UNIDAD 1 – Automatización de Tareas

1. [UNIDAD 1: Introducción](https://learn.nextu.com/mod/page/view.php?id=3587&pid=P_WEB_APPS)

Bienvenido a **Herramientas, Build Systems y tecnologías emergentes**

La automatización de tareas permite ejecutar aquellas que, por lo general, están escritas como plugins. En esta Unidad aprenderás cómo integrar a tus proyectos las más usadas en el desarrollo de JavaScript, siempre considerando cuáles son las más beneficiosas. Es decir, reconociendo cuáles tareas facilitarán el trabajo o, por el contrario, cuáles lo hacen más complejo.

Puntos de aprendizaje

Unidad 1: Automatización de tareas

Los objetivos del Programa que corresponden a esta Unidad son:

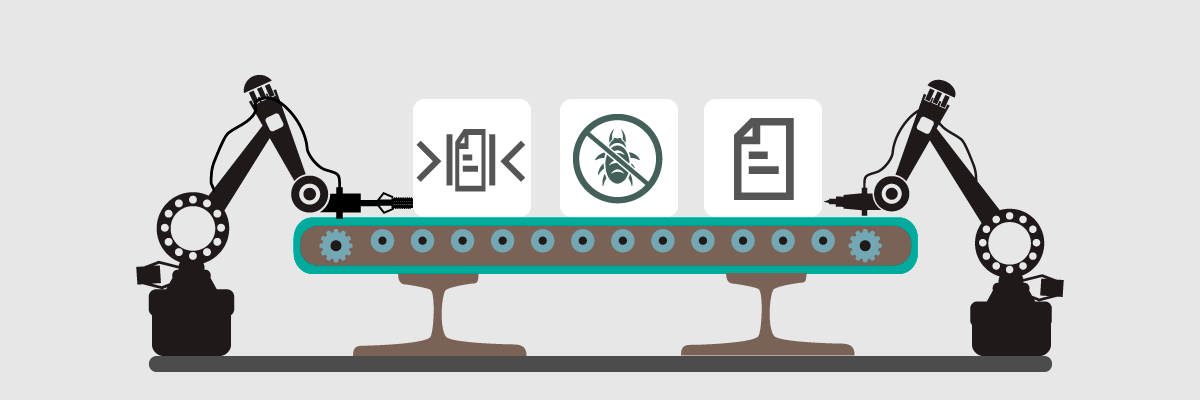
* Crear ambientes de programación y a gestionar recursos.
* Implementar la construcción de sitios web a partir de componentes.

Los objetivos del Curso 8 que corresponden a esta Unidad son:

* Identificar tendencias web.
* Aplicar el concepto ambiente web.
* Usar herramientas para la gestión de tareas.

1. Lección 1: Task Runners (Automatización de tareas)
   1. [Lectura: Introducción a la automatización de tarea](https://learn.nextu.com/mod/lesson/view.php?id=3588&pid=P_WEB_APPS)

**La Automatización de Tareas**



¿Qué es?

La automatización nos permite ejecutar tareas que por lo general están escritas como plugins. Veamos cuáles podemos ejecutar y cómo llamarlas.   
  
Las siguientes son algunas de las tareas más usadas en el desarrollo en Javascript. Úsalas como inspiración para integrarlas a los proyectos en los que creas que puedan ser beneficiosas pero recuerda: siempre debes pensar si implementar una nueva tarea facilitará tu flujo de trabajo o, por el contrario, lo complica. ¿Por qué? A veces podemos tener infraestructuras complejas de tareas que en lugar de reducirnos el trabajo lo incrementan.

Tareas más comunes de automatización

1. Minify

Consiste en comprimir código, ya sea js - css - html o cualquier otro archivo, en un archivo de lógica, estilo y estructura equivalente que pesa menos, eliminando espacios innecesarios, reduciendo los nombres de variables a pocas letras y eliminando comentarios.

2. Linting

Se refiere a mostrar posibles bugs del código sin ejecutarlo y advertencias o errores que se pueden agregar, para ayudarnos a seguir guías de estilo o evitar patrones de código que indican malas prácticas.

3. Transpiling

Es el proceso de convertir un lenguaje interpretado a otro, es equivalente a compilar.   
En este caso el lenguaje objetivo será Javascript pero podemos partir desde otras versiones de Javascript (ES6), código en otros lenguajes ya existentes (Java, Scala, Ruby, Python, C/C++) o lenguajes creados exclusivamente para ser transpilados a js (Coffeescript o Typescript).

4. Licensing and Versioning

Podemos aprovechar cada vez que construimos nuestro proyecto para escribir en el compilado final (build) la licencia de nuestro código, información de autoría y versiones.

5. Sourcemaps

Si hacemos procesos de minificacion o transpiling podemos generar mapas de código que nos permitan ver cómo se transformó nuestro código y utilizar esta información para debugear el código.

6. Rename, delete, create y map functions through files

También es muy fácil ejecutar tareas propias de los sistemas de archivos, reemplazar, cortar pedazos, renombrar archivos, borrarlos, crear nuevos archivos copiando anteriores o procesar archivos con funciones y luego guardarlos nuevamente con otros nombres. Todas estas tareas son sencillas y muy poderosas.

7. Documentación

Existen múltiples herramientas que nos permiten crear la documentación en el momento de compilación para tenerla siempre al día.   
Usos más avanzados nos permiten, por ejemplo, generar la documentación y, si estamos desplegando a producción, actualizar el sitio estático que mantiene la documentación con la nueva versión de la documentación y, además, medir cómo es el cubrimiento de la documentación de la base de código, a medida que crece.

8. Ejecución y cubrimiento de pruebas

Así como podemos generar la documentación cada vez que construimos el proyecto, también podemos correr las pruebas unitarias que hayamos creado y ver el cubrimiento, así como podemos detener la carga del código a la producción si éstas fallan y emitir alarmas a los desarrolladores.

9. Despliegue automático de integración continua

La integración continua es sencilla de implementar si utilizamos algunas tareas que ejecuten las herramientas respectivas. La rama ‘production’ de tu repositorio puede ser probada (tests unitarios) y luego desplegada así: si todo sale bien, todos los días en la madrugada, justo cuando el tráfico en el sitio es el mínimo, y si algo sale mal, puedes configurar emails o alertas automáticas a los desarrolladores.

10. Uso automático de repositorios

Imagina configurar útiles metodologías de trabajo como por ejemplo: que cada vez que tu código compile y estén todas las pruebas bien, automáticamente se actualice tu rama de desarrollo de git, o con un sólo comando combine la rama de producción, active hooks de git en los servidores relevantes y despliegue el código.

11. Servir aplicaciones

Existen también múltiples herramientas que fácilmente nos dan servidores de desarrollo con características como sockets, o que incluso pueden emular bases de datos de mockup o ambientes particulares.

12. Descargar URL's

Podemos escuchar URLs específicas donde se desplieguen los recursos y descargarlos para ponerlos en nuestra versión actual de la aplicación.

13. Watchers

Nos permiten observar continuamente archivos y actuar ante el cambio en estos. Entonces, si guardamos el archivo en el editor o un archivo en particular, por ejemplo, podemos aprovechar para hacer un git pull - merge - push y servir de nuevo la versión de desarrollo.

14. Livereload

Existen plugins que también nos permiten empujar nuevas versiones de la aplicación, a clientes que ya están ejecutándose, usando sockets de manera automática.

15. Procesamiento de imágenes

Muchas tareas como descargar, generar thumbnails, cortar o procesar imágenes para añadir marcas de agua o efectos, pueden ser ejecutadas en tiempo de compilación de las tareas.

16. Servir y subir archivos vía FTP / SSH

También podemos crear servidores FTP directamente desde nuestras tareas o pedir y enviar archivos a un servidor vía SSH, lo que nos permite desplegar nuestra aplicación directamente en nuestras tareas.

17. Renderizar componentes web en el servidor

Podemos utilizar emulación del browser en el lado del servidor para que si nuestros clientes no pueden ejecutar el Javascript de nuestras páginas les sirvamos una versión ya renderizada en html. Estas versiones son generadas por las tareas con herramientas como phantomjs.

