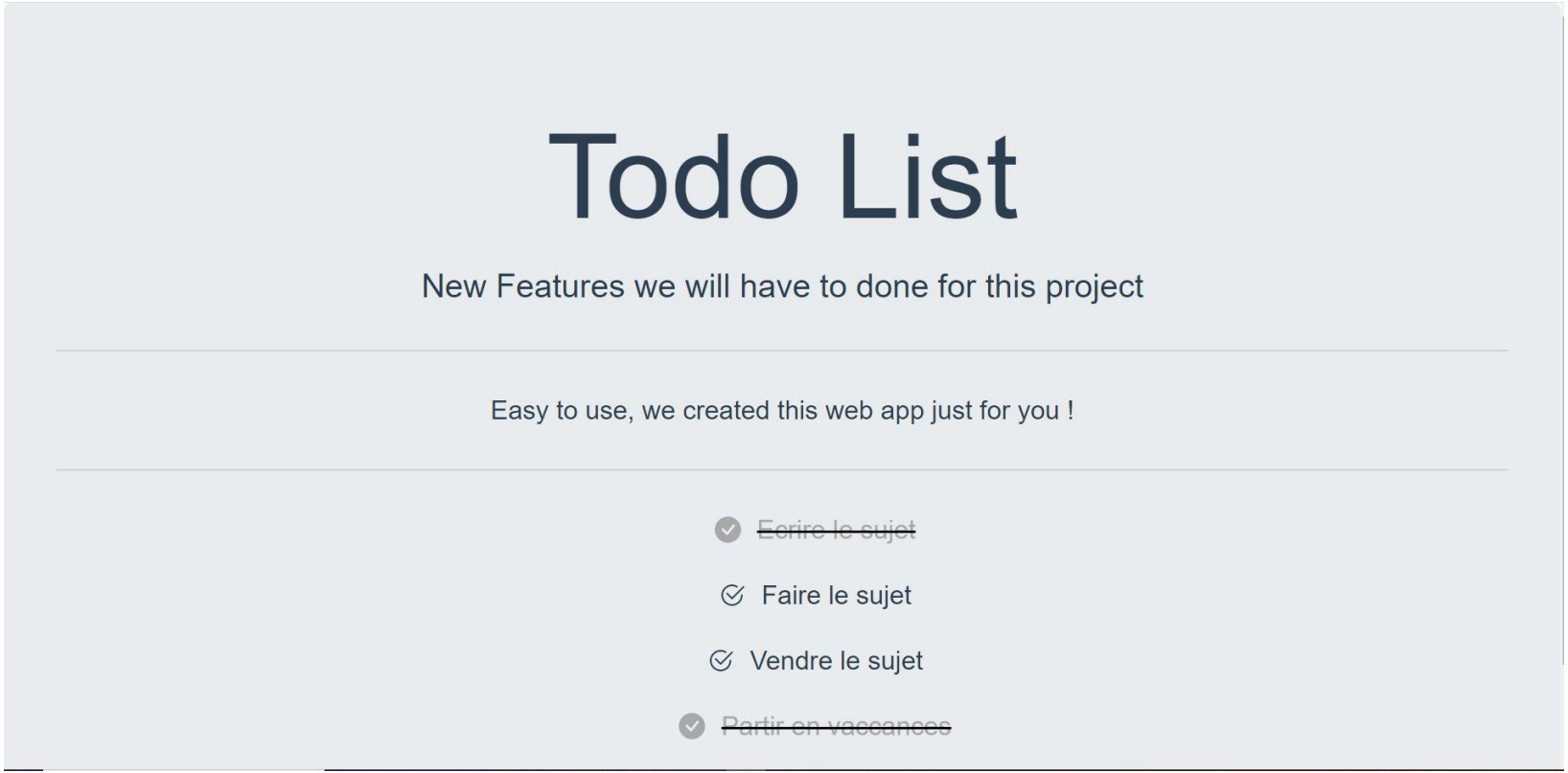
Fourth Step View



Fifth Step : Click

- Now that we successfully displayed our list of Todo, we want to be able to check or uncheck the different tasks in order to see what we still have to do.
- To do that you will add a click listener who will $emit called toggle from the SingleTodo component to the App component an event with the ID of the todo as param.
- The App component will toggle the field “todo” of this specific task from the ID.
- Then you will make a conditional rendering on the “todo” field in the SingleTodo component and change the icon and the class depending if the task is done or not.

