

RETO TÉCNICO

Ciente: Habitissimo

Autor: David Ramón Chica

15 de noviembre de 2020

Número de propuesta: V1

DOCUMENTACIÓN TÉCNICA

En la carpeta build tenéis compilada la página web para que podáis verla directamente, con sólo descargar esta carpeta junto con la de assets, podéis acceder a la página web. Yo os recomiendo tener en la carpeta donde vayáis a tener el compilado, los archivos de build y la carpeta assets.

Por otra parte, si queréis ejecutarla en un entorno de desarrollo os indico más adelante el entorno de trabajo que he montado para desarrollarla y compilarla. En cuanto al código lo podéis ver dentro de la carpeta components.

Para este proyecto, principalmente he utilizado NPM, NodeJs y Webpack. Lo primero que he hecho es bajarme NodeJs a través de su página web e instalarlo. Dado que toda la programación la he hecho bajo el sistema Mac OSX a través de la consola he instalado todos los paquetes que me han hecho falta a lo largo del desarrollo con npm. Lo siguiente que instalé fue Webpack para poder empaquetar todos los .js e instalar paquetes para las pruebas unitarias etc.

Me he creado un package.json personalizado, donde podéis ver con detalle todas las dependencias que tiene mi proyecto. Desde JQuery, Babel, Jest y Webpack hasta paquetes de style-loader y css-loader. En el apartado scripts, he creado unos “atajos” para poder lanzar las pruebas unitarias y el entorno de desarrollo de webpack.

Con el comando “npm test” ejecuta las pruebas unitarias que he hecho del archivo util.js. El código de las pruebas unitarias lo podéis encontrar en el archivo util.test.js.

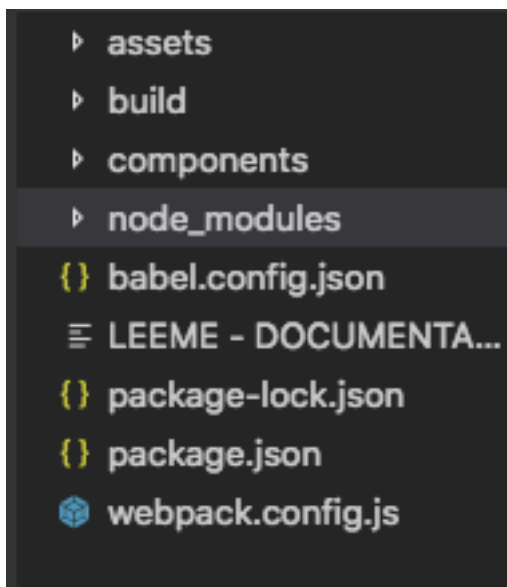
Una vez que tengáis todos los paquetes instalados, si queréis lanzar el proyecto en modo desarrollo, en localhost, yo he utilizado el comando “npx webpack-dev-server” que es equivalente a poner “npm start”.

He utilizado una API pública para la búsqueda, que devuelve un listado de puestos de trabajo en JSON(está en inglés). En la llamada después del contains le paso el valor del input. La llamada que he utilizado ha sido esta: “http://api.dataatwork.org/v1/jobs/autocomplete?contains=”

La página web es responsive y fiel al diseño que vuestro equipo de UX ha realizado. He realizado un control de errores a través de un try catch y un .fail cuando se hace la llamada a la API pública. A parte en el archivo util.js he programado un par de funciones para programación defensiva, comprobando si un valor es vacío, nulo o undefined.