18. Construir múltiples versiones de nuestras aplicaciones

Si nuestro código de Javascript, por ejemplo, alimenta un sitio web, una aplicación de electrón (escritorio) y una aplicación móvil, podemos construirlos todos a la vez , probarlos todos a la vez, o hacer cualquier tarea que se nos ocurra con tantos objetivos de compilación como queramos.

19. Enviar emails

Crear y enviar emails de manera automatizada para avisar de cambios de versión, despliegues de código, alarmas, entre otros.

20. Gestionar archivos estáticos en CDN’s

Si nuestra infraestructura de archivos estáticos se vuelve muy compleja, podemos administrar múltiples CDN (Redes de envío de contenido), cambiar estas URLs en nuestros archivos estáticos y automatizar toda esta infraestructura de contenido fácilmente.  
Con la variedad de tareas que podemos automatizar, podremos realizar diversos proyectos muy interesantes, llevando tareas manuales muy complejas a una forma simple y fácil, complementando nuestros proyectos con muy buenas tecnologías.

* 1. Actividad 1

Tiempo de pensar

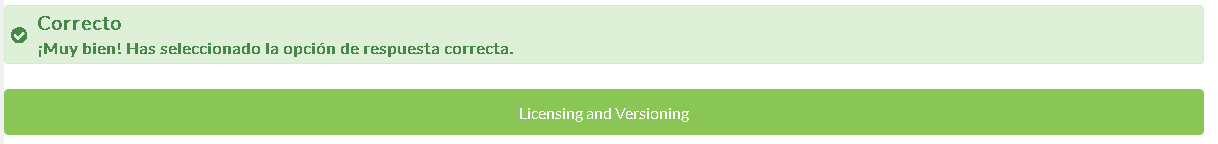
Ahora que sabes que la automatización nos permite ejecutar tareas que por lo general están escritas como plugins y cuáles son las más usadas, te invitamos a realizar esta actividad.   
  
Instrucciones:  
  
Asocia los nombres de algunas de las tareas más comunes de automatización, ubicados en la columna izquierda, con su respectiva definición, en la columna derecha.  
¡Adelante!



* 1. Actividad 2

Tiempo de pensar

Recuerda que siempre debes pensar si implementar una nueva tarea facilitará tu flujo de trabajo o, por el contrario, lo complicará. ¿Por qué? A veces podemos tener infraestructuras complejas de tareas que en lugar de reducirnos el trabajo lo incrementan.  
  
Instrucciones:  
  
A continuación te presentamos una definición.  
Deberás leerla y determinar a cuál de las tareas de automatización que se plantean en las opciones de respuesta corresponde.¡Ánimo!  
  
Difinición:  
  
Con esta tarea podemos aprovechar cada vez que construimos nuestro proyecto para escribir, en el compilado final (build), la licencia de nuestro código, información de autoría y versiones.



* 1. Ejecutando mi primera tarea

Comando para ejecutar un prueba con npm:

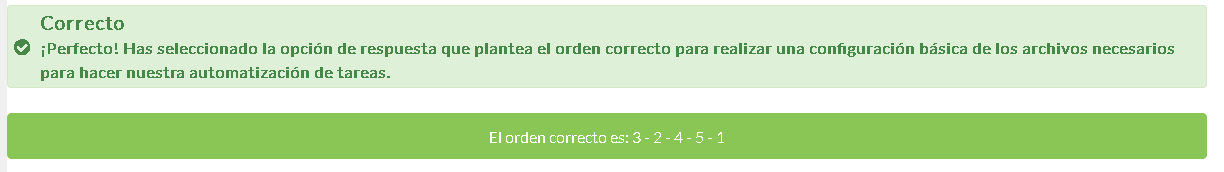
npm run test

* 1. Actividad 3

Tiempo de pensar

Ahora que sabes que la automatización te permite ejecutar tareas que por lo general están escritas como plugins, te invitamos a realizar la siguiente actividad.  
  
Instrucciones:  
  
A continuación presentaremos unos enunciados que corresponden a los pasos para realizar una configuración básica de los archivos necesarios para realizar nuestra automatización de tareas. El objetivo es que realices un análisis que te permita ordenarlos mentalmente y que, luego, elijas la opción de respuesta que plantea el orden correcto. Entonces, lee con atención y, cuando estés seguro, responde.  
  
Enunciados:

1. Para ejecutar las tareas que definimos en este lugar, realizamos la siguiente operación en la consola “npm run test”. Esta operación imprime el mensaje escapeado dentro de comillas.
2. Después aparecerá dentro de la consola una solicitud de ciertos campos que responderemos con los datos solicitados.
3. Abrimos un terminal de nuestra computadora y nos dirigimos con nuestra terminal al lugar de hayamos creado nuestro proyecto. Por medio del siguiente comando iniciamos un nuevo proyecto nodejs npm init.
4. Confirmamos y hemos creado nuestro nuevo proyecto. Aparece un nuevo archivo package.json
5. Debido a que este archivo está en formato json, buscamos la clave nombrada con scripts. Allí encontramos la clave con el nombre de test que viene por defecto cuando creamos nuestro primer proyecto node.



2.6 Laboratorio

LABORATORIO DE  
**Instalación y Configuración**

La empresa de conciertos Luz Azul cuenta con un sitio web estático donde ponen la información relevante a los instrumentos que utilizan y, en el futuro, pondrán la información de los conciertos realizados. El director de la orquesta quiere empezar a promocionar los conciertos de manera internacional y ha encomendado al director de tecnología que se implemente una estrategia de internacionalización para la página. También continuamente el equipo de fotografía e imagen cambia los recursos que se muestran en la página web y esto implica esfuerzos por parte del equipo de tecnología de la empresa.

El director de tecnología te ha pedido que, como desarrollador web, crees una estrategia de internacionalización en la cual se produzca el sitio web de Luz Azul en cada uno de los idiomas que se planean introducir. Además ha resaltado otros aspectos: debes crear una manera en la que los recursos sean definidos por el equipo de fotografía e imagen y se integren al sitio web de manera automática.  
En una revisión de código del sitio actual se encontró que usa foundation, jquery y what-input.js como dependencias de código javascript / css. Debes instalar estas dependencias en el sitio y la infraestructura de tareas que diseñamos.

Contenido:

Para el desarrollo de este laboratorio debes descargar el código base que contiene la página estática de Conciertos Luz Azul.

Contenido

Instrucciones

* 1. **Estructura del proyecto:**  
       
     Descarga el código base e inspecciónalo.  
     Indica la estructura del proyecto y cuál es la función de cada carpeta. Luego describe la estructura del archivo index.html, indicando donde se incluyen las hojas de estilo, los scripts de Javascript y las imágenes que se muestran en la página.  
       
     El proyecto consiste de un archivo index.html donde está el código html del proyecto. Adicionalmente tenemos 3 carpetas css, js y img.
     + **Carpeta js:** Contiene los archivos Javascript del proyecto, adicionalmente contiene una carpeta vendor con las dependencias en Javascript.
     + **Carpeta img:** Se encuentran todas las imágenes que provee el equipo gráfico
     + **Carpeta css:** Contiene las dependencias de css de la librería Foundation y el css de la aplicación en app.css
  2. **Manejo de scripts:**  
       
     Diseña las tareas que se deben ejecutar para los manejos de scripts de las librerías usadas.
     + Los scripts deben ser descargados por un **manejador de paquetes** que permite, a su vez, descargar las dependencias de manera local.
     + Se debe mantener un archivo para saber qué librerías y en qué versión deben estar instaladas. **(Package.json)**
     + Se debe hacer una **tarea** que revise qué archivos son dependencias de la aplicación y los contenga en una carpeta haciendo **copias de los archivos** para la versión que se va a construir.
     + Estos archivos .js pueden juntarse en un solo archivo **vendor-bundle.js,** por ejemplo, utilizando una **tarea que los concatene.**
     + Los scripts de la aplicación también pueden juntarse con **vendor-bundle.js** en **app.js** para servir un sólo archivo Javascript con otra **tarea de concatenación.**
     + El nombre del archivo a importar debe ser inyectado en el **index.html** por una tarea de **reemplazamiento.**
  3. **Manejo de recursos:**  
     Diseña las tareas que implementan una infraestructura para el manejo de los recursos gráficos, creados por el equipo de fotografía.  
     Para eso, realiza una **tarea de reemplazamiento,** tomando las llaves desde el archivo json y las reemplaza por las correctas en todos los html.
  4. **Estrategia de localización del sitio web:**  
     Diseña la infraestructura de tareas requerida para generar las versiones localizadas del sitio web. Debes tener archivos html esqueleto que sean los que se tomarán de base, utilizando un diccionario de etiquetas reemplazar en cada bundle para cada sitio web estático generado. Index.html finalmente detectara el lenguaje a cargar y luego inyecta el html necesario según el lenguaje.
     + Carpeta con plantillas html
     + Carpeta con diccionarios en JSON
     + Construcción del bundle html para cada lenguaje con tareas de reemplazamiento y luego copia.
     + Meta Plantilla de index.html que reemplaza el index.html y mediante javascript detecta qué lenguaje debe ser cargado para luego, cargarlo.

