Contenido

[1. Autores del trabajo, planificación y entrega 2](#_Toc448254544)

[1.1 Autores 2](#_Toc448254545)

[1.2 Planificación 2](#_Toc448254546)

[1.3 Entrega 2](#_Toc448254547)

[2. Requisitos del prototipo a implementar 3](#_Toc448254548)

[2.1 Requisitos funcionales 3](#_Toc448254549)

[2.2 Otros requisitos 3](#_Toc448254550)

[3. Criterios de comparación en la implementación 4](#_Toc448254551)

[3.1 Criterio 1: Nombre del criterio 4](#_Toc448254552)

[3.2 Criterio 2: Nombre del criterio 4](#_Toc448254553)

[3.N Criterio N: Nombre del criterio 4](#_Toc448254554)

[4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A 5](#_Toc448254555)

[4.1 Documentación de diseño 5](#_Toc448254556)

[4.2 Documentación de construcción 5](#_Toc448254557)

[4.3 Documentación de pruebas 5](#_Toc448254558)

[4.4 Documentación de instalación 5](#_Toc448254559)

[4.5 Manual de usuario 5](#_Toc448254560)

[5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B 6](#_Toc448254561)

[5.1 Documentación de diseño 6](#_Toc448254562)

[5.2 Documentación de construcción 6](#_Toc448254563)

[5.3 Documentación de pruebas 6](#_Toc448254564)

[5.4 Documentación de instalación 6](#_Toc448254565)

[5.5 Manual de usuario 6](#_Toc448254566)

[6. Comparación de las dos implementaciones 7](#_Toc448254567)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 7](#_Toc448254568)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 7](#_Toc448254569)

[7. Comparación de la implementación de las tecnologías 8](#_Toc448254570)

[8. Conclusiones 10](#_Toc448254571)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Grupo Tarde 2

Sergio Esteban Adán

Rebeca Muraru

Carlos Fuentes Martínez

Alejandro Freire Rodríguez

## 1.2 Planificación

<https://app.ganttpro.com/shared/token/60d647c77b5376b94ea29f744c484d1db34acc610b0069669a117feb7523e55c>

## 1.3 Entrega

<http://github.com/Kratemis/TG3>

En este apartado debe incluirse un enlace (URL) a un repositorio en GitHub o en BitBucket creado para el trabajo.

En dicho repositorio debe encontrarse, al menos los siguientes archivos en la rama máster:

* Informe del trabajo: con el nombre TG3\_final.docx
* Presentación del trabajo: TG3\_final.pptx
* Prototipos obtenidos implementando cada una de las tecnologías (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
  + PrototipoTecnologiaA\_final.zip (o .rar)
  + PrototipoTecnologiaB\_final.zip (o .rar).

Dichos archivos serán los que se tendrán en cuenta para la calificación del trabajo.

**2. Requisitos del prototipo a implementar**

**2.1 Requisitos funcionales**

|  |  |
| --- | --- |
| **REQ.** | **DESCRIPCIÓN** |
| RF01 | Añadir artículo |
| RF02 | Añadir stock |
| RF03 | Vender/Consumir stock |
|  |  |

**2.2 Otros requisitos**

|  |  |
| --- | --- |
| **REQ.** | **DESCRIPCIÓN** |
| R01 | Separación de vistas |
| R02 | Interfaz intuitiva |

**3. Criterios de comparación en la implementación**

**3.1 Criterio 1: Tiempo de instalación**

Nombre del criterio: Tiempo de instalación de ambos sistemas.

Descripción: Minutos y segundos exactos que dura la instalación en una máquina de similares características.

Tipo de valor: Numérico (segundos).

**3.2 Criterio 2: Base de Datos**

Nombre del criterio: Facilidad de aprendizaje de base de datos

Descripción: Horas/minutos necesarias para el aprendizaje necesario para realizar el despliegue de la base de datos de la aplicación.

Tipo de valor: Numérico (Horas/minutos).

**3.3 Criterio 3: Líneas de código**

Nombre del criterio: Líneas de código

Descripción: Líneas de código usando programación optimizada que tiene la aplicación en ambos sistemas

Tipo de valor: Numérico (unidades).

**3.4 Criterio 4: Facilidad de aprendizaje**

Nombre del criterio: Facilidad de aprendizaje

Descripción: Horas/minutos necesarias para el aprendizaje necesario para realizar el despliegue de la aplicación

Tipo de valor: Numérico (Horas/minutos).

# 4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología Django

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología A, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## 4.1 Documentación de diseño

Nuestro diseño del prototipo de cafés consta de una página principal con una barra de navegación en la cual tenemos:

* Listado cafés

- Tipos de café los muestra (nombre, cantidad, descripción, foto)

- Botón para consumir unidades.

* Añadir café

- Formulario para meter cafés en la base de datos.

(nombre foto stock descripción botón añadir)

* Stock

- Tipos de café los muestra (nombre, cantidad, descripción)

- Dos botones añadir 1 o añadir 16 que es una caja.

A continuación, realizamos un diagrama de interfaz para ver el procedimiento de nuestro prototipo además de un diagrama BPMN que se trata de un diagrama der procesos:

Diagrama de interfaz

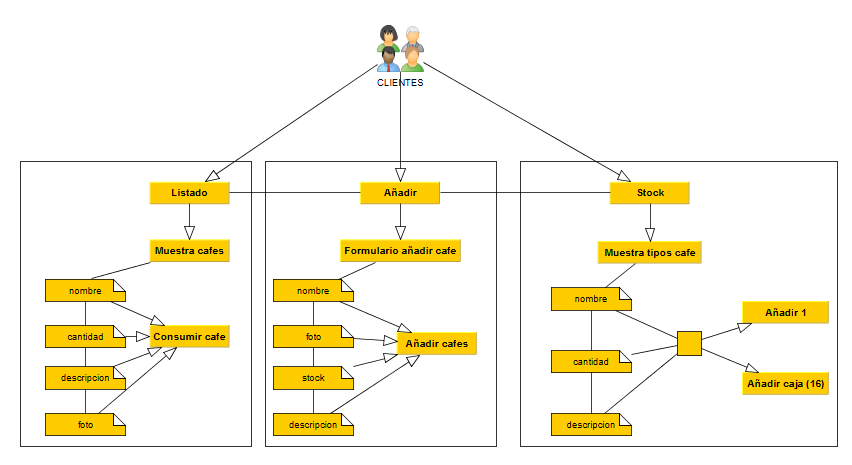
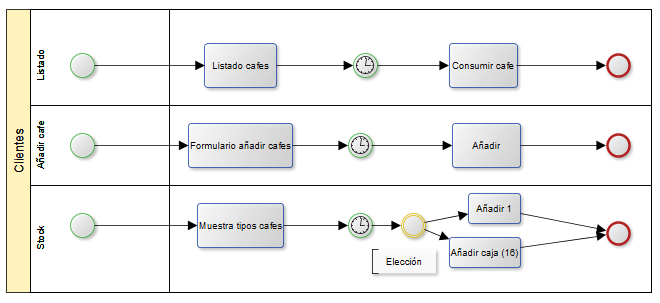


Diagrama BPMN



El diagrama anterior se basa en los procesos que tenemos que realizar para las tres acciones principales que son Stock, Añadir café y Listado.