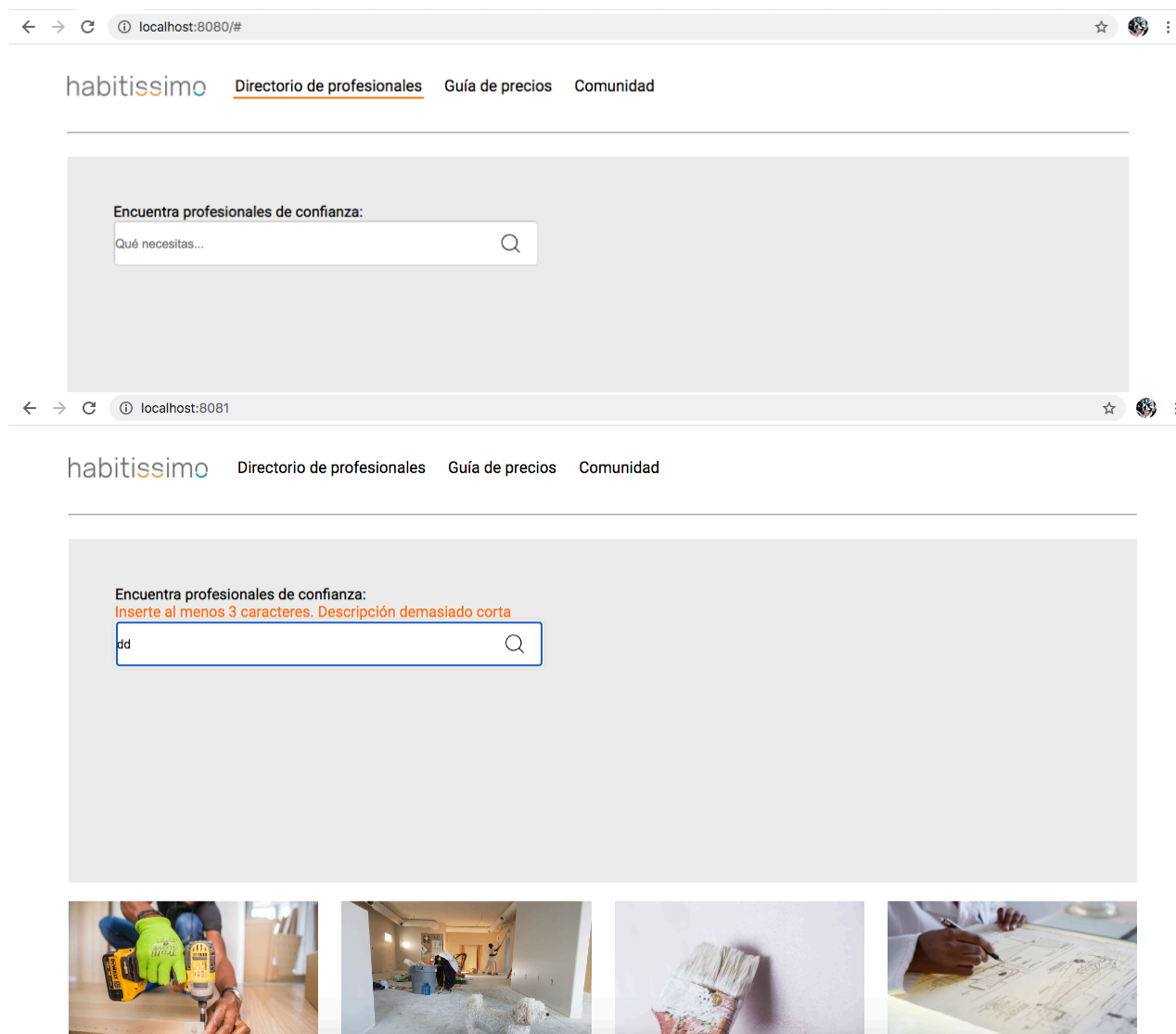
Estructura del proyecto

He dividido el proyecto en varios archivos. Esta es la estructura que se puede visualizar:



Os he dejado compilada la página web en la carpeta build. Descargaros la carpeta assets y la ponéis junto al index.html y el bundle.js. Con sólo ejecutarla ya podéis visitarla y probarla. No necesitáis montar todo el entorno de desarrollo con NodeJS y Webpack. Lo he hecho así como un poco simulando que la página web se subiría a un entorno de producción o explotación.

Adjunto capturas de pantalla de la página web y del funcionamiento tanto en diferentes resoluciones como en dispositivos móviles.



Encuentra profesionales de confianza:

81 coincidencia/s

soft

network software manager
programming and software development project manager
software publisher
software installer
software trainer
software quality engineer
software tester
software project manager
bioinformatics software engineer
software consultant
software programmer
system software developer



habitissimo © 2009 - 2020



Servicios más populares

Reformas viviendas

VER MÁS >>



Servicios más populares

Reformas cocinas

VER MÁS >>



Servicios más populares

Pintores

VER MÁS >>



Servicios más populares

Arquitectos

VER MÁS >>

Lorem, ipsum dolor.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquid ex ea commodo consequat. Quis aute
iure reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non
proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.

Saber más >>

He ido un poco más allá del reto y cuando se selecciona una opción de la búsqueda se puede ver un poco más de información. Se puede visualizar el ID del resultado de búsqueda en el card de la derecha.