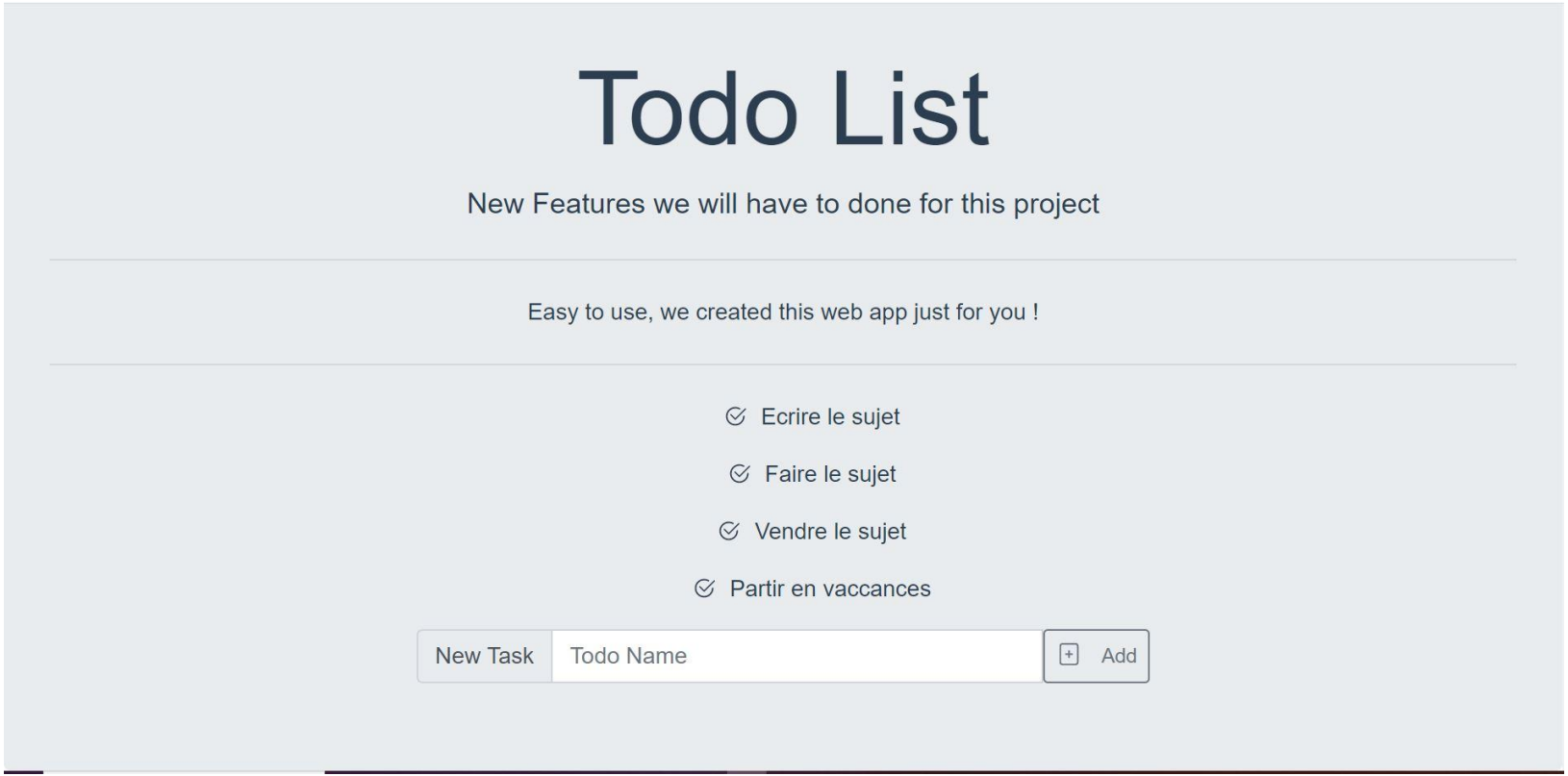
Fifth Step View



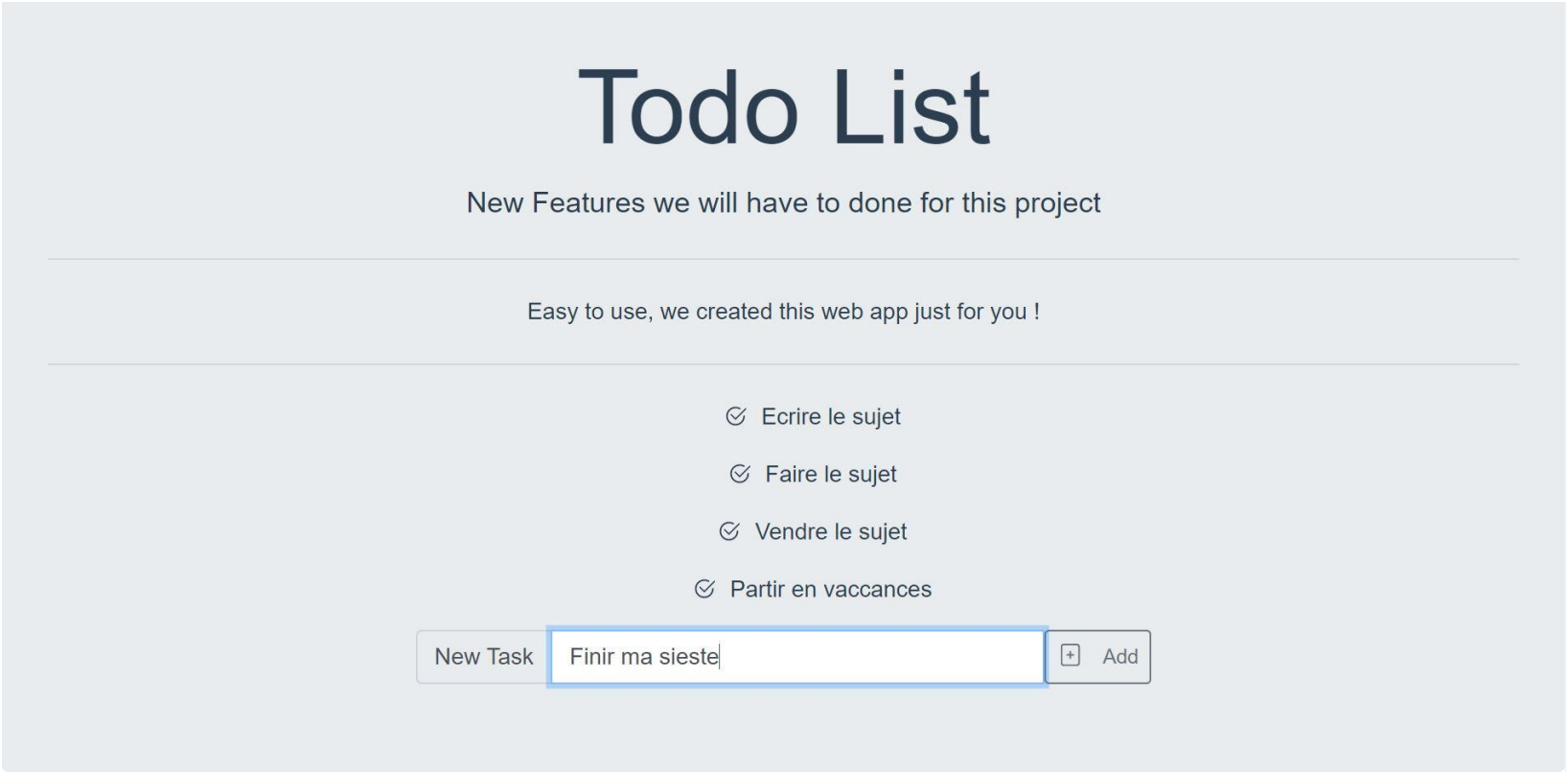
Sixth Step : AddForm

- We now want to be able to add some tasks to make it more dynamic.
- In order to do that we will call a new component AddForm from MyJumbotron component.
- The AddForm will have a single input of type text (obviously) and a single button in which we will call a method when @click on it.
- This component will $emit an event to the App component with the name of the new task the user wrote.
- See the three pictures below for have the good behaviour.