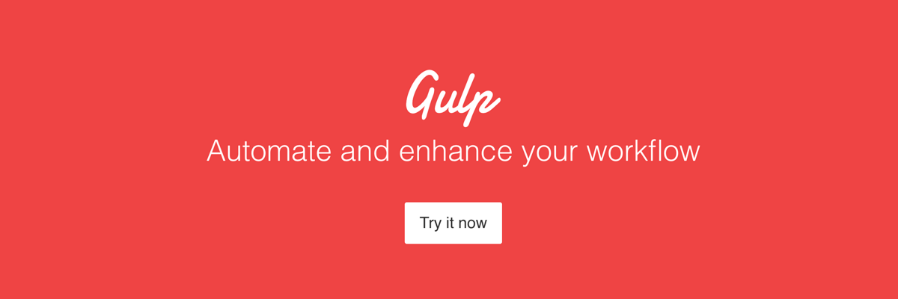
Buenas Prácticas

Recuerda que puedes guardar varias versiones de archivos intermedios a medida que tu cadena de procesamiento de tareas se va completando. Usualmente es mejor servir al final la menor cantidad de archivos posibles para reducir la cantidad peticiones.

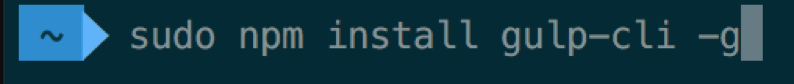
1. Lección 2: Gulp.JS

3.1 Lectura: Qué es Gulp.JS y cómo lo implementamos

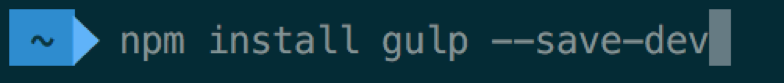
¿Qué es Gulp.JS y cómo lo implementamos?



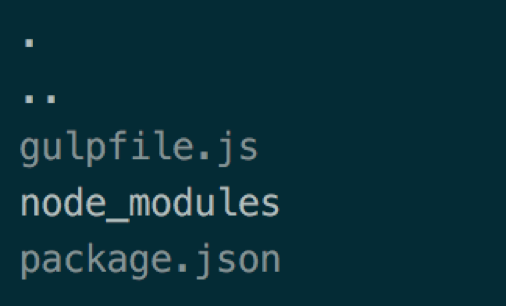
Es una herramienta en forma de script, escrita en Nodejs, que ayuda a automatizar tareas comunes en el desarrollo de aplicaciones. Muchos programadores front-end solemos realizar tareas muy repetitivas. Esta herramienta fue creada para facilitar el proceso de desarrollo, gastando menos tiempo en el proceso de escribir código, realizar validaciones y pruebas de nuestro código.  
  
Para iniciar un nuevo proyecto con esta herramienta tan sólo debemos ejecutar unos pocos comandos para tener nuestros paquetes listos para su uso. Son los siguientes:



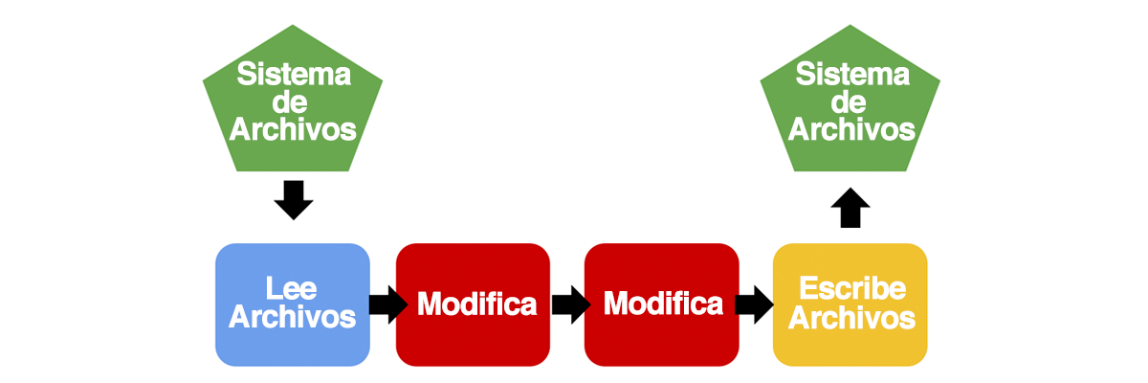
Después de instalar nuestro paquete de forma global, nos ubicamos en nuestra base de nuestro proyecto, donde encontramos nuestro archivo package.json y instalamos el siguiente paquete:



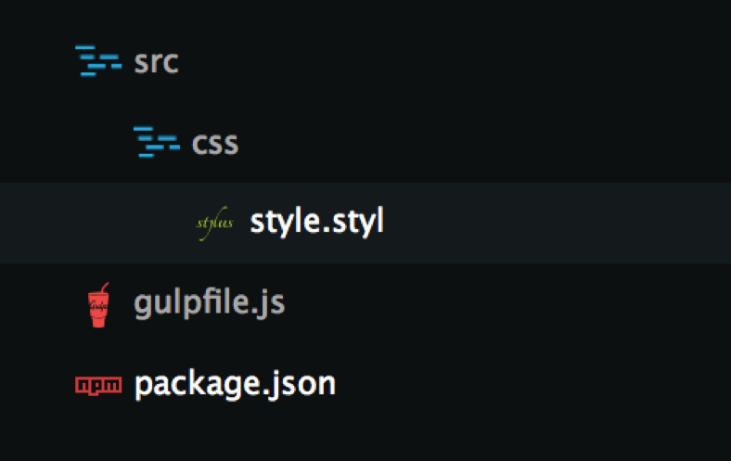
Este comando indica que vamos a instalar un paquete o librería con el nombre gulp y que deseamos que se guarde como un librería de dependencia, únicamente de desarrollo, que no la vamos a necesitar obligatoriamente en producción, sino que únicamente la usaremos en un entorno de desarrollo donde hacemos normalmente nuestras pruebas locales, mucho antes de realizar un proceso de lanzamiento en nuestro sitio oficial.  
  
Enseguida, después de realizar este proceso, creamos en la base de nuestro proyecto un nuevo archivo con el nombre de gulpfile.js donde definiremos nuestras tareas o procesos a automatizar, en la base de nuestro proyecto deberíamos de tener la siguiente estructura:



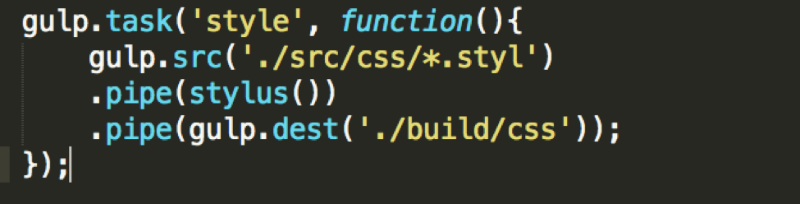
Para ver realmente el funcionamiento de esta herramienta, debemos conocer la forma en que trabaja, en la siguiente imagen veremos la forma en que trabaja Gulp.