←

→

↺

localhost:8081

☆

habitissimo

Directorio de profesionales


Guía de precios

Comunidad


Encuentra profesionales de confianza:
81 coincidencia/s


soft


network software manager
programming and software development project manager
software publisher
software installer
software trainer
software quality engineer
software tester
software project manager
bioinformatics software engineer
software consultant
software programmer
system software developer




ID: 11a3fde03891cabf49f6df1656abb0f6
Occupation: software publisher

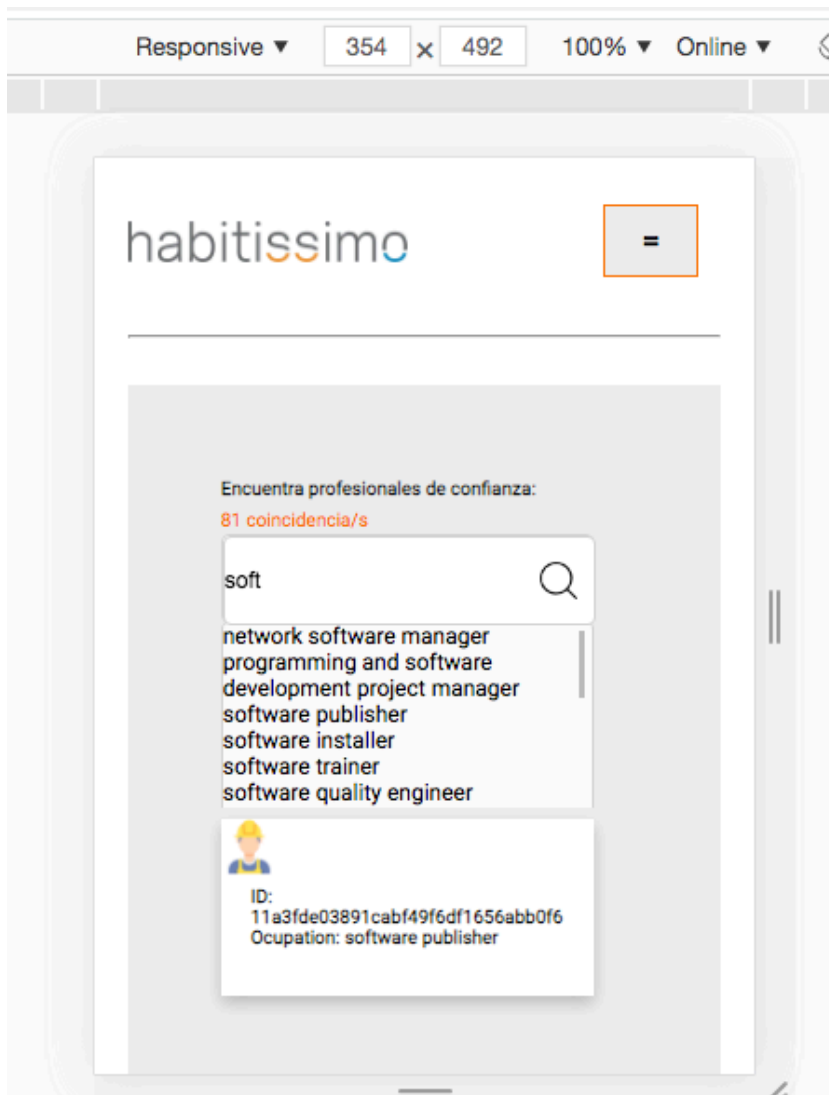


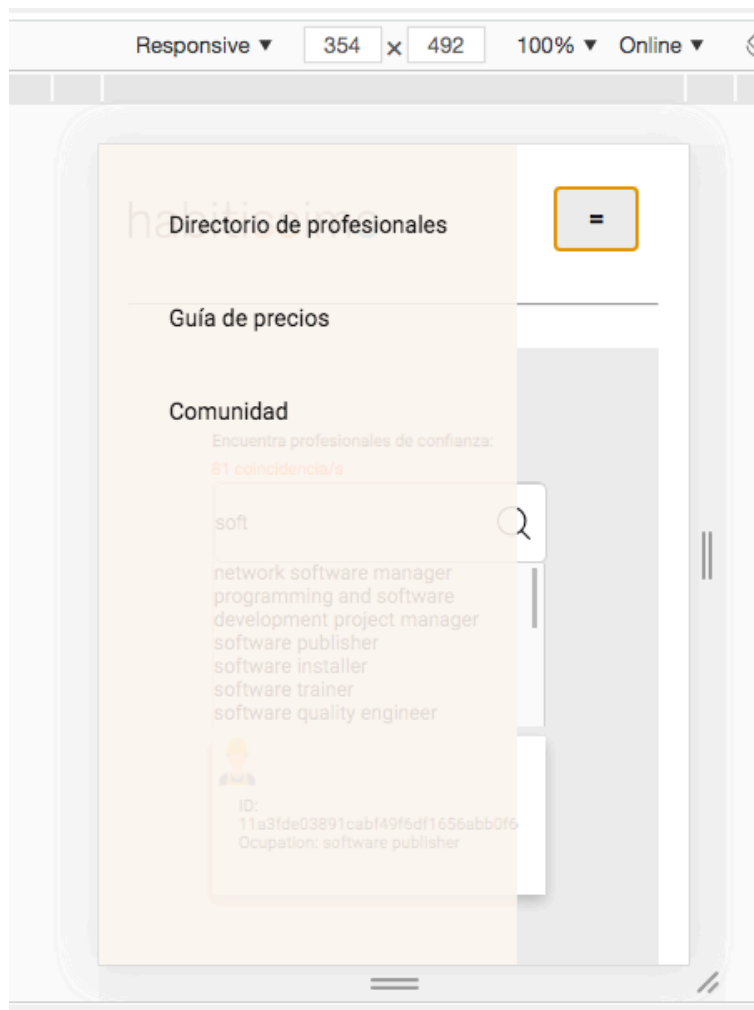






DISEÑO RESPONSIVE





COMANDOS UTILIZADOS EN LA CREACIÓN Y DESARROLLO DEL PROYECTO:

npm init —> para crear el package.json personalizado.
npm install jquery
npm install webpack@4.42.2 —save-dev
npm install webpack-cli@3.3.12 —save-dev
npm install webpack-dev-server@3.11.0 —save-dev o npm i webpack-dev-server
webpack —> para actualizar el contenido del servidor de desarrollo.
npm install html-webpack-plugin —> procesa los archivos html en webpack.
npm i style-loader css-loader
npm i —save-dev jest
npm install @babel/preset-env —save-dev
npm install style-loader —save
npm install css-loader —save

para compilar en desarrollo
npx webpack-dev-server

para compilar en producción y generar los archivos index.html y bundle.js en la carpeta build
npx webpack -p

GESTIÓN DE EXCEPCIONES

En el index.js podéis observar que la llamada JQuery.ajax() que hago a la API publica, está englobada en un try catch. A parte utilizo .fail() que se ejecuta cuando en la solicitud ha ocurrido algún error.

```
    })
    .fail((jqXHR, textStatus, errorThrown) => {
      $("#search-result > li").remove();

      if (textStatus === "parsererror") {