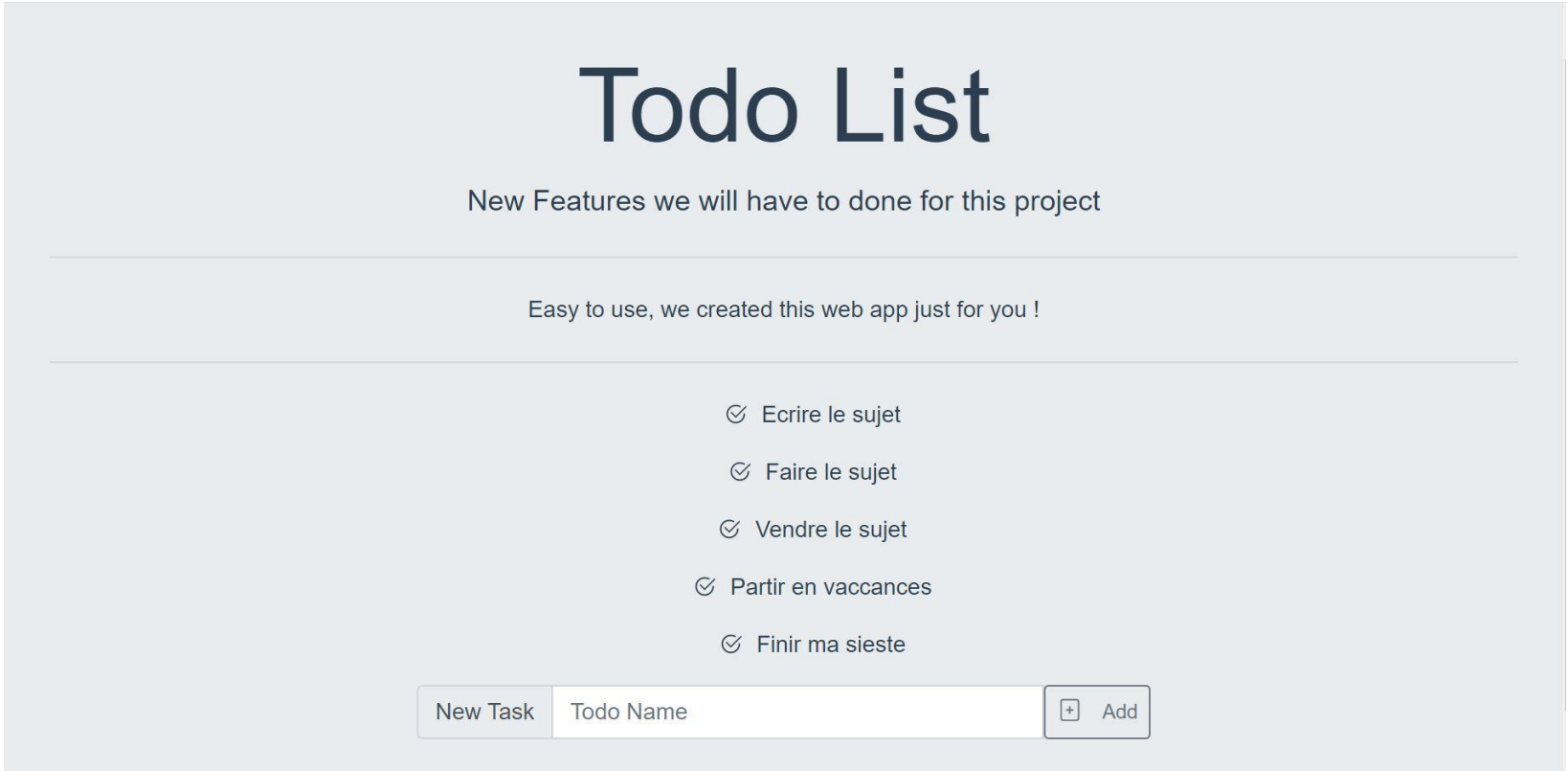
Sixth Step Views



Sixth Step Views



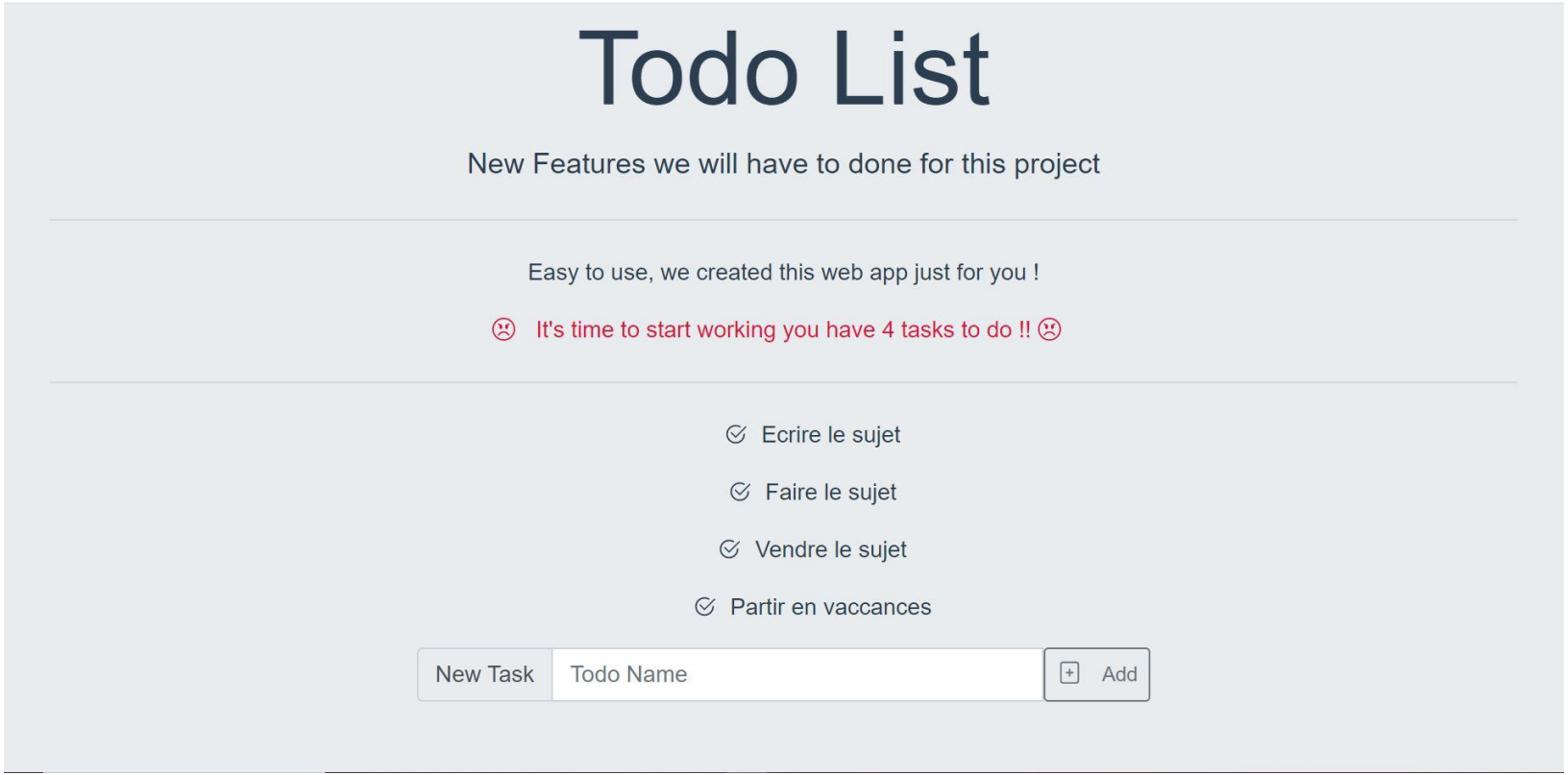
Sixth Step Views



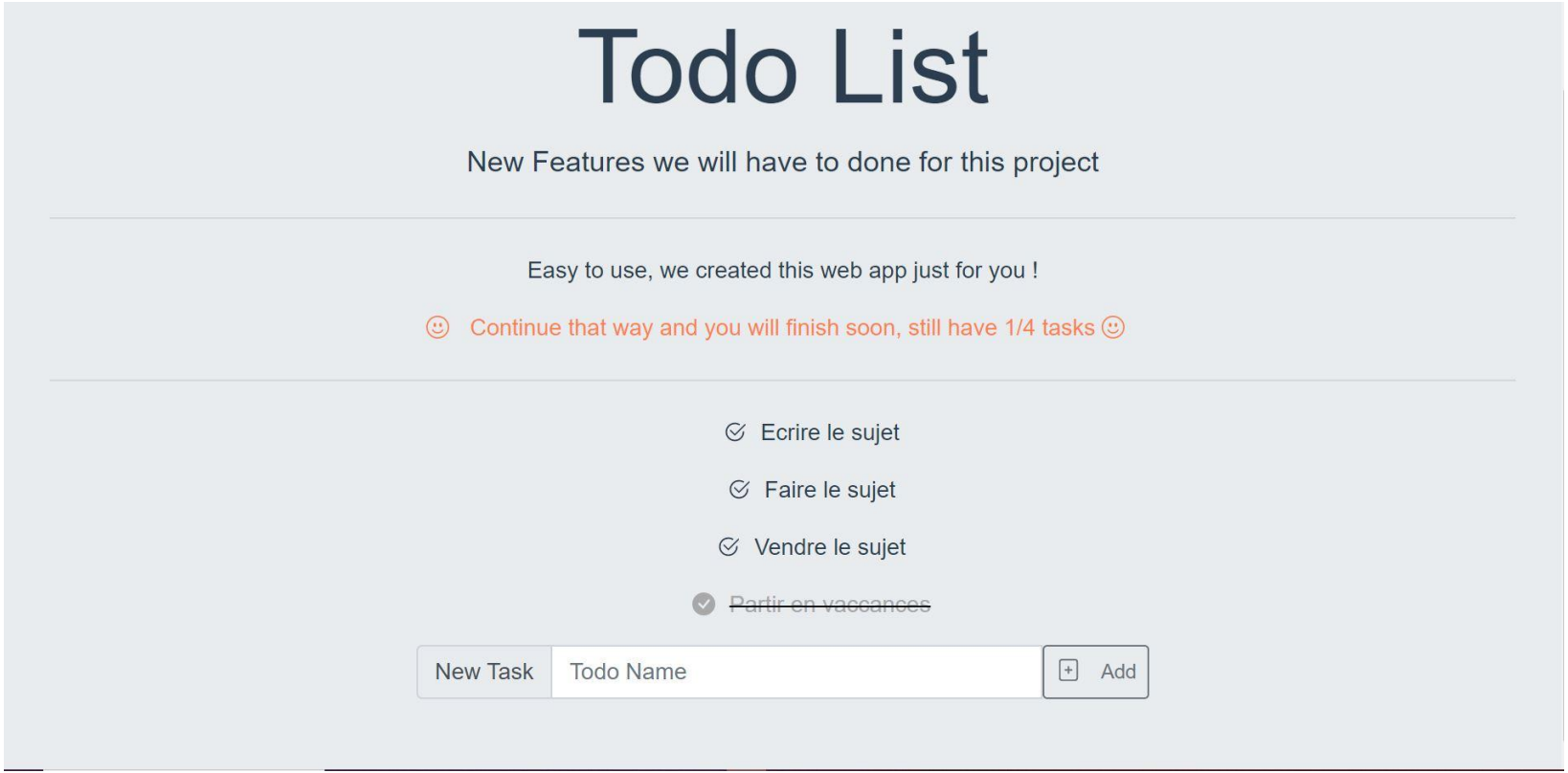
Seventh Step : Sentence

- We'll now create a new component Sentence who will display a different sentence depending of the number of tasks done.
- You will also adapt the style to look alike the screens below.
- If none tasks is done it will display : `It's time to start working you have x tasks to do !!`
- If less than the half are done : `Continue that way and you will finish soon, still have x/y tasks`
- If more than the half are done : `Good !! you made half or more x/y tasks`
- If all the tasks are done : `Congratulation !! You finished all the x tasks`