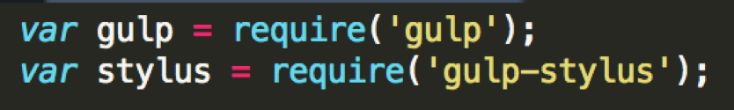
Las tareas que normalmente realizamos en el desarrollo de aplicaciones, suelen ser tareas como preprocesadores de estilos, ya sean realizadas con Stylus, Sass, Less, entre otros, que vienen siendo archivos físicos con la sintaxis determinada por la herramienta que se haya usado. Estos archivos físicos no son válidos o no podemos usarlos directamente en nuestro navegador, sino que debemos hacer una tarea de procesar ese archivo para obtener nuestro archivo válido o que nuestro navegador pueda entender o interpretar.   
  
En este caso sería pasar un archivo .styl a .css. Aquí veríamos el flujo real de gulp, pues tomamos los archivos que necesiten ser procesados y luego modificados, procesados o concatenados y reescritos en nuevos archivos, que serán los archivos que agregaremos a nuestro aplicativo o desarrollo que estemos realizando.  
  
Veamos un ejemplo sencillo de procesar un archivo stylus a css:  
  
En la base de nuestro proyecto creamos la siguiente estructura de carpetas “src/css/”.   
Dentro de esta estructura creamos un archivo estilo.styl que viene siendo archivo que vamos a procesar por medio de gulp. Deberíamos tener, hasta este punto, la siguiente estructura y archivos:



Ya que contamos con estos archivos y nuestra estructura de carpeta deseada, programaremos nuestra tarea en el archivo gulpfile.js, que procese nuestro archivo estilo.styl y como resultado, crearemos la siguiente estructura de carpetas “build/css/estilos.css” desde la base de nuestro proyecto.



Como vemos, registramos una nueva tarea de gulp con la siguiente función gulp.task(‘’, ‘’) que es la función que utilizamos para registrar nuevas tareas en gulp.  
Registramos nuestra tarea con el nombre de “style” y a continuación pasamos una función anónima.   
La función que encontramos dentro de la función anónima, osea, gulp.src(‘’), es la que realiza la búsqueda del archivo o archivos que vamos a procesar.   
A continuación encontramos la función .pipe(), dentro de esta función. Ejecutamos una función que va a ser la encargada de procesar o convertir nuestro archivo o archivos según lo que hemos asignado en el gulp.src(‘’).   
La función que se ejecuta en este caso es stylus(), pero te preguntarás de dónde proviene esta función. Pues bien, la definimos en un lugar previo, antes de definir nuestras tareas, así:

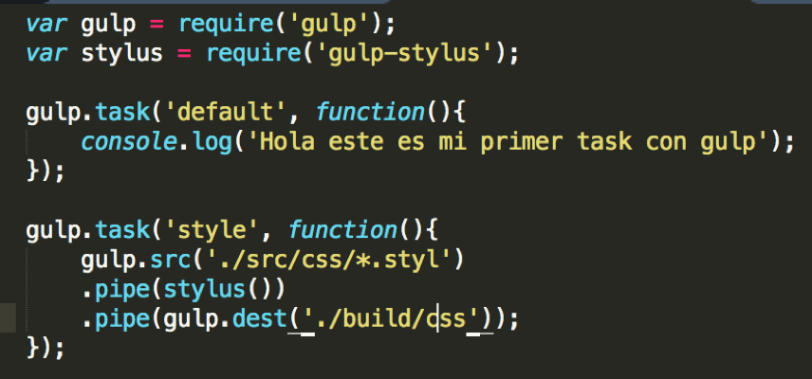


Para que realmente todo funcione correctamente, debemos instalar primero nuestros paquetes necesarios y luego definirlos dentro de nuestro archivo gulpfile.js  
En este caso tenemos dos paquetes que debemos instalar:

El que ya hemos mencionado y es con el siguiente comando: “npm install gulp --save-dev”

Y una nueva librería, esta es la encargada de procesar los archivos .styl y la instalamos con el siguiente comando “npm install gulp-stylus --save-dev”

De esta forma ya tendríamos todo finalizado y listo para ejecutar nuestra primera tarea automatizada en nuestros flujos de trabajo diario.   
  
Este sería nuestro resultado a nivel de archivo gulpfile.js.



Y la ejecución de esta tarea daría el siguiente resultado:

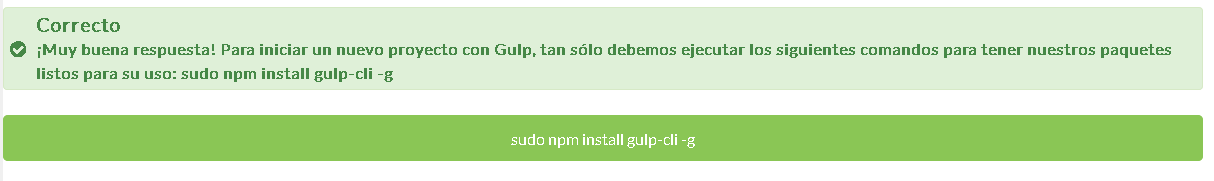


3.2 Actividad

Tiempo de pensar

Como ya lo sabes, Gulp es una herramienta en forma de script, escrita en Nodejs, que ayuda a automatizar tareas comunes en el desarrollo de aplicaciones.  
  
Instrucción:  
  
Lee con atención la pregunta y, para completar la acción que allí se menciona, selecciona la opción de respuesta que consideras correcta.

Pregunta:  
  
Para iniciar un nuevo proyecto con Gulp, tan sólo debemos ejecutar unos pocos comandos para tener nuestros paquetes listos para su uso. ¿Cuáles son?



3.3. Lectura: Funciones esenciales de Gulp

Funciones esenciales de Gulp

  
  
Las siguientes son funciones que trae la librería Gulp y que nos facilitan la programación:

**src(‘url-file’ o [‘url-file’,’url-file’]):**

Esta función devuelve un archivo o una lista de archivos en formato blob (un formato manejado por Javascript). Esta función recibe como parámetros un listado o array de string, que vienen siendo direcciones de archivos o patrones para identificar archivos. Además, nos ayuda a obtener los archivos que vayamos a procesar, sea el caso de procesar hojas de estilos o templates como jade, entre otros ejemplos.

**dest(‘’):**

Esta función no facilita el traslado de los archivos procesados por gulp, realmente aplica como un especie de return o respuesta final de cada tarea, aunque realmente no siempre devolvamos archivos físicos procesados. Nos permite dar un lugar en específico a los archivos que acaban de ser tratados, según las indicaciones que hayamos dado a nuestra tarea.   
Un ejemplo muy claro del uso de esta función es cuando procesamos un archivo de algún preprocesador de estilos ya sea Stylus, Sass, Less y queremos darle destino a nuestro archivo resultado, dentro de la función .pipe(gulp.dest(‘./build/css’)). Este sería un ejemplo claro de la forma en que utilizamos esta función.

**task(‘’, [‘’], fn):**

Esta función nos permite definir nuestras tareas o una operación a automatizar. El primer parámetro que recibe esta función es el nombre que le vamos a asignar a esta tarea. Este nombre debe ir entre comillas ‘nombre fn’, seguido de este parámetro que es opcional, es un listado o un array de tareas ya definidas, que se quiere que se realicen antes de ejecutar nuestra tarea que hemos definido, ejemplo, [‘compitarjs’,’compilarcss’] entre otros. Y en nuestro último parámetro que lo he definido como fn o función, asignamos una función anónima, que la definimos normalmente de la siguiente manera, function(){}, de esta forma podemos definir nuestras tareas.