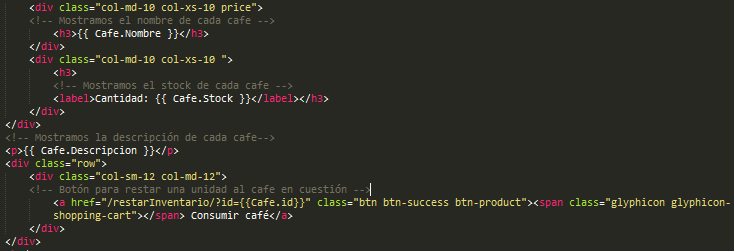
## 4.2 Documentación de construcción

El prototipo realizado en Django, al ser un modelo vista controlador, separa los archivos HTML de los archivos Python (controladores).

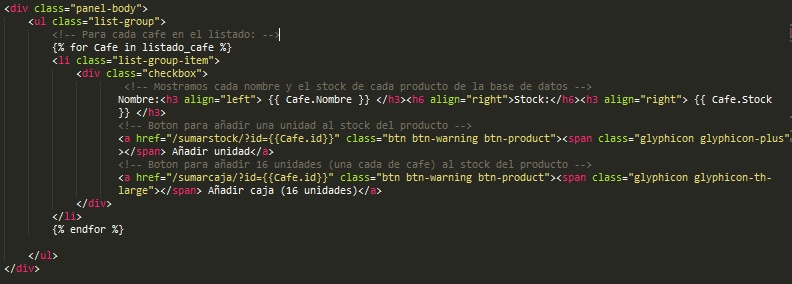
Por esta razón, podemos encontrar tres vistas de la aplicación en la carpeta *templates:*

Stock.html, inventario.html y listado.html. Si obviamos el NavBar o barra de navegación de las vistas su código es muy sencillo.

**Listado.html**

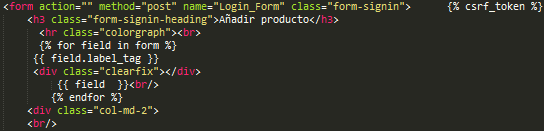


**Stock.html.**



**Inventario.html**

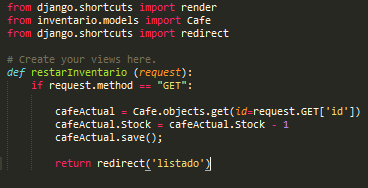
En la página de inventario tenemos un formulario simple para añadir un nuevo producto a nuestra aplicación de inventario.



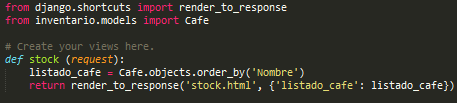
En la parte de los controladores:

**Views.py (Listado)**

Listado tiene un controlador para llamar a la vista listado.html (Similar a la del siguiente apartado. Además tiene una controladora para restar inventario (una unidad).

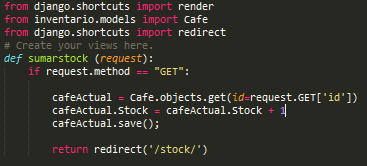


**Views.py (Stock)**



Este controlador llama a la vista stock.html pasándole el listado\_cafe por parámetro.

Además, stock tiene dos controladoras, para sumar una unidad y una caja:



La controladora que suma una caja, lógicamente sumará 16 (cantidad de unidades que vienen en una caja) en lugar de 1.

**Views.py (Inventario)**

Inventario es la vista en la que aparece el formulario para añadir café. Su controladora es:



**4.3 Documentación de pruebas**

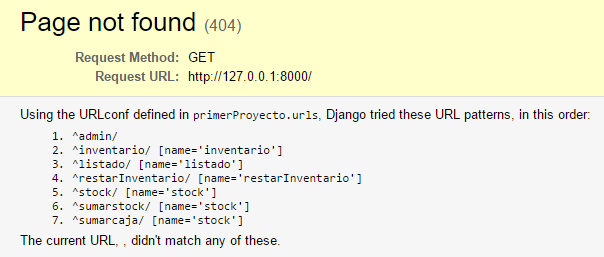
Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba

Durante los casos de prueba, el equipo encargado de esta fase, encontró determinadas deficiencias en el prototipo, en especial en el manejo de excepciones como por ejemplo consumir cafes sin existir stock.



Quedando la cantidad en negativo.

Otro error del prototipo puede encontrarse al manejar la URL directamente, donde si por ejemplo, borramos una parte de ella para intentar ir a la raíz de la página web, nos devuelve un error, mostrando:



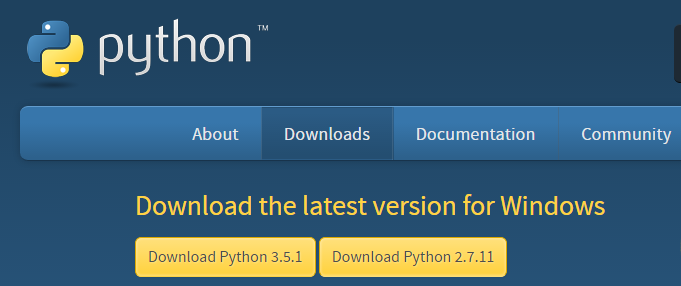
Al tratarse de un prototipo, la aplicación se ha desarrollado mínimamente, demostrando el manejo de Django pero en ningún caso se ha tratado de desarrollar una versión final. Debido a esta razón, se encuentran fallos tan críticos en la aplicación.

## 4.4 Documentación de instalación

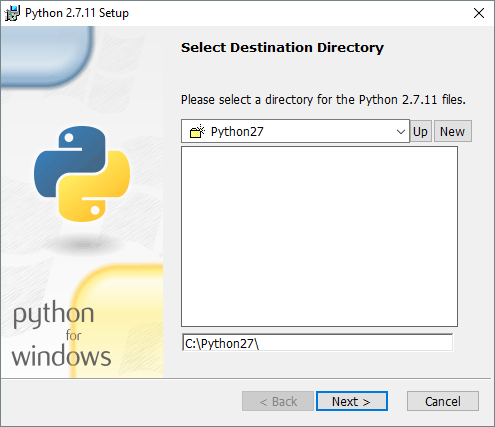
Como ya sabemos Django utiliza el lenguaje de programación Python por esta razón antes de todo debemos instalar este lenguaje.

* Instalación de Python

El primer paso es acceder a la página web de Python (<https://www.python.org/downloads/> ) acceder a la pestaña de descargas y pulsar en ella.



Tras este paso nos aparecerá en nuestra bandeja de descargas un ejecutable, pinchamos sobre él y nos saldrá la siguiente ventana:



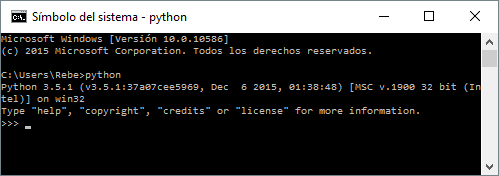
Como se puede observar lo guardamos en C:\ en nuestro disco principal.

Una vez finalizada la instalación pulsamos en aceptar y ya podemos empezar a trabajar con Python.

Para trabajar con Python podemos usar cualquier editor de texto como por ejemplo:

* Gedit
* Sublime text
* Atom

Nuestros primeros pasos en Python son abriendo la ventana de comandos. Usando el comando ***Python*** abrimos la ventana de comandos de este:



Una vez dentro ya podemos comenzar a programar.