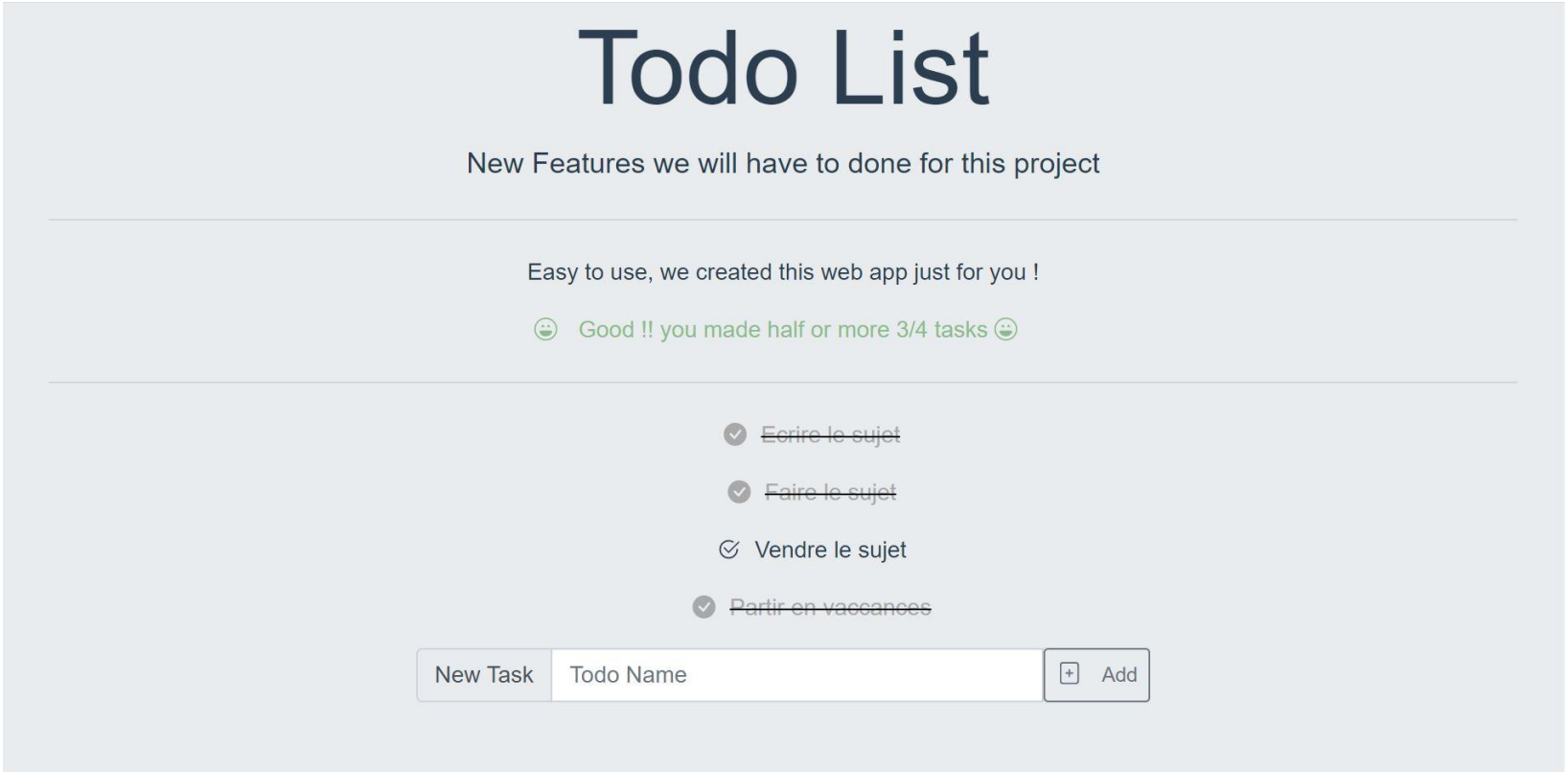
Seventh Step Views



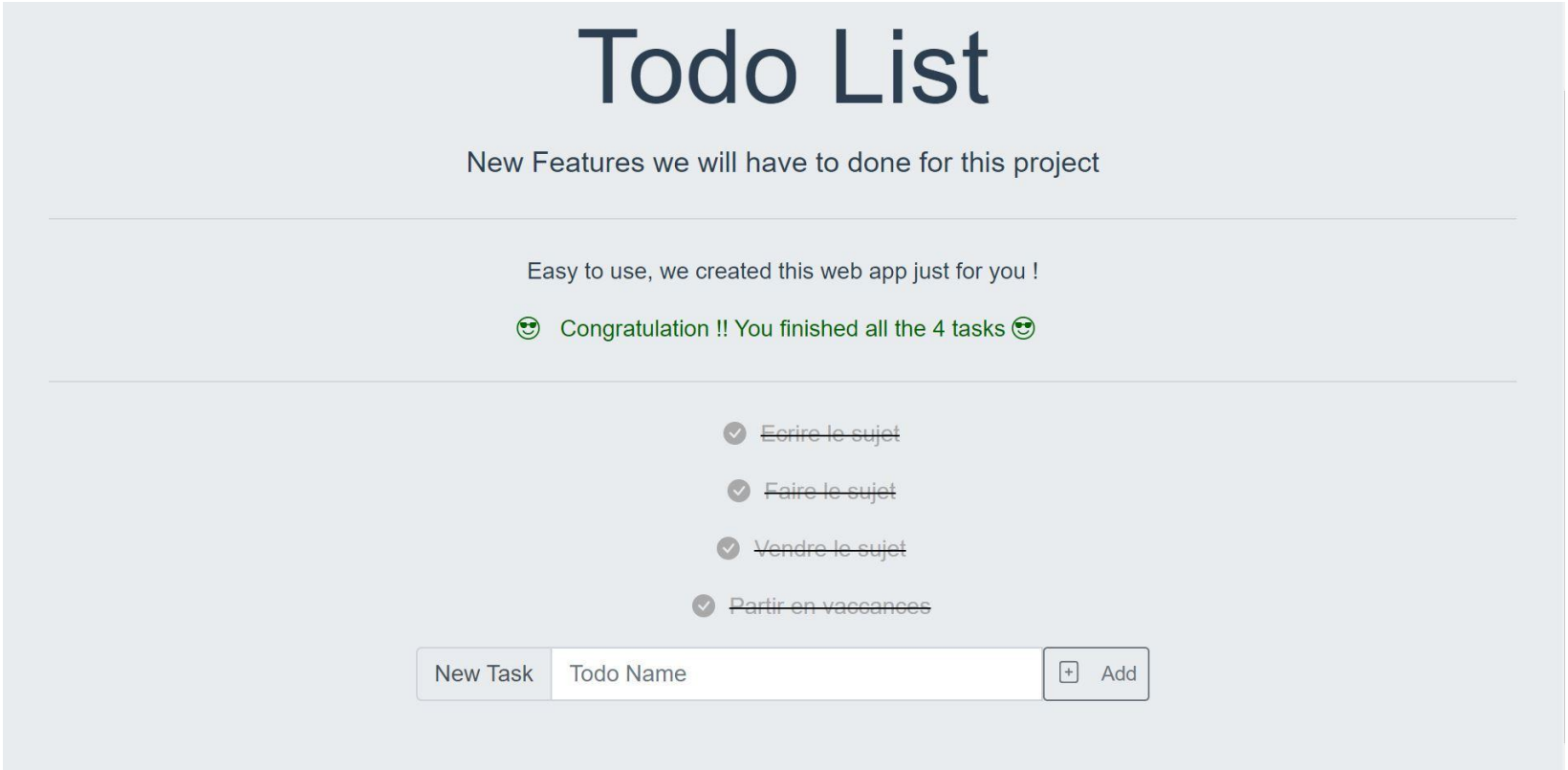
Seventh Step Views



Seventh Step Views



Seventh Step Views



TodoList : V2

FullStack

First Step : API

- First of all, we want to congrats you for your bravery discovering a new language and don’t complain !
- Secondly, we want you to create a very basic Express API for you ToDo.
  - POST /todo {name: String, id: Number, createdAt: String, todo: Boolean}
  - GET /todo
  - GET /todo/:id
  - PUT /todo/:id {todo: Booolean}
- The DB will be on Mongoose and you will have to define your model.
  - Mongoose isn’t complicated but allow you to “control” a little bit more the data before putting them in your DB

Second Step : Vue Router

- You will create a MyHeader component with a menu “New” and “List” with router-link.
- As explained during the watch on Vue Router, we want you to separate your application in four differents routes “/list” and “/new”, “done” and “todo”.
  - /new ⇒ It will display the form who allow the user to create a new Todo.
  - /list ⇒ It will display all the Todo, no matter they are completed or not.
  - /done ⇒ It will display the completed Todo
  - /todo ⇒ It will display the uncompleted Todo
  - ```
    const routes = [
    ```
  - ```
      { path: '/new', component: AddForm },
    ```
  - ```
      { path: '/list', component: ListTodo, props: {whatToDisplay: "all"}},
    ```
  - ```
      { path: '/done', component: ListTodo, props: {whatToDisplay: "done"}},
    ```
  - ```
      { path: '/todo', component: ListTodo, props: {whatToDisplay: "todo"}}
    ```
  - ```
    ]
    ```