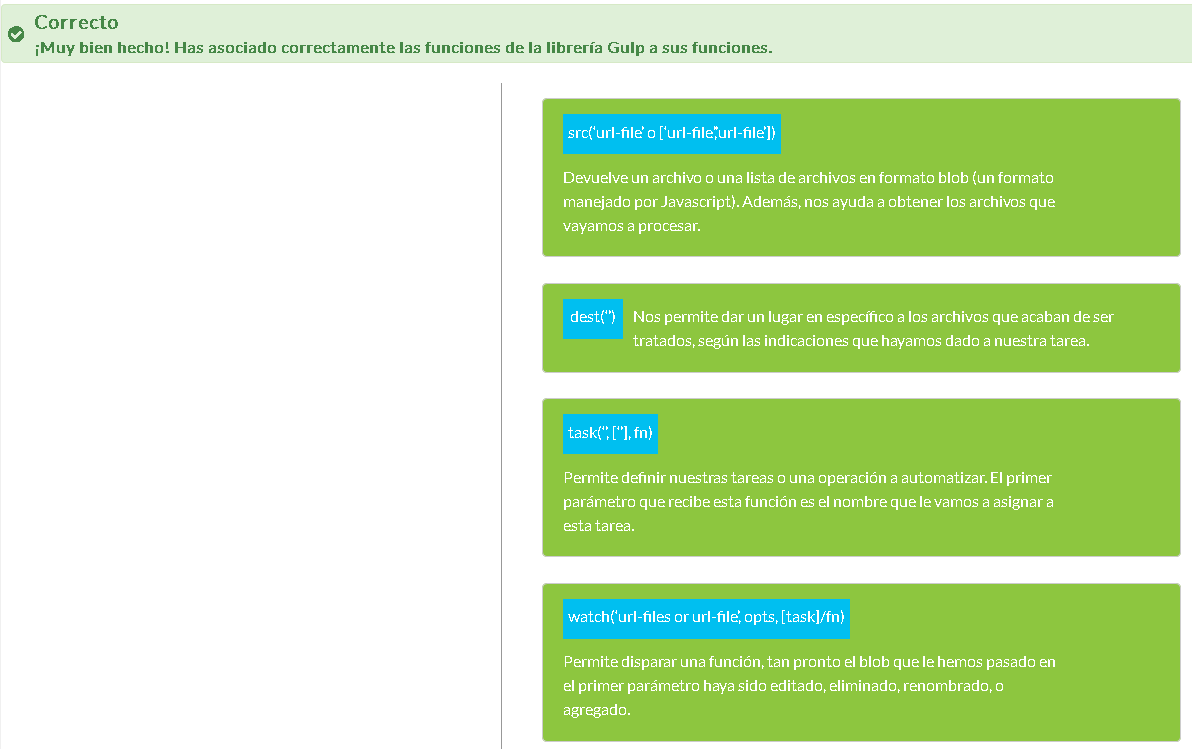
**f(‘url-files or url-file’, opts, [task]/fn):**

Esta función nos permite disparar una función, tan pronto el blob que le hemos pasado en el primer parámetro haya sido editado, eliminado, renombrado o agregado. Esta función es muy útil ya que nos ayuda a automatizar, ya sea una tarea o un conjunto de tareas.  
  
Ya que hemos conocido las funciones básicas de gulp, podemos aplicar estas funciones en tareas más comunes en el desarrollo de aplicaciones que realizamos diariamente, tales como concatenar archivos javascript o poder realizar un transpiler de archivos escritos en Ecma6, minificar archivos js o css, procesar hojas de estilos escritas ya sea en Stylus, Less o Sass, entre muchas otras tareas que encontramos en el diario vivir de un desarrollador o programador front-end.

3.4. Actividad 2

Tiempo de pensar

Recordemos que Gulp por ser una herramiento o un paquete, trae algunas funcionalidades o funciones que a lo largo de nuestro desarrollo de aplicaciones podremos utilizar para automatizar nuestras tareas más comunes.  
  
Instrucciones:  
  
Asocia las funciones de la librería de Gulp, ubicadas en la columna izquierda, a las descripciones de la columna derecha.



3.5. Probando Gulp

Para hacer un test

npm install --save-dev gulp

npm install --global gulp-cli

Para correr una tarea de Gulp:

gulp TaskName

Para instalar gulp-uglify:

npm install gulp-uglify --save-dev

Para instalar una versión especifica de Gulp:

npm install [gulp@3.9.1](mailto:gulp@3.9.1)

npm install --save-dev gulp@3.9.1

Nota para version de Gulp 2.x o 3.x, los watch se definen de la siguiente manera:

gulp.watch('./src/js/\*.js', ['mainminjs']);

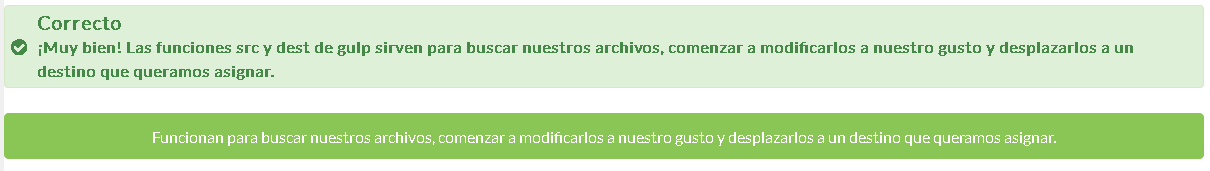
Para Gulp 4.x, los watch se definen de la siguiente manera:

gulp.watch('./src/js/\*.js', gulp.series('mainminjs'));

3.6. Actividad 3

Tiempo de pensar

Ya hemos definido gulpjs y conocemos teóricamente cómo funciona esta herramienta de automatización de tareas, ahora queremos invitarte a realizar la siguiente actividad.   
  
Instrucción:  
Lee con atención la pregunta y responde seleccionando una única opción.   
  
Pregunta:  
¿Para qué sirven las funciones src y dest de gulp?



3.7. Laboratorio

LABORATORIO DE  
**Instalación y Configuración**

45 MINUTOS

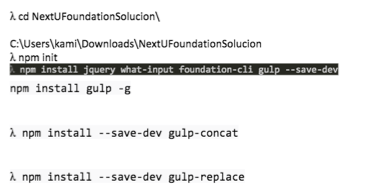
La empresa de conciertos Luz Azul cuenta con un sitio web estático donde ponen la información relevante a los instrumentos que utilizan y, en el futuro, pondrán la información de los conciertos realizados. El director de la orquesta quiere empezar a promocionar los conciertos de manera internacional y ha encomendado al director de tecnología que se implemente una estrategia de internacionalización para la página. También continuamente el equipo de fotografía e imagen cambia los recursos que se muestran en la página web y esto implica esfuerzos por parte del equipo de tecnología de la empresa.  
Se ha diseñado una infraestructura de tareas para el manejo de los scripts, la internacionalización y el manejo de los recursos.  
  
Ahora que se ha diseñado la estrategia de automatización de la página web, el director de tecnología la ha aprobado y quiere empezar a implementar la infraestructura de tareas que automatizan el sitio. Éste es el propósito de este laboratorio.  
Para ello debes utilizar gulp.js y npm para automatizar los procesos requeridos.  
  
El director de tecnología también se ha encargado de curar las cadenas de texto de la aplicación para producir las versiones localizadas, por lo que debes integrar los archivos JSON en las tareas y generar las versiones localizadas del sitio.

Contenido:

Para el desarrollo de este laboratorio debes descargar el código base que contiene la página estática de Conciertos Luz Azul.

Instrucciones

* 1. Identifica las librerías utilizadas en el código y quita las etiquetas de dependencias de ellos.
  2. Instala node y npm y luego, utilizando npm, busca las dependencias e instálalas.
  3. Utiliza las tareas que diseñaste para el manejo de scripts, en el laboratorio anterior, para inyectar las dependencias en el html como un sólo archivo de cada tipo.
  4. Implementa las tareas de manejo de recursos gráficos. Recuerda que debes leer una lista de url’s o archivos donde se hayan subido los recursos y luego cambiar estos recursos dentro de los html’s mediante reemplazamientos o encontrar una estrategia equivalente.
  5. Implementa las tareas para generar versiones localizadas del sitio web.  
       