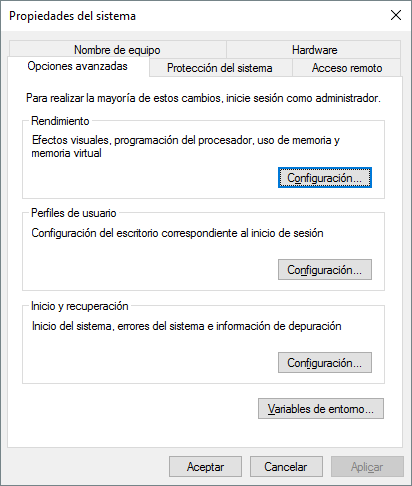
Con el lenguaje Python instalada procedemos a instalar Django.

Para la instalación de Django nos vamos a la página de este (<https://www.djangoproject.com/download/> ) e instalamos la versión django-1.6.

Nos descargaremos este archivo ***Django-1.6.tar.gz*** y lo descomprimimos en el mismo directorio que instalamos Python.

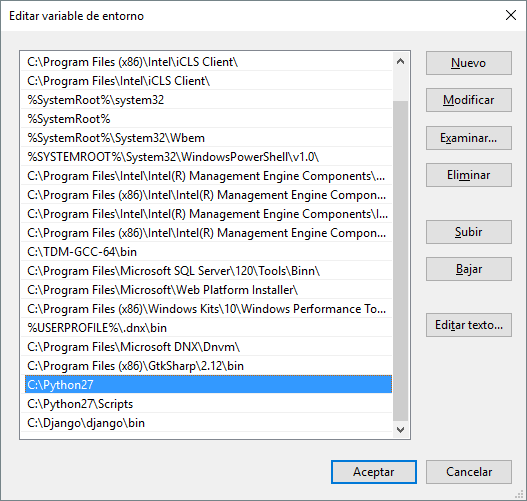
Tras este paso crearemos las variables de entorno necesarias para poder acceder desde cualquier lugar de nuestro Windows a los ejecutables que necesitemos. Para ello debemos de seguir los siguientes pasos:

* Ir a panel de control
* Sistema
* Configuración avanzada del sistema



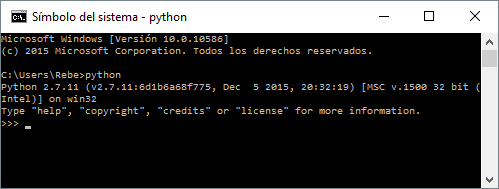
Una vez aquí pulsamos en variables de entorno y buscamos ***path*** le damos a editar y añadimos las siguientes:

* **C:\Python27**
* **C:\Python27\Scripts**
* **C:\Django\django\bin**



De esta manera ya podemos acceder de forma directa a Python desde la consola.

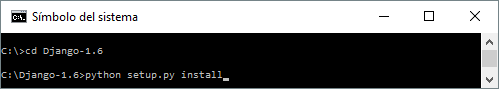
El siguiente para ver su funcionamiento abrimos la consola y ejecutamos ***python***



Como se puede observar en la imagen está instalado perfectamente.

Para comenzar a trabajar con el framework Django debemos abrir la consola y acceder a la carpeta en donde descomprimimos Django.

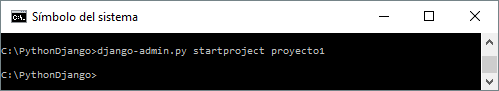
A continuación debemos ejecutar el siguiente comando ***Python setup.py install***



Tras ejecutar ese comando nos saldrán en la consola datos instalándose que veremos, tardara unos segundos. Tras esto ya tenemos listo Django.

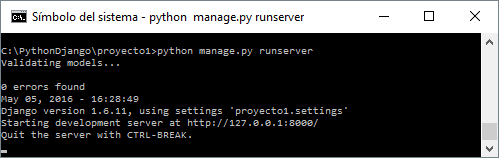
Para comenzar nuestro primer proyecto a través de la ventana de comandos nos ubicamos en la carpeta en donde queramos crearlo y una vez allí ejecutamos el siguiente comando:

* ***Django-admin.py startproject proyecto1***

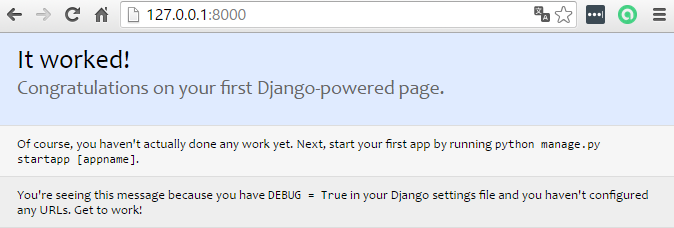


Para poder ver nuestro proyecto1 en la web el siguiente paso es ejecutar el siguiente comando en la consola dentro de la carpeta ***proyecto1***:

* ***python manage.py runserver***



A continuación abrimos nuestro navegador con la ruta que nos ha dado [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/)y observamos nuestra página.



## 4.5 Manual de usuario

Para iniciar la aplicación, tendremos que abrir un terminal de Windows (Botón Windows + R, escribir CMD y pulsar Intro). Una vez en el terminal, nos dirigiremos a la carpeta donde tengamos la aplicación (en mi caso en la raíz del disco C:\primerProyecto), por ejemplo cd c:\primerProyecto. Una vez que estemos posicionados en nuestra carpeta, escribiremos python manage.py runserver.

Ahora podemos abrir un navegador cualquiera y nos dirigiremos a la siguiente URL: 127.0.0.1:8000/listado y se nos abrirá la página principal de nuestra aplicación.

Para el manejo fácil de la aplicación se ha utilizado una barra de navegación que es visible desde cualquier página:



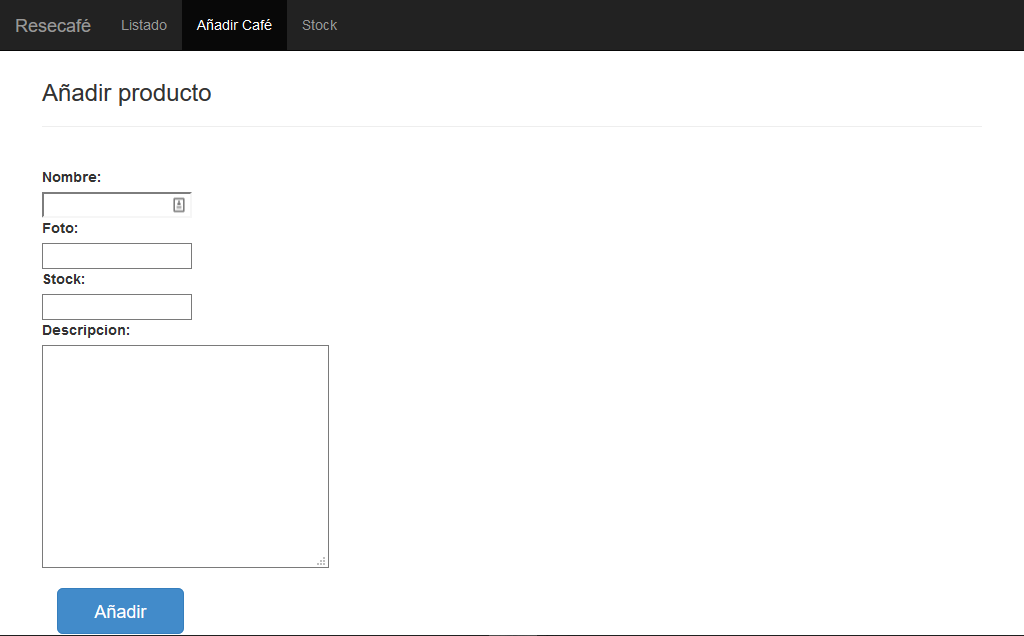
Desde esta barra se puede acceder a cualquiera de las páginas. En primer lugar, comenzaremos por Listado.