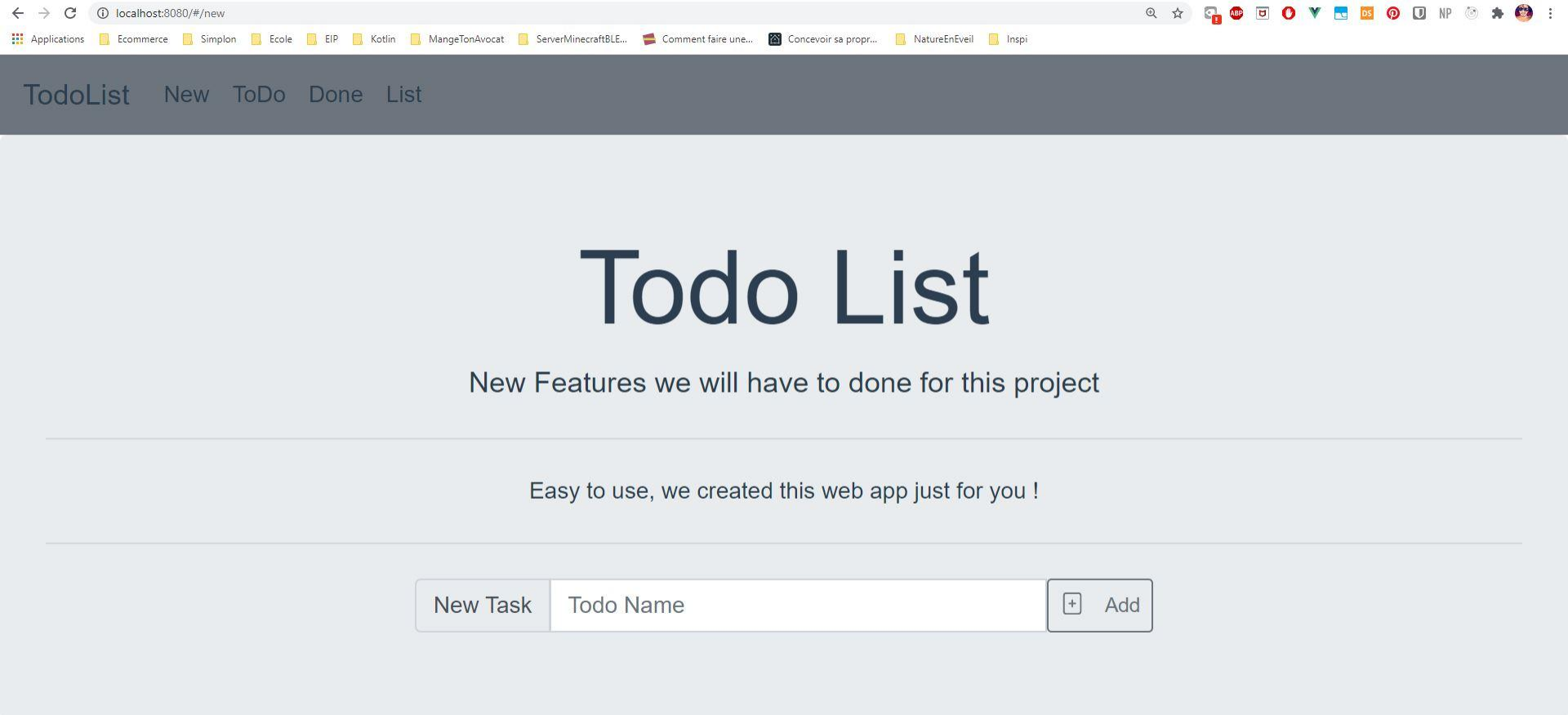
Second Step v2

- Look at “mounted” and “updated” component’s lifecycle
- Use the props whatToDisplay to filter the result of the Axios request !
- Have fun

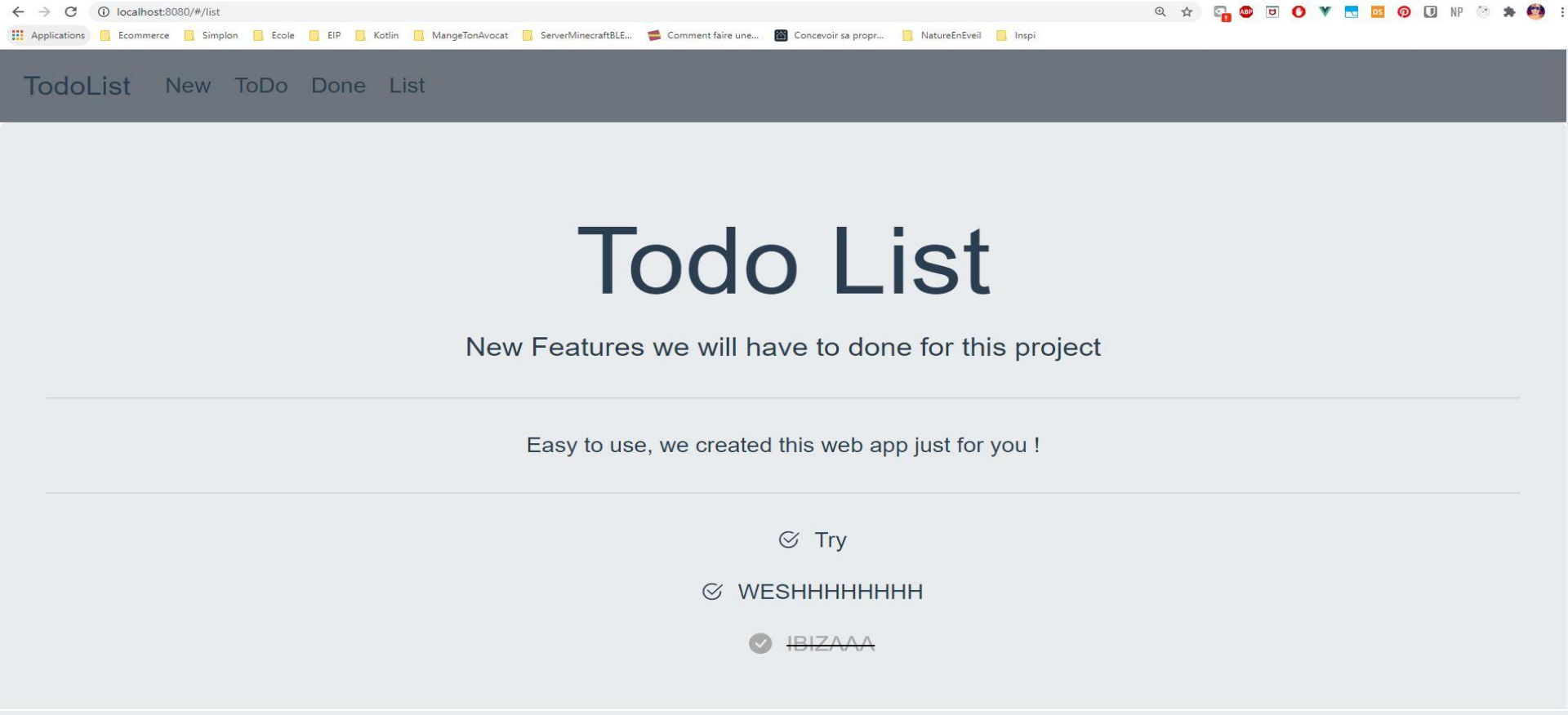
Third Step : Axios

- Now that we created an API, and that we separated our list and form, it’s time to interact with our API.
- As you guess :
  - In the ListTodo component you will now have to consume your API with Axios.
  - In the AddForm component you will have to POST a new Todo with Axios.
  - In the SingleTodo component you will have to PUT the new value of “todo” field for the todo you clicked on, with Axios.
    - After changing the Todo itself, SingleTodo component will have to $emit an event for ListTodo to GET again the actual states of Todos