     Finalmente, así debe lucir tu código:



Buenas Prácticas

Recuerda que puedes guardar varias versiones de archivos intermedios a medida que tu cadena de procesamiento de tareas se va completando. Usualmente es mejor servir al final la menor cantidad de archivos posibles para reducir la cantidad peticiones.

1. Lección 3: Grunt.JS
   1. [Lectura: ¿Qué es Grunt.JS y cómo lo implementamos?](https://learn.nextu.com/mod/lesson/view.php?id=3590&pid=P_WEBDEV_V2)

Qué es Grunt.JS y cómo lo implementamos?

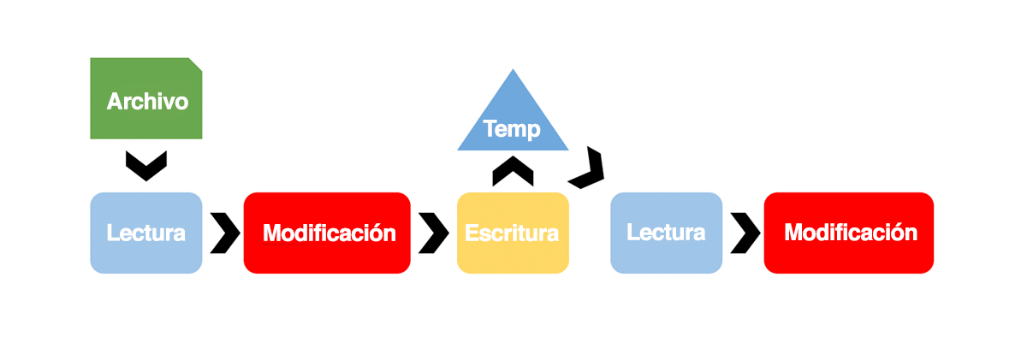


Continuando con nuestras herramientas de automatización, nos encontramos con una librería o paquete de Nodejs llamada Grunt.  
  
Grunt es una herramienta que nos facilita el desarrollo de aplicaciones web pues optimiza nuestro tiempo mientras realizamos tareas complejas.  
  
Es decir que las tareas que anteriormente demandaban mucho tiempo, a pesar de ser realizadas diariamente, se realizan en segundos por medio de una configuración básica en la que creamos un archivo en la base de nuestro proyecto, con el nombre de: gruntfile.js

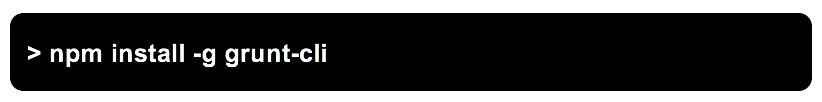
Para eso, debemos:

* Tener instalado Nodejs en cada uno de los computadores en los que realizaremos pruebas o desarrollaremos nuestro proyecto. También debemos contar con el archivo package.json en la base de nuestro proyecto, ya que este archivo nos indicará qué librerías de Nodejs tenemos o hemos instalado, siendo ésta una información muy importante al momento de configurar nuestro automatizador de tareas Grunt.

Ahora que sabemos qué es Grunt, veamos qué tareas podemos realizar con esta herramienta o qué necesitamos para ejecutarla y cómo es su funcionamiento.



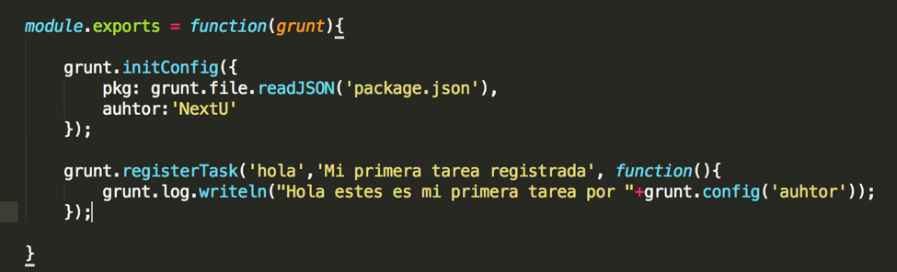
En esta imagen podemos observar el flujo o el manejo que Grunt le da a nuestros archivos.   
  
Por cada modificación de un archivo, si continua una nueva tarea, crea un archivo temporal y continúa con las modificaciones. Si no continua, termina su ciclo en la modificación, dejando el archivo ya transformado según la tarea que le hayamos asignado.  
  
Para que veamos ejemplificados lo que podemos realizar con Grunt, debemos tener la siguiente configuración para tener nuestro ambiente de automatización de tareas listo para ejecutar nuestras tareas con Grunt.   
  
Debemos instalar los paquetes necesarios, en este caso que es grunt debemos instalar de forma global el paquete grunt-cli con la siguiente línea de comando en nuestra consola principal.



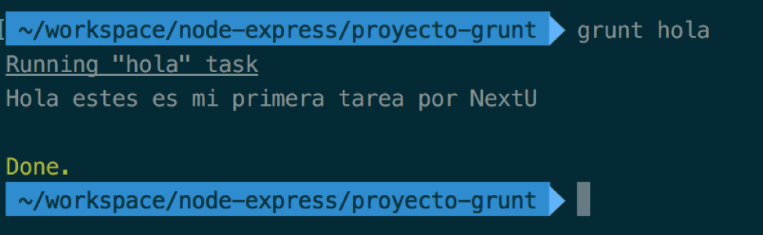
¡Tenga en cuenta que la instalación de **grunt-cli** no instala el corredor de tareas **Grunt**! El trabajo de la CLI de Grunt es simple: ejecuta la versión de Grunt que se ha instalado junto a un **Gruntfile**. Esto permite que múltiples versiones de **Grunt** se instalen simultáneamente en la misma máquina.  
  
luego de haber instalado esta librería, instalamos el paquete que correrá nuestras tareas previamente configuradas en el archivo Gruntfile.js, el paquete llamado Grunt es nuestra librería qué necesitamos para completar la funcionalidad ya que nos brinda muchas de las funcionalidades que se ejecutan el archivo de configuración y la instalación la realizamos de la siguiente manera



Ya con los paquetes instalados podemos comenzar a utilizar grunt en nuestro proyecto, seguido a esto, solo es comenzar con la configuración de nuestra herramienta en nuestro nuevo archivo **Gruntfile.js** y lo haremos de la siguiente manera



Luego de que tengamos una configuración básica, podemos entrar a ejecutar nuestra primera tarea y lo podremos hacer de la siguiente manera, debemos acceder por medio de la consola y ubicarnos en la raíz de nuestro proyecto y ejecutaremos nuestro comando de la siguiente manera y obtendremos el siguiente resultado.



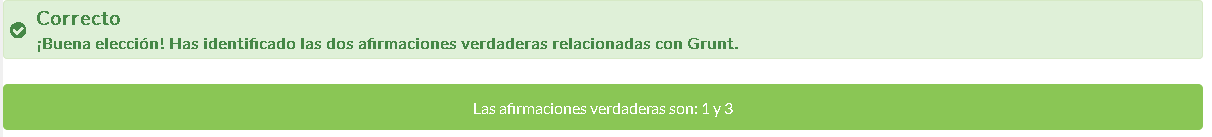
Así de simple podremos tener nuestro ambiente configurado con esta herramienta para automatizar tareas y poder realizar nuestras tareas en menos tiempo y utilizando las mejores prácticas al momento de programar.  
  
En **Grunt** encontramos las siguientes funciones que nos ofrece esta librería, para sacar mejor provecho de esta herramienta y son las siguientes.  
  
Encontramos dentro de nuestra librería la siguiente función nombrada como **initConfig(),** esta función explícitamente nos da a entender que es para inicializar nuestra configuración de grunt y poder configurar nuestras tareas que deseamos utilizar, también nos ofrece la librería de **Grunt** con el nombre de **resgiterTask()** y esta función nos ayuda a registrar nuestras tareas las cuales llamaremos por medio de la consola con la palabra grunt (+) más el nombre que asignemos a nuestra tarea, también nos provee de una función para poder ejecutar multitareas con el nombre de **registerMultiTask()** que veremos en un breve ejemplo más adelante estas son algunas de las funciones que nos provee **Grunt** para la realización de tareas automatizadas.