En esta página se muestran los cafés disponibles para consumir:



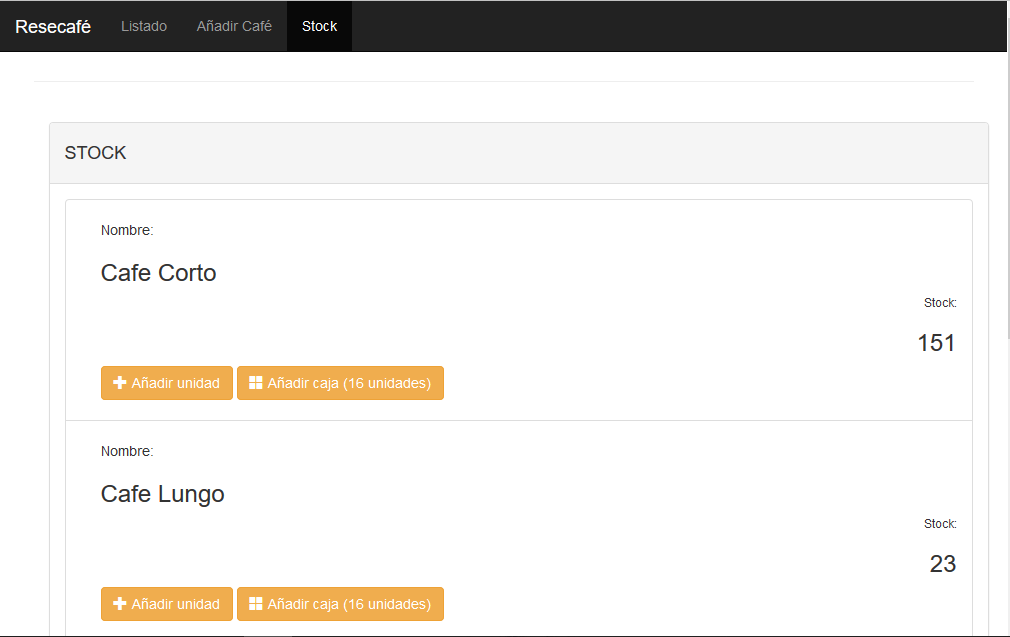
Si pulsamos en el botón verde, se restará uno al inventario, simulando una consumición del café.

Si ahora pulsamos en Añadir café, en la barra de navegación, nos dirigiremos a la página que contiene el formulario para añadir un nuevo tipo de café:



En esta página añadimos un nombre, el link de una foto online, la cantidad de café que tenemos y una pequeña descripción. Al pulsar en Añadir, se insertará en la base de datos y aparecerá en Listado.

Si por el contrario, nos ha llegado nuevo stock de un café que ya teníamos disponible en la tienda, utilizaremos la pestaña Stock:



Desde esta pestaña podremos añadir una unidad de café (si nos hemos equivocado y hemos pulsado demasiadas veces en Consumir café en la página Listado) o podremos añadir una caja de cafés, que tiene 16 unidades, si es que hemos comprado nuevo stock.

# 5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología Nodejs

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

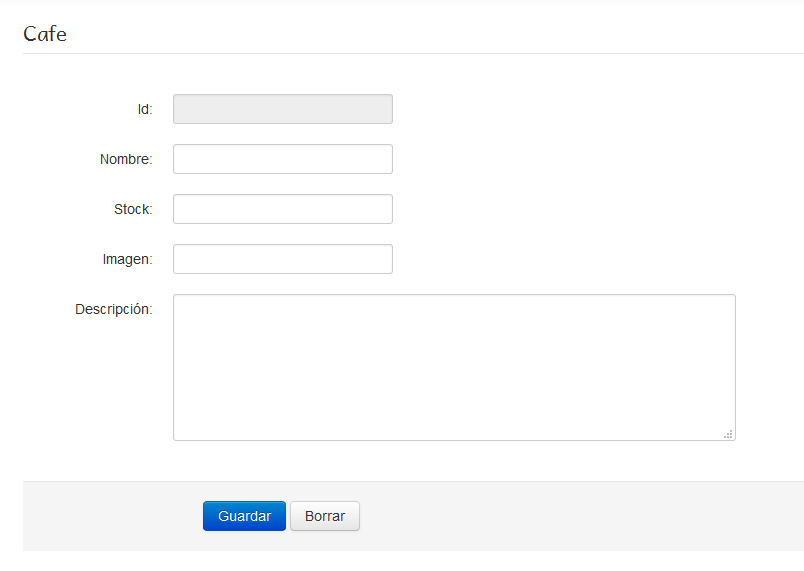
## 5.1 Documentación de diseño

Hemos desarrollado una aplicación muy sencilla. Con un par de opciones. La aplicación está diseñada para gestionar los cafés que se consumen, por lo que hay un par de opciones . Una interfaz simple, con tres actividades :Añadir Café, Consumir y ver ventas.

La primera pantalla que nos encontramos es la siguiente.



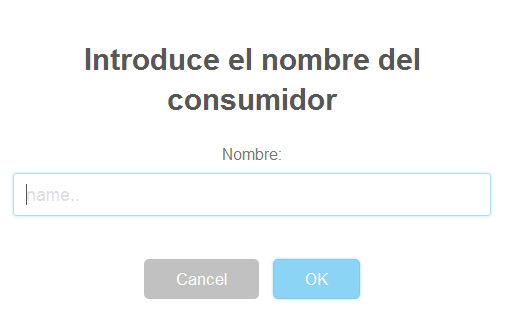
Si pulsamos Añadir Café,hemos desarrollado un formulario con los campos Nombre, Stock, imagen y descripción. Con dos botones; guardar y borrar.



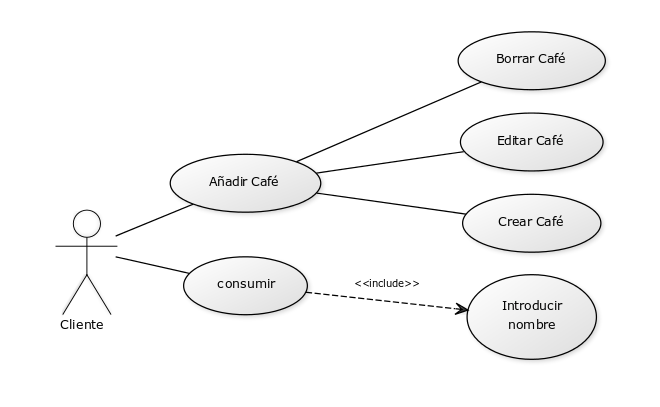
Un formulario en el cual podremos guardar el café o borrar uno ya existente.

Cuando pulsemos en Consumir tenemos la siguiente interfaz:

Un cuadro de texto que almacenará el nombre del usuario que consume.



Como vemos, son utilidades bastante intuitivas. A continuación, mostramos un diagrama de casos de uso, mostrando que actividades puede realizar un cliente en nuestra aplicación.