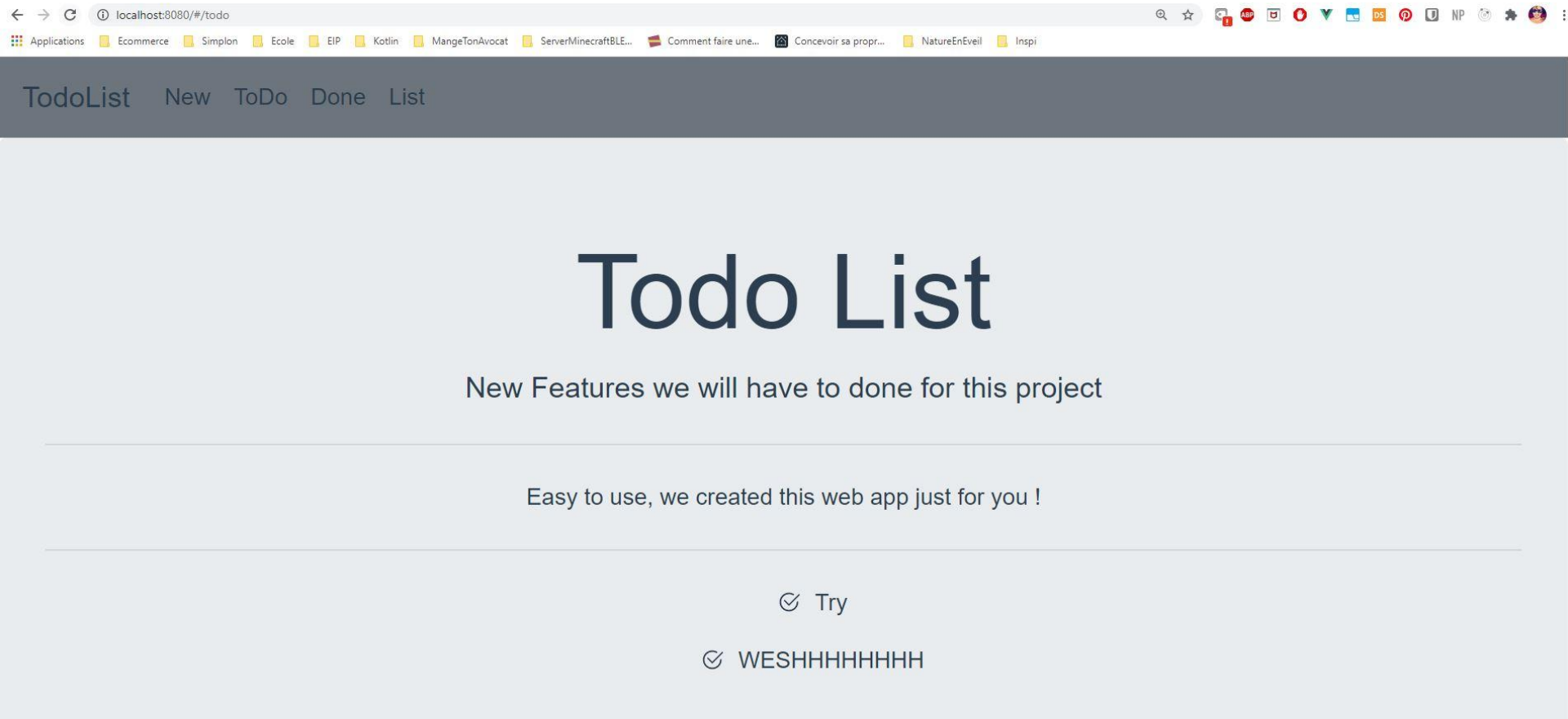
V2 Views


localhost:8080/#/new
Applications
Ecommerce
Simplon
Ecole
EIP
Kotlin
MangeTonAvocat
ServerMinecraftBLE...
Comment faire une...
Concevoir sa propr...
NatureEnEveil
Inspi
TodoList
New
ToDo
Done
List
Todo List
New Features we will have to done for this project
Easy to use, we created this web app just for you !
New Task
Todo Name
Add

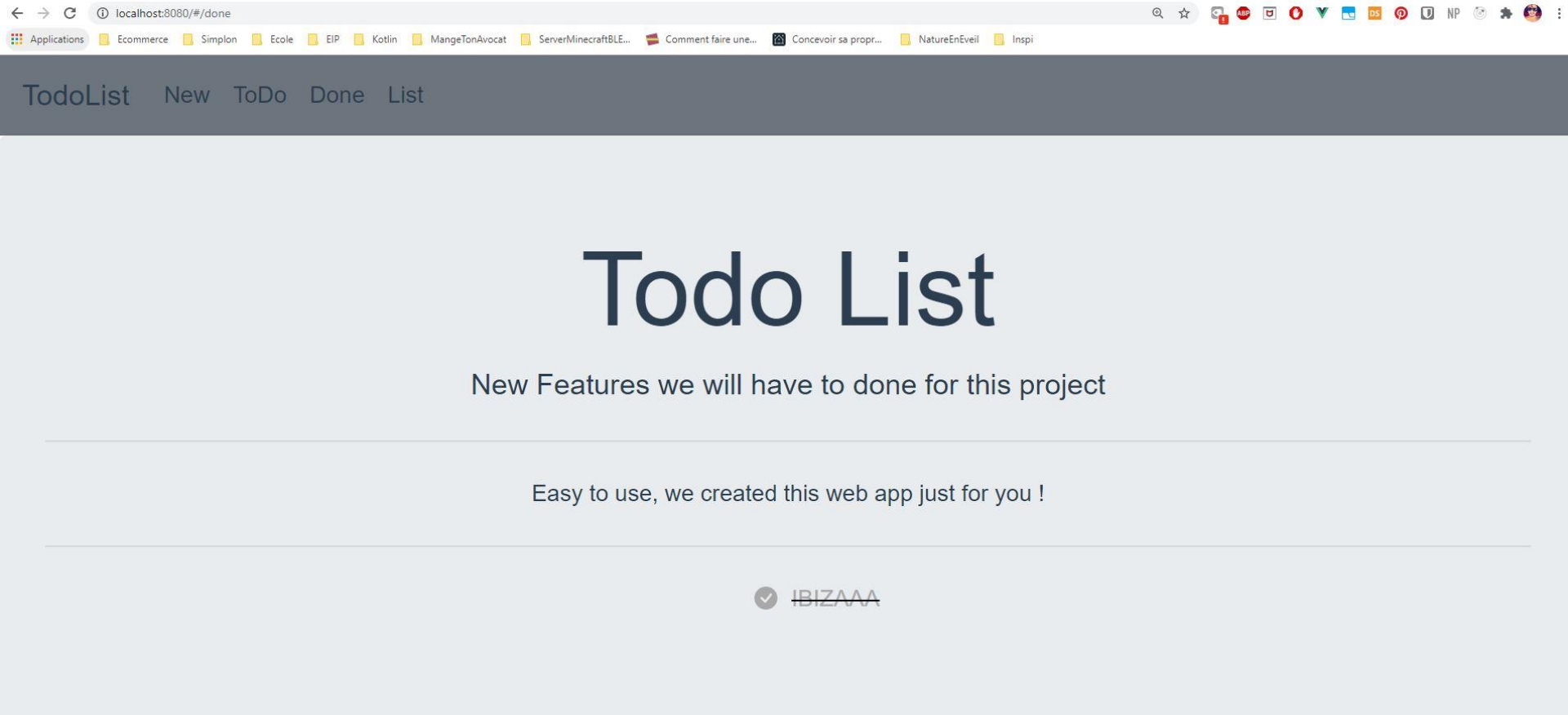
V2 Views


localhost:8080/#/list
Applications
Ecommerce
Simplon
Ecole
EIP
Kotlin
MangeTonAvocat
ServerMinecraftBLE...
Comment faire une...
Concevoir sa propr...
NatureEnEveil
Inspi
TodoList
New
ToDo
Done
List
Todo List
New Features we will have to done for this project
Easy to use, we created this web app just for you !
Try
WESHHHHHHHH
IBIZAAA