* 1. Actividad 1

Tiempo de pensar

Como ya lo sabes, Grunt es una herramienta que nos facilita el desarrollo de aplicaciones web pues optimiza nuestro tiempo mientras realizamos tareas complejas.  
  
Instrucciones:  
  
En esta actividad, deberás leer las afirmaciones y determinar cuáles de ellas son verdaderas. Cuando estés completamente seguro, selecciona la opción de respuesta adecuada.   
  
Afirmaciones:

1. Grunt es una librería o paquete de Nodejs.
2. Para poder implementar Grunt, debemos crear un archivo en la base de nuestro proyecto, con el nombre de: grunt.js
3. Para usar Grunt debemos contar con el archivo package.json en la base de nuestro proyecto, ya que este archivo nos indicará qué librerías de Nodejs tenemos o hemos instalado.
4. Por cada modificación de un archivo con Grunt, si continua una nueva tarea, crea un archivo fijo y continúa con las modificaciones.



* 1. Probando Grunt

Instalar:

npm install grunt-cli -g

npm install grunt --save-dev

Instalacion de minify

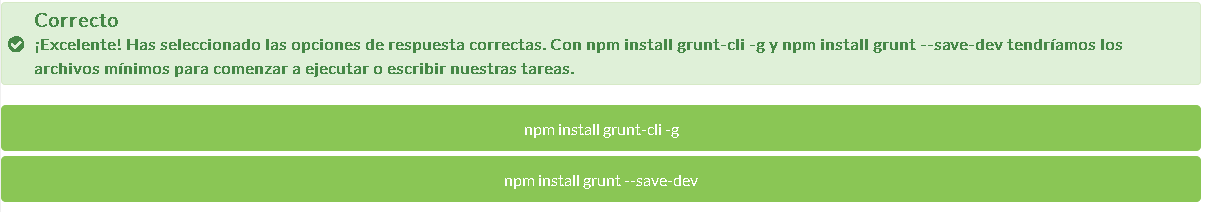
npm install grunt-contrib-uglify --save-dev

Para importar un plugin en Grunt

grunt.task.loadNpmTasks('grunt-contrib-uglify');

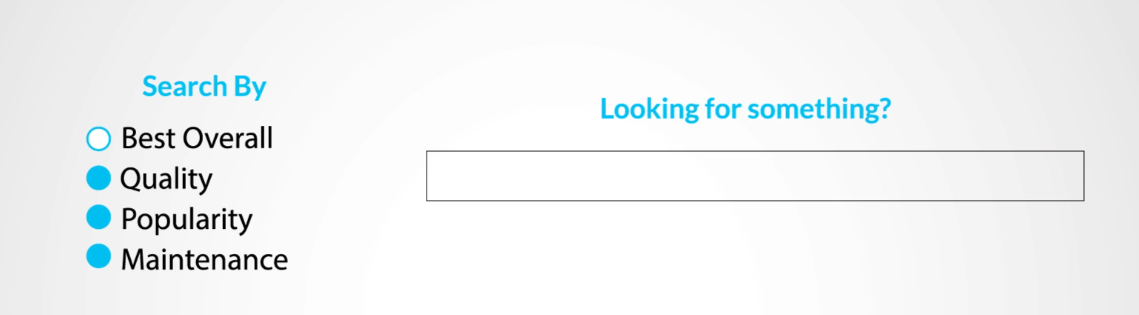
4.4. Actividad 2

Tiempo de pensar

Recuerda, Grunt es una herramienta que nos facilita el desarrollo de aplicaciones web pues optimiza nuestro tiempo mientras realizamos tareas complejas, por medio de una configuración básica en la que creamos un archivo en la base de nuestro proyecto, con el nombre de: gruntfile.js  
  
Instrucciones:  
Lee el enunciado y elige dos de las opciones de respuesta que mencionan los archivos que lo completan.  
  


1. Lección 4: NPM como herramienta para automatización de tareas
   1. [NPM como automatizador de tareas](https://learn.nextu.com/mod/lesson/view.php?id=3591&pid=P_WEBDEV_V2)

Se pueden ejecutar tareas de bash normales o plugins que instalemos



npm install stylus -g

npm install uglify-js --save-dev

npm install mkdirp -g 🡪 paquete para crear directorios usando npm

* 1. Actividad 1

Tiempo de pensar

Recuerda que existen muchas herramientas para la automatización de tareas pero la principal de todas es el mismo manejador de paquetes npm de nodejs.  
  
Instrucciones:

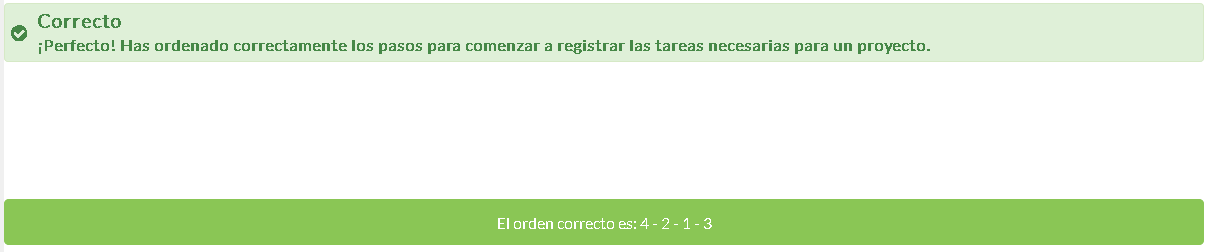
A continuación encontrarás, en desorden, algunos pasos para comenzar a registrar las tareas necesarias para un proyecto.

Deberás ordenarlos mentalmente, de manera lógica.

Cuando estés completamente seguro, selecciona la opción de respuesta que plantea el orden correcto.

Pasos:

1. Llenamos la información que nos solicita y encontramos una parte de nuestro archivo nuevo generado package.json, con el nombre scripts.
2. Iniciamos un nuevo proyecto nodejs con el comando npm init
3. Allí comenzaremos a registrar las tareas que sean necesarias para nuestro proyecto.
4. Creamos un proyecto con el nombre de proyecto-npm



* 1. Lectura: Usos comunes y buenas prácticas de NPM

Usos comunes y buenas prácticas de npm



Usos comunes:

Si necesitas llevar a cabo operaciones en tu paquete, antes de que sea usado y de manera que no dependa del sistema operativo o la arquitectura de la consola de destino, usa el script prepublish.  
  
Éste incluye tareas como:

* Compilar el código de origen de CoffeeScript en JavaScript.
* Crear versiones minificadas del código de origen de JavaScript.
* Atraer recursos remotos que tu paquete usará.

La ventaja de hacer todo esto durante el prepublish es que es de inmediato y en un mismo lugar, de esta forma puedes reducir la complejidad y la variabilidad. Esto significa que:

* Puedes depender de **coffee-script** como un **devDependency.** De esta manera tus usuarios no necesitarán tenerlo instalado.
* No necesitas incluir minificadores en tu paquete. Esto reduce el tamaño para tus usuarios.
* No necesitas apoyarte en que tus usuarios tengan curl, wget u otras herramientas del sistema.

Buenas prácticas:

* No salgas con un código de error no nulo, a no ser que así lo quieras. Excepto por los scripts que no están instalados, esto hará que la acción de npm falle y retroceda. Si la falla es mínima o solo va a prevenir algunas características opcionales, es mejor solo imprimir una advertencia y la salida satisfactoriamente.
* Intenta no usar scripts para que hagan lo que npm puede hacer por ti. Lee sobre **package.json** para que veas todas las cosas que puedes especificar y habilitar solo describiendo apropiadamente tu paquete. En general, esto te guiará a un estado más robusto y consistente.
* Inspecciona el ambiente para determinar dónde poner las cosas. Por ejemplo, si el ambiente variable de **npm\_config\_binroot** está configurado como **/home/user/bin,** entonces no trates de instalar ejecutables en **/usr/local/bin,** ya que el usuario probablemente lo configuró así por una razón.
* No le pongas prefijos a tus comandos de scripts con “sudo”. Si los permisos de raíz son requeridos por alguna razón, entonces fallará con ese error y el usuario deberá ser sudo para ejecutar el comando npm.
* No uses **instalar,** usa el archivo **.gyp** para compilación y usa prepublish para todo lo demás. Casi nunca deberías configurar explícitamente una preinstalación o una instalación de script. Si estás haciendo esto, considera si tienes otra opción, ya que el único uso válido de la instalación o preinstalación de scripts es para la compilación, lo cual debe hacerse en la arquitectura.