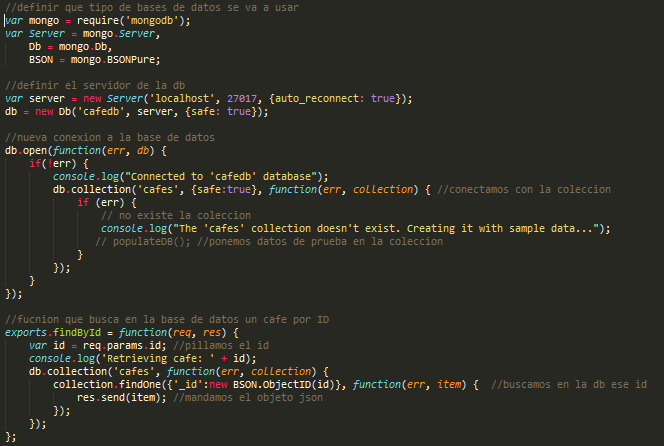
## 5.2 Documentación de construcción

Se ha implementado Node.js utilizando Express, Bootstrap y MongoDB.

Es por esto por lo que el proyecto cuenta con distintos archivos:

* **Routes/cafes.js:**

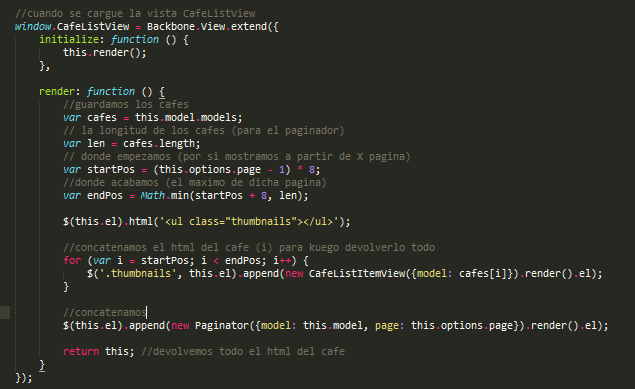
Módulo js encargado de lo relacionado con la base de datos Mongo, actualizar datos, borrarlos, enviarlos, etc.



* **Public/js/views:**

En js/views se encuentran los archivos js dedicados al manejo de las vistas. Aquí podemos ver el controlador que se ocupa de mostrar la lista de cafés al inicio.

* + **cafelist.js:**



* **Public/js/main.js**

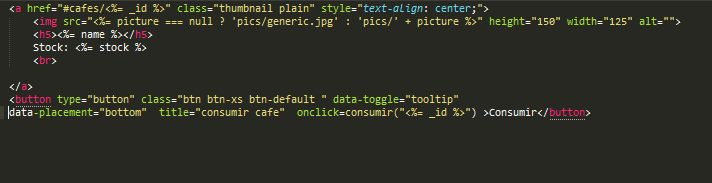
Este código es el encargado de saber que vista cargar según se indique en el navegador.



* **Public/tpl**

Aquí se pueden encontrar el código html de las vistas (About, lista de cafés, vista individual de un café y navbar).

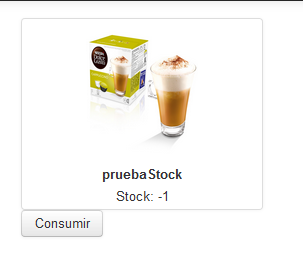
* + **CafeListItemView.html:**



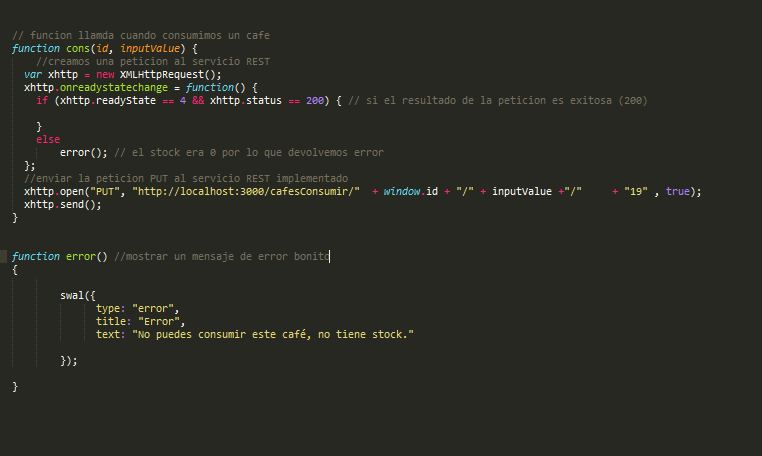
## 5.3 Documentación de pruebas

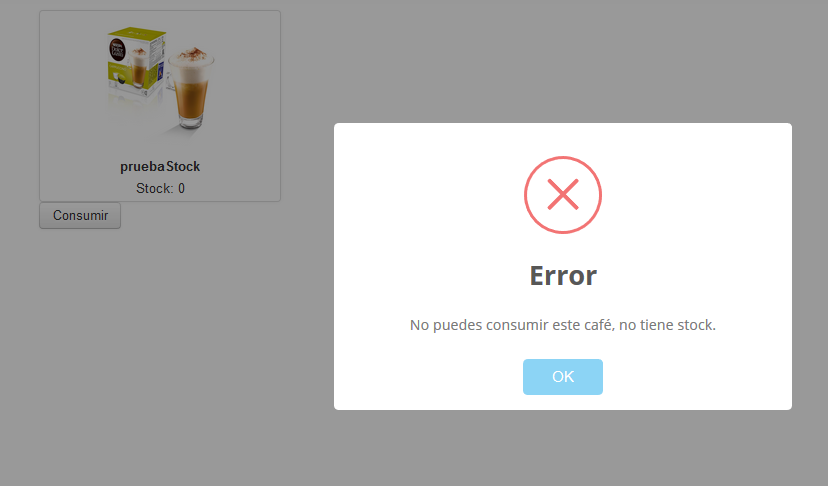
* Corrección del stock:

Cuando el stock llegaba a 0, la aplicación seguía restando:



Para corregirlo se implementó un mensaje de error cuando el stock es 0:





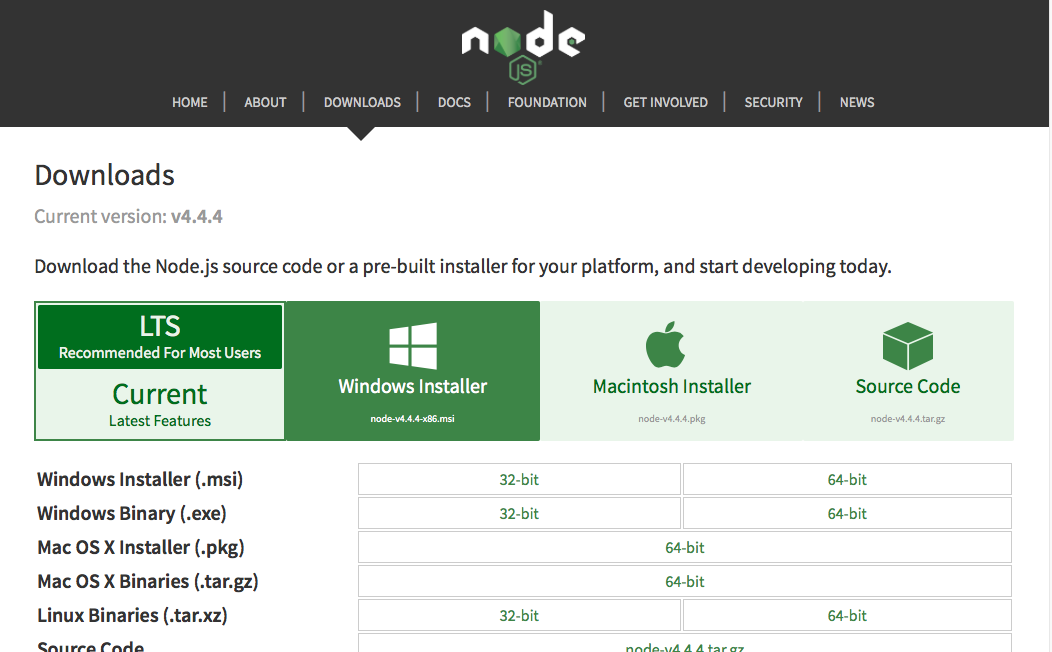
* El stock contiene letras:

Este error se da cuando al crear un café añades letras, por ejemplo: “cinco”.