        $("#search-filter-count").text("Error al procesar la búsqueda");
      } else if (textStatus === "timeout") {
        $("#search-filter-count").text(
          "Error: Tiempo de espera agotado en la solicitud"
        );
      } else if (textStatus === "abort") {
        $("#search-filter-count").text("Error: solicitud rechazada");
      } else if (
        jqXHR.status === 0 ||
        jqXHR.status === 404 ||
        jqXHR.status === 500
      ) {
        if (isValueEmpty(errorThrown)) {
          $("#search-filter-count").text(
            "En estos momentos no se ha podido realizar la búsqueda"
          );
        } else {
          $("#search-filter-count").text(
            "En estos momentos no se ha podido realizar la búsqueda. Motivo: " +
            errorThrown
          );
        }
      }

      throw "Error al realizar la búsqueda";
    });
  } catch (e) {
    $("#search-filter-count").text(
      "En estos momentos no podemos realizar su búsqueda. Pongase en contacto con nuestro soporte técnico. Disc"
    );
  }
}
```

PRUEBAS DE LOS TEST UNITARIOS CON JEST

Para ejecutar las pruebas, he creado un acceso rápido en el package.json, podemos acceder a él con "npm test"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> jest

(node:10571) ExperimentalWarning: The fs.promises API is experimental

PASS components/util.test.js (10.969 s)

- ✓ probar que un valor es nulo (10 ms)
- ✓ probar que un valor es undefined
- ✓ probar que un valor es empty (1 ms)
- ✓ probar que funciona cuando tiene un valor

Test Suites: 1 passed, 1 total

Tests: 4 passed, 4 total

Snapshots: 0 total

Time: 19.596 s

Ran all test suites.

MacBook-Pro-de-David:Habitissimo Prueba davidramonchica\$

Test Suites: 1 passed, 1 total

Tests: 4 passed, 4 total