5.4. Actividad 2

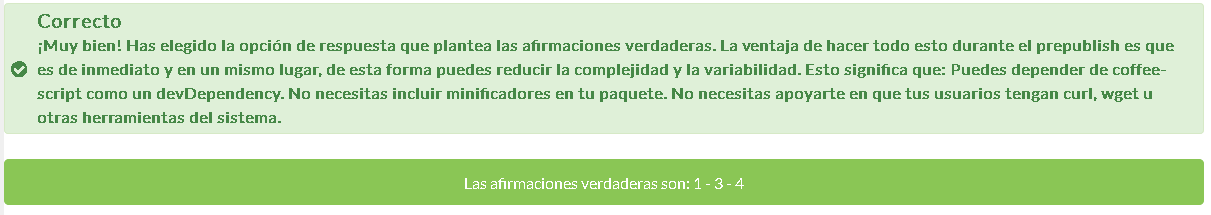
Tiempo de pensar

Ya lo sabes, si necesitas llevar a cabo operaciones en tu paquete, antes de que sea usado y de manera que no dependa del sistema operativo o la arquitectura de la consola de destino, usa el script prepublish.  
  
Instrucciones:

* Lee las afirmaciones que planteamos a continuación relacionadas con la ventaja de hacer prepublish.
* Determina cuáles de ellas son verdaderas.
* Selecciona la opción de respuesta correcta.

Afirmaciones:

1. Puedes depender de coffee-script como un devDependency. De esta manera tus usuarios no necesitarán tenerlo instalado.
2. Reduce la complejidad, pero no la variabilidad. 🡪 Falso
3. No necesitas incluir minificadores en tu paquete. Esto reduce el tamaño para tus usuarios.
4. No necesitas apoyarte en que tus usuarios tengan curl, wget u otras herramientas del sistema.



1. UNIDAD 1: Prueba

|  |  |
| --- | --- |
| **Comenzado en** | Friday, 20 de November de 2020, 12:08 |
| **Estado** | Terminados |
| **Finalizado en** | Friday, 20 de November de 2020, 12:16 |
| **Tiempo empleado** | 7 mins 29 segundos |
| **Puntos** | 10/10 |
| **Calificación** | **100** de un total de 100 |
| **Comentario de retroalimentación** | Buen trabajo! |

Top of Form

### Pregunta 1

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

La instalación de grunt-cli no instala el corredor de tareas Grunt.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

El trabajo de la CLI de Grunt es ejecutar la versión de Grunt que se ha instalado junto a un Gruntfile, esto permite que múltiples versiones de Grunt se instalen simultáneamente en la misma máquina.

La respuesta correcta es: Verdadero

### Pregunta 2

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Cuál de las siguientes opciones es la función de Gulp que se describe en el enunciado?

Esta función nos permite disparar una función, tan pronto el blob que le hemos pasado en el primer parámetro haya sido editado, eliminado, renombrado o agregado.

Seleccione una:

a. src

b. dest

c. task

d. watch

#### Retroalimentación

Esta función es muy útil ya que nos ayuda a automatizar, ya sea una tarea o un conjunto de tareas.

La respuesta correcta es: watch

### Pregunta 3

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Cuál de las siguientes opciones es la función de Grunt que se describe en el enunciado?

Esta función ayuda a registrar las tareas, las cuales llamamos por medio de la consola con la palabra grunt más el nombre que le asignemos a la tarea.

Seleccione una:

a. initConfig()

b. resgiterTask()

c. registerMultiTask()

d. watch()

#### Retroalimentación

En Grunt encontramos varias funciones que nos permitirán sacarle mejor provecho a esta herramienta.

La respuesta correcta es: resgiterTask()

### Pregunta 4

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

De las siguientes opciones, elige aquellas que contienen características de Grunt.

Seleccione una o más de una:

a. Debemos contar con el archivo package.json en la base de nuestro proyecto, ya que este archivo nos indicará qué librerías de Nodejs tenemos o hemos instalado.

b. Grunt hace que las tareas sean exitosas aunque nos tome un poco más de tiempo.

c. En la configuración básica debemos crear un archivo en la base de nuestro proyecto, con el nombre de: gruntfile.js

d. Debemos tener instalado Nodejs en cada uno de los computadores en los que realizaremos pruebas o desarrollaremos nuestro proyecto.

#### Retroalimentación

Grunt es una herramienta que nos facilita el desarrollo de aplicaciones web pues optimiza nuestro tiempo mientras realizamos tareas complejas.

Las respuestas correctas son: Debemos contar con el archivo package.json en la base de nuestro proyecto, ya que este archivo nos indicará qué librerías de Nodejs tenemos o hemos instalado., En la configuración básica debemos crear un archivo en la base de nuestro proyecto, con el nombre de: gruntfile.js, Debemos tener instalado Nodejs en cada uno de los computadores en los que realizaremos pruebas o desarrollaremos nuestro proyecto.

### Pregunta 5

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

npm run test es un comando que nos permite iniciar un nuevo proyecto node.js

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

Recuerda que para iniciar un nuevo proyecto debes usar el comando npm init.

La respuesta correcta es: Falso

### Pregunta 6

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

Después de ejecutar nuestra primera tarea, procedemos a automatizarla con la función watch de gulp, que es la encargada de indicar si un archivo se ha cambiado, agregado o borrado y realizar cambios a alguna operación.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

Recuerda que debemos cambiar nuestro archivo main.js para que por sí solo ejecute la tarea que hemos creado.

La respuesta correcta es: Verdadero

### Pregunta 7

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

Existen muchas herramientas para la automatización de tareas pero la principal de todas es el mismo manejador de paquetes npm de nodejs.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

Con éste podremos ejecutar tanto tareas de bash normales, como también tareas de plugins o paquetes que instalemos.

La respuesta correcta es: Verdadero

### Pregunta 8

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Selecciona del listado las opciones que contienen buenas prácticas de npm.

Seleccione una o más de una:

a. No salgas con un código de error no nulo, a no ser que así lo quieras. Excepto por los scripts que no están instalados, esto hará que la acción de npm falle y retroceda.

b. Intenta no usar scripts para que hagan lo que npm puede hacer por ti.

c. No le pongas prefijos a tus comandos de scripts con "sudo".

d. Usa instalar para la compilación y para todo lo demás.

#### Retroalimentación

Es importante conocer muy bien las buenas prácticas para usar npm, pues te evitarán dolores de cabeza y errores innecesarios.

Las respuestas correctas son: No salgas con un código de error no nulo, a no ser que así lo quieras. Excepto por los scripts que no están instalados, esto hará que la acción de npm falle y retroceda., Intenta no usar scripts para que hagan lo que npm puede hacer por ti., No le pongas prefijos a tus comandos de scripts con "sudo".

### Pregunta 9

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

El script prepublish nos permite llevar a cabo operaciones en tu paquete, antes de que sea usado y de manera que no dependa del sistema operativo o la arquitectura de la consola de destino.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

Todo lo que hagas durante el prepublish es inmediato y está en un mismo lugar, así puedes reducir la complejidad y la variabilidad.

La respuesta correcta es: Verdadero

### Pregunta 10

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Determina si la siguiente afirmación es falsa o verdadera:

Gulp es una herramienta en forma de script, escrita en Nodejs, que ayuda a automatizar tareas comunes en el desarrollo de aplicaciones.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

Esta herramienta fue creada para facilitar el proceso de desarrollo, gastando menos tiempo en el proceso de escribir código, realizar validaciones y pruebas de nuestro código.

La respuesta correcta es: Verdadero

Bottom of Form