Nuestro prototipo actualmente no tiene implementado ninguna solución a este problema.

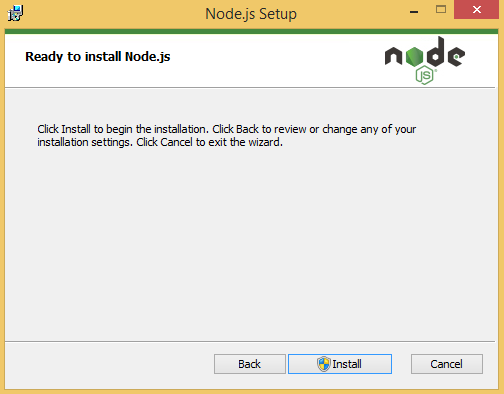
## 5.4 Documentación de instalación

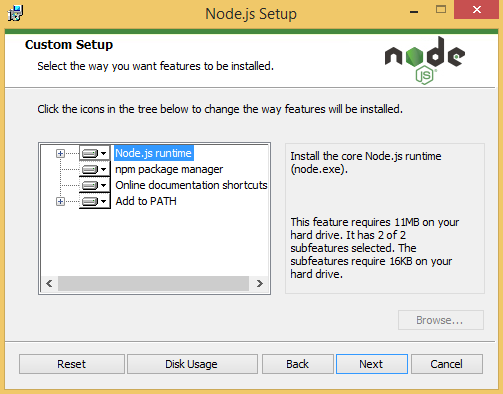
Accedemos a la web <https://nodejs.org/en/download/> y seleccionamos la plataforma Windows.

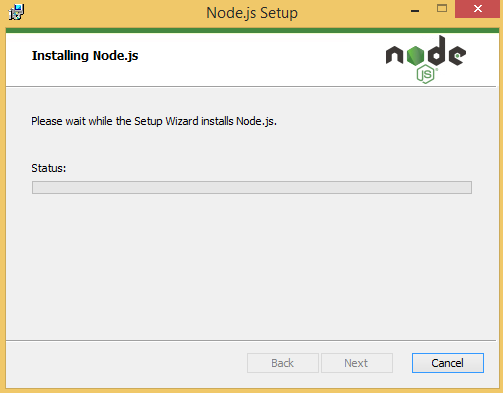
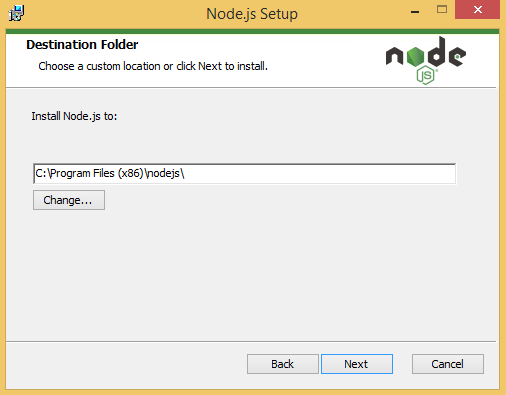
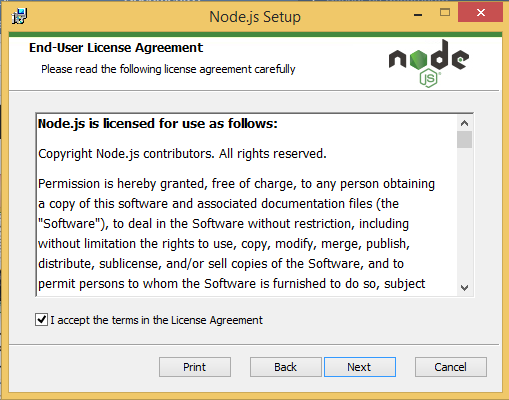


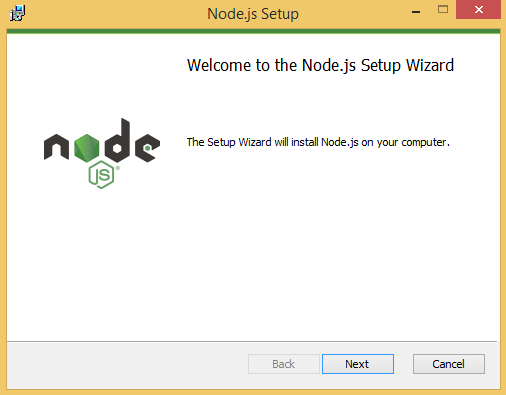
Al pulsar sobre Windows Installer te descargará el instalador para este sistema, un archivo con extensión "msi" que te mostrará el típico asistente de instalación de software.

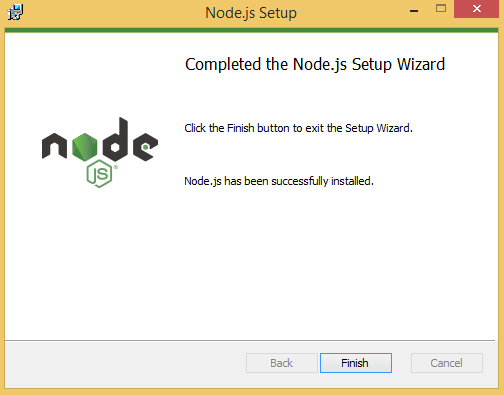
Una vez descargado, ejecutas el instalador y ¡ya lo tienes!

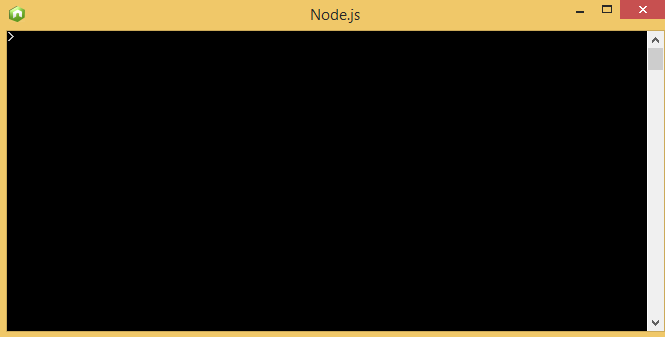
A partir de ahora, para ejecutar "Node" tienes que irte a la línea de comandos de Windows e introducir el comando "node".