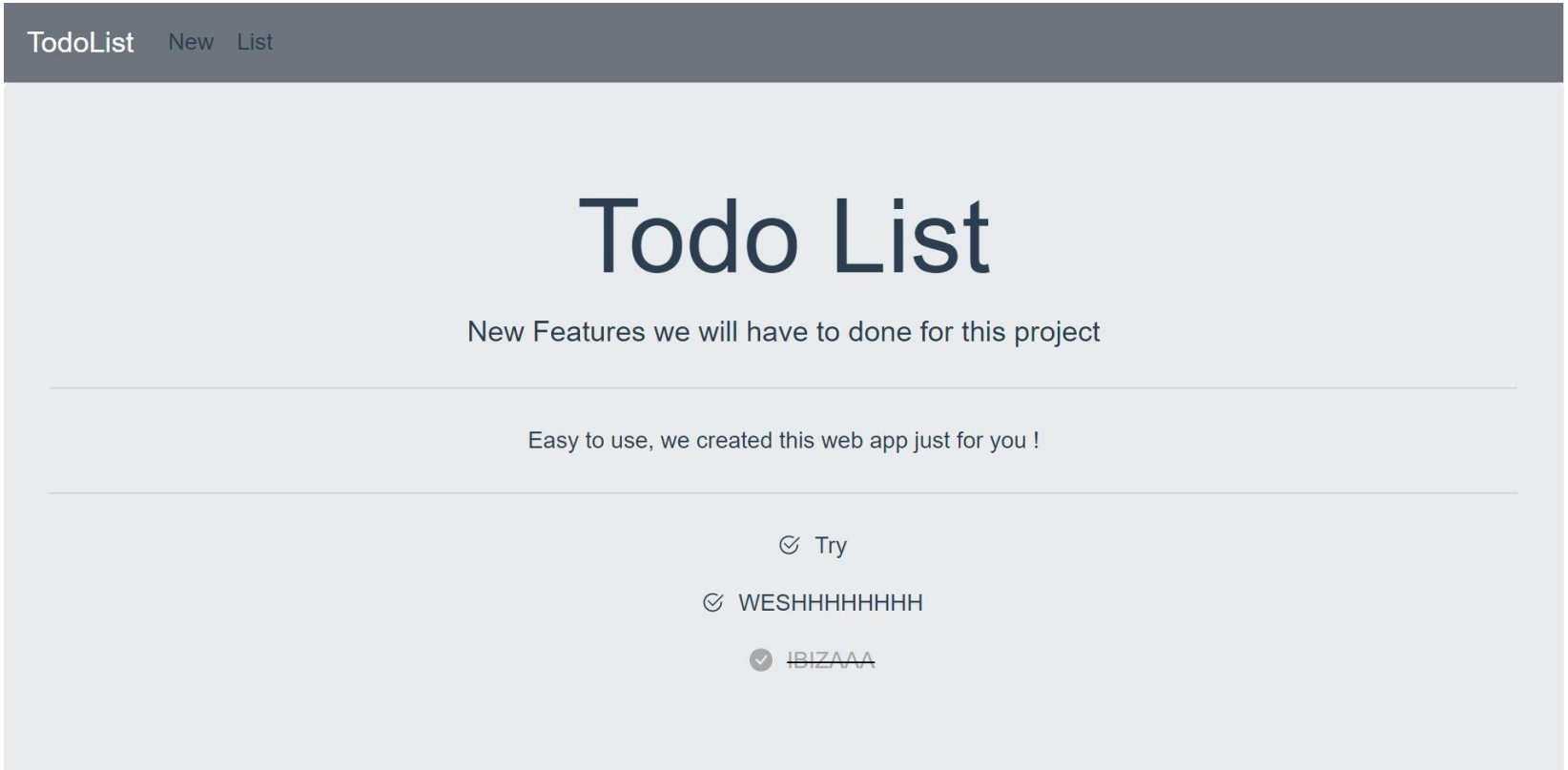
V2 Views


localhost:8080/#/todo
Applications
Ecommerce
Simplon
Ecole
EIP
Kotlin
MangeTonAvocat
ServerMinecraftBLE...
Comment faire une...
Concevoir sa propr...
NatureEnEveil
Inspi
TodoList
New
ToDo
Done
List
Todo List
New Features we will have to done for this project
Easy to use, we created this web app just for you !
Try
WESHHHHHHHH

V2 Views


localhost:8080/#/done
Applications
Ecommerce
Simplon
Ecole
EIP
Kotlin
MangeTonAvocat
ServerMinecraftBLE...
Comment faire une...
Concevoir sa propr...
NatureEnEveil
Inspi
TodoList
New
ToDo
Done
List
Todo List
New Features we will have to done for this project
Easy to use, we created this web app just for you !
IBIZAAA

V2 Views


TodoList
New
List
Todo List
New Features we will have to done for this project
Easy to use, we created this web app just for you !
Try
WESHHHHHHHH
IBIZAAA

TodoList : V3

Vuex

First Step : Store

- Congratulation again for finishing or almost finishing the V2 of this project.
- As you saw, working only with props and $emit is a bit annoying and needs to write too much code in every components.
- That is why we now want a store with Vuex.
    - For now, we only want you to store your ToDo in your vuex and working with your store to add, get and edit your ToDo.
    - To fulfill your store you will call the API once in your App.vue component then store the result.
    - Every time you Add or Edit a ToDo, you will first POST or PUT, then you will Update your Store.
    - To GET your ToDos, you will reach your store in ListToDo component
    - To PUT your ToDo, you will PUT to the API (for the DB to be aware) and you will change the Store.
    - To POST your Todo, you will POST to the API (“”””””) and you will change the Store.

When this will be done, we could say you saw every notion of Vue.js and no matter how it was hard or complicated, it will never be as complicated as it was ^^ !

First Step : Store

- To change the state of Todos, you will create some “mutations” as well as some “actions” (on the opposite of the Vuex Watch we assist today).
- The state will be an Array of Object {name: String, id: Number, createdAt: String, todo: boolean}.

Bonus

- Delete Todo when clicking on an icon
- Change the style to impress your future boss (Use Quasar)
- Add an Author for each Todo and store everything in your Store and DB
- Make Todos as Cards
- Let the user Drag and drop Todos
- And so much more if you have imagination ! :)

Mandatory

- Try, again & again !
- Do not get to much pressure but work hard !
- Take the advantage of your IDE (snippets and so on)
- Use GIT