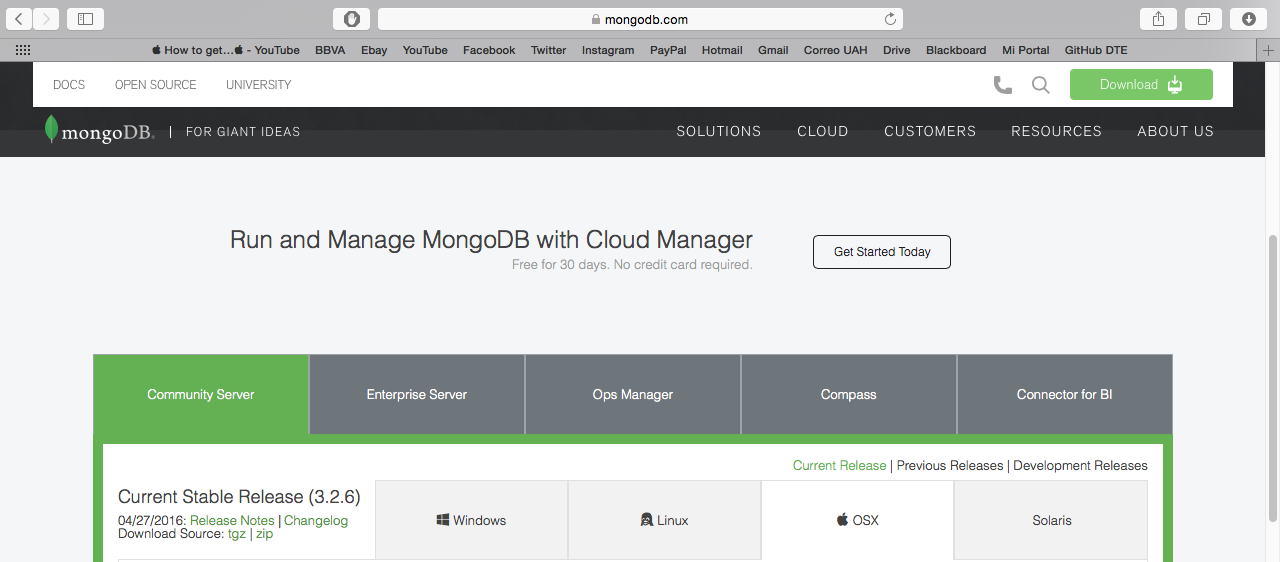
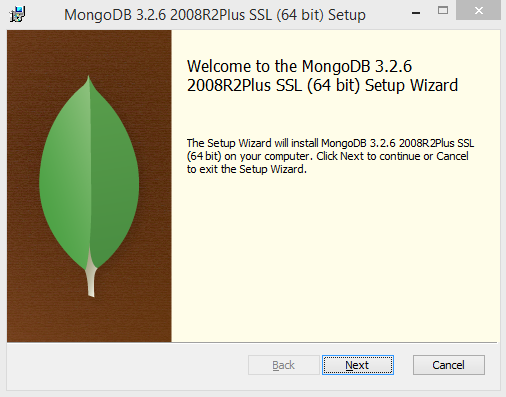


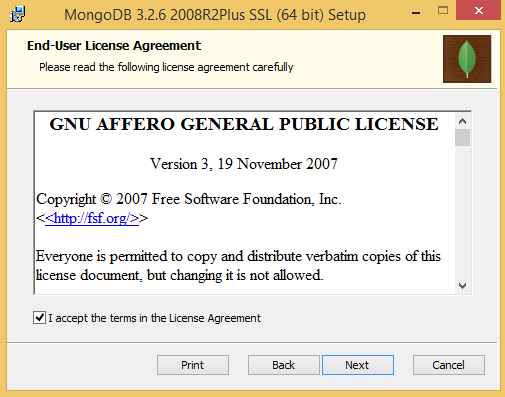
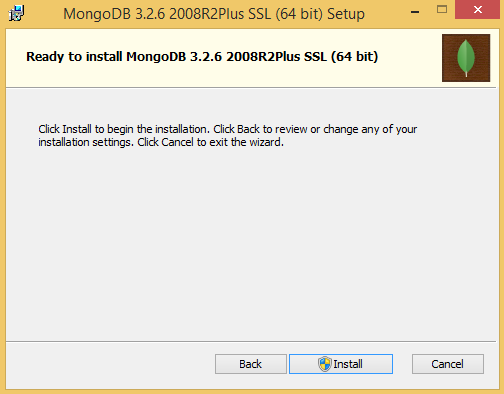
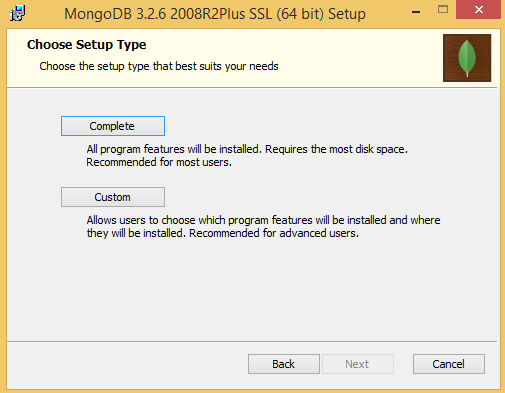


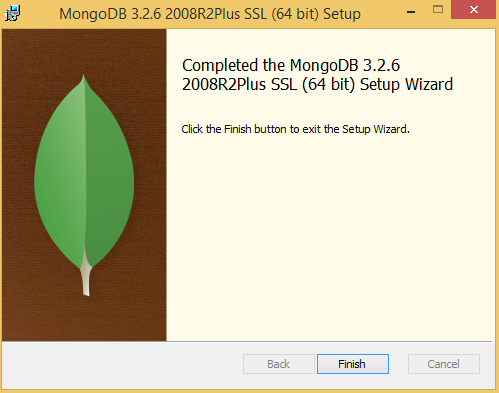




También es necesario instalar un sistema gestor de base de datos como Mongodb:

****Para instalar el gestor de base de datos MongoDB, vamos a su web, y al apartado descargas. Allí seleccionamos Windows, y se nos descargará el instalador. Se abre un asitente que nos guará en la instalación.

****

****

**Para iniciar el sevidor Mongo, accedemos a la siguiente ruta:**

C:\Program Files\MongoDB\Server\3.2\bin\

Ejecutar: mongod

## 5.5 Manual de usuario

Para arrancar la aplicación, abrimos el MongoDB y el nodejs en su carpeta de origen:

C:\Program Files\MongoDB\Server\3.2\bin\

mongod

C:\Users\Usuario\Downloads\Nodejs2>node server

Y en el buscador ponemos el siguiente link:

http://localhost:3000/#

La interfaz es bastante sencilla. Nada más entrar a la aplicación tenemos esta vista.

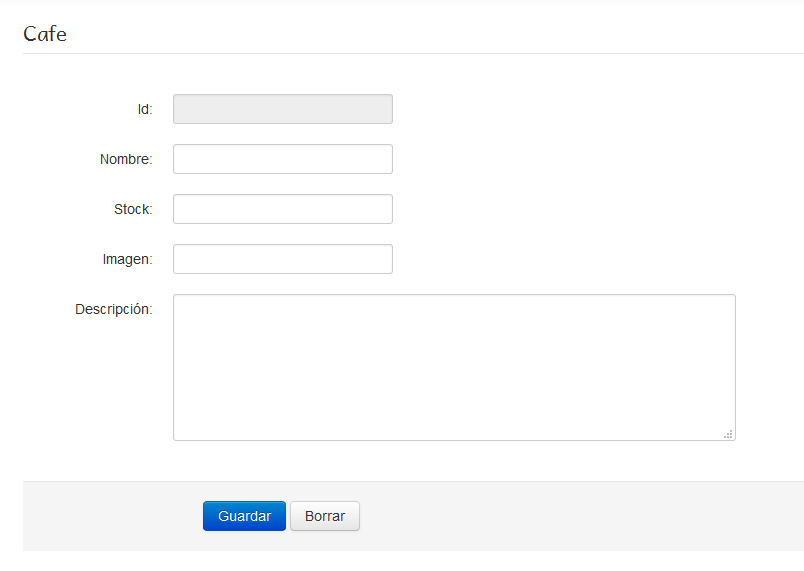


Desde arriba hasta abajo, tenemos:

-Cafés Delegación:

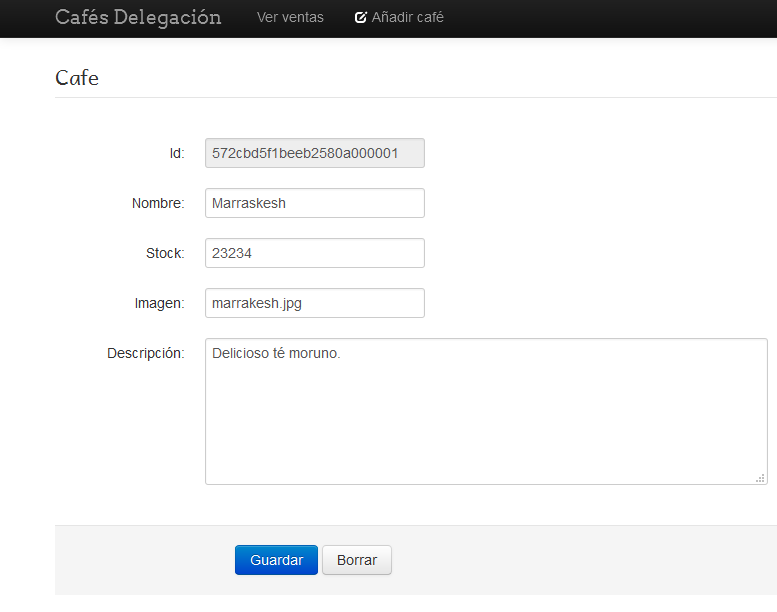
Si pulsamos aquí nos dirigimos a la página principal de la aplicación.

-Añadir café: Si pulsamos aquí, Se nos desplegará un formulario para añadir cafés al inventario.



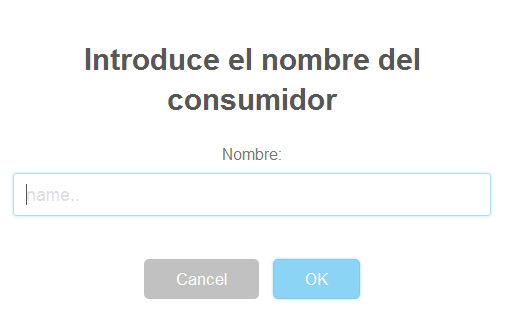
Rellenamos el nombre, el número de unidades que hay en stock, cargamos una imagen y una descripción. Y pulsamos en Guardar

Una vez creado un café, accedemos a la pestaña Cafés delegación, y si queremos borrarlo, solo tenemos que pulsar en la foto, y nos dirigirá a los detalles. Y solo tenemos que pulsar en Borrar.



- Consumir:

Pulsamos en Consumir, y nos aparece un formulario para rellenar el nombre de la persona que va a consumir ese café



Cuando pulsemos en OK, el stock se reducirá en una unidad.

# 6. Comparación de las dos implementaciones

## 6.1 Evaluación de los criterios en la implementación usando la tecnología Django

Debe incluir al menos una tabla con la siguiente estructura.

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Tiempo de instalación | 623s |
| Base de datos | 0s (no requiere aprendizaje) |
| Líneas de código | 485 total |
| Facilidad de aprendizaje | 3h30min |

## 6.2 Evaluación de los criterios en la implementación usando la tecnología Nodejs

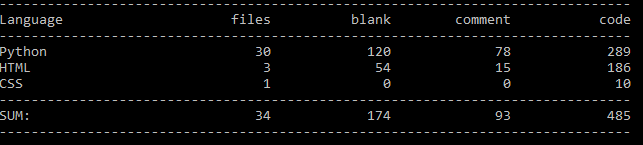
| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Tiempo de instalación | 730s |
| Base de datos | 3h (instalación de mongo) |
| Líneas de código | 61254 total |
| Facilidad de aprendizaje | 5h15min |

## 7. Comparación de la implementación de las tecnologías

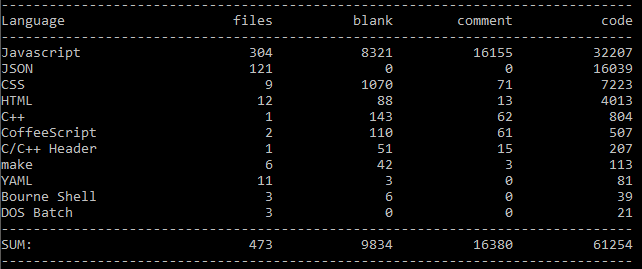
Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

| **CRITERIOS** | **TECNOLOGÍA Django** | **TECNOLOGÍA Nodejs** | **COMENTARIOS** |
| --- | --- | --- | --- |
| Tiempo de instalación | 623s | 730s | Nodejs requiere de más tiempo ya que incluye la instalación de una base de datos. |
| Base de datos | 0s (no requiere aprendizaje) | 3h (instalación de mongo) | Django no necesita ningún aprendizaje de la base de datos porque la crea por defecto al contrario de mongodb que requiere una instalación. |
| Líneas de código | 485 total | 61254 total | Como se puede observar en las imágenes Django utiliza menos líneas de código y es más eficiente que Nodejs . |
| Facilidad de aprendizaje | 3h30min | 5h15min | Nos resulta más fácil Django ya que funciona con Python que es un lenguaje más básico y sencillo. |

Líneas de código Django:



Líneas de código Nodejs:



# 8. Conclusiones

Tras la realización de ambos prototipos del inventario de cafés, se llega a la conclusión de que ambos sistemas son ciertamente potentes y útiles, pero se han encontrado ciertas desventajas y ventajas que serán comentadas a continuación.

**Nodejs** tiene una gran facilidad de comunicación con los navegadores web y con las bases de datos como Mongo, pero el hecho de tratarse de un lenguaje nuevo en el lado del servidor hace que falten librerías generales.

El código generado no se encuentra correctamente organizado, como se puede ver en las tablas generadas de líneas de código, el proyecto de nodejs cuenta con multitud de archivos, subcarpetas y sitios donde es necesario tocar aun siendo esta una aplicación muy simple. Haciendo que la mantenibilidad del código sea una ardua tarea.

**Django** por otro lado no cuenta con prácticamente ninguna de las desventajas mencionadas anteriormente, es elegante y flexible, ordenado y sobre todo simplificado.

Las desventajas de django son que al tratarse de un framework viejo, evoluciona lento en comparación con nodejs. Al basarse en Python, los servidores con dicho hosting son más escasos.

En general, django es más práctico y fácil de usar, por lo que ante un proyecto general, nos decantamos antes por dicho framework.