UNIR SOLUTIONS

Documento de especificaciones (versión 2)

CLIENTE AVI-A

Proyecto:	Versión del documento:	Autor	Fecha
AVICI - A	2	Gontzal Aparicio	21 de julio de 2016

Resumen

El proyecto que se plantea está formado por una aplicación web que permite al usuario obtener ciertas métricas de las estaciones de bicicletas de diversas ciudades españolas y compararlas mediante visualizaciones. Además, pueden incluirse datos acerca de la utilización detallada de cada estación e incluso si da tiempo, un pequeño sistema que permita buscar trayectos más cortos entre dos puntos en base a la disponibilidad de cada estación.

1. REQUISITOS Y ARQUITECTURA

A continuación se detallan los requisitos de de desarrollo, de arquitectura lógica y física y los requisitos de interfaz gráfica de los que se compone este proyecto.

1.1. Requisitos de desarrollo

Los requisitos de desarrollo describen las funcionalidades que debe realizar el sistema durante su ejecución, definiendo estas funcionalidades de una manera simple y concisa.

- El sistema debe de ser capaz de recolectar datos en distintos formatos de diversas páginas web que contengan la información requerida de las estaciones de bicicletas de ciudades seleccionadas previamente.
- El sistema deberá actualizar periódicamente la base de datos para mantenerla actualizada “en tiempo real”.
- Se presentará al cliente, mediante una aplicación web, una serie de visualizaciones de datos que le permitirán extraer conocimiento de datos planos. Entran en este conglomerado estadísticas comparativas entre ciudades como, distancias entre

paradas, uso de bicicletas, recorrido medio recorrido por ciudad, recorrido más largo vs densidad poblacional...etc.

1.2. Requisitos de arquitectura lógica

Seguidamente se definen los requisitos de arquitectura lógica que se han considerado oportunos para la realización eficiente de este proyecto.

- La parte de servidor, será desarrollada en el lenguaje de programación Python, más concretamente en el framework Django para facilitar la creación de la aplicación web y centrar el esfuerzo en los demás aspectos de la herramienta.

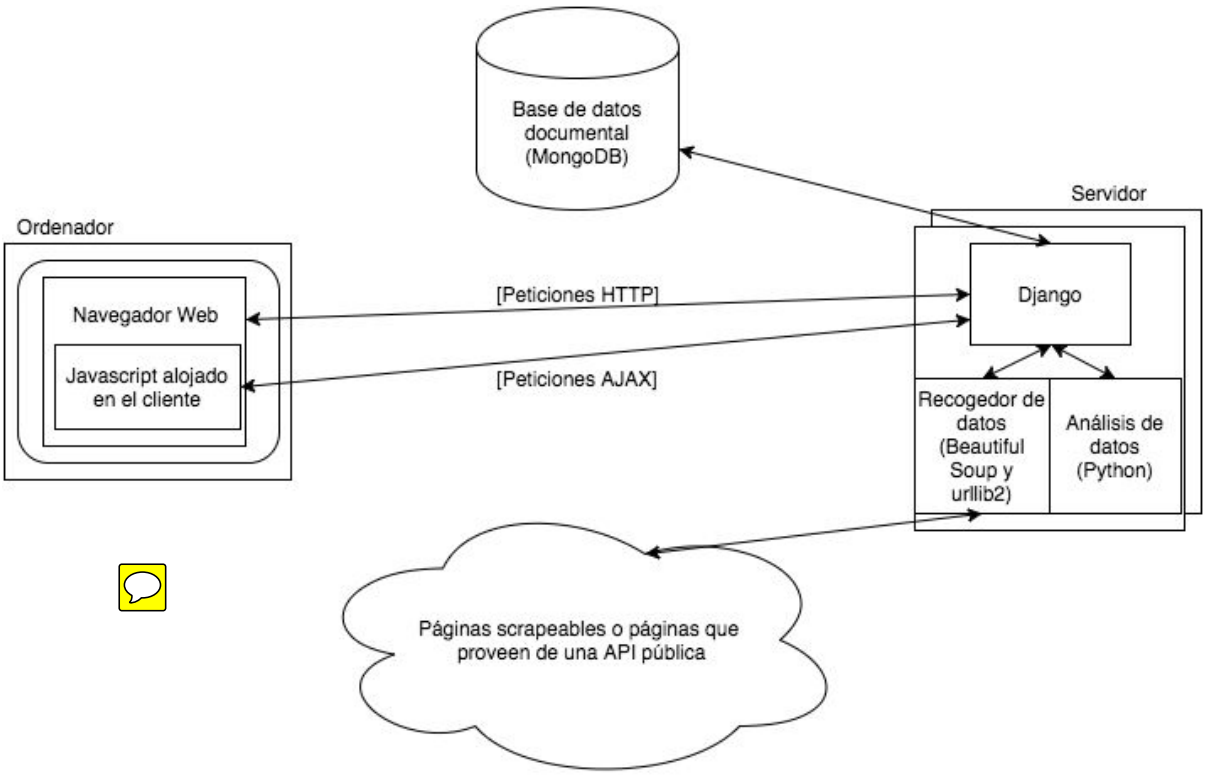
- Se utilizarán diversos paquetes de Python para la ayuda a la hora de scrapear los datos o de analizar los mismo. Para scrapear por ejemplo, se utilizará el paquete “BeautifulSoup”, un paquete muy usado para scrapear datos mediante Python.

- La comunicación entre el cliente y el servidor se hará en gran medida mediante peticiones AJAX (es decir, peticiones asíncronas) procurando facilitar la experiencia de usuario y limitando el número de peticiones HTTP planas que se hagan.

- Las visualizaciones más sencillas han de estar desarrolladas en la librería para Javascript D3.js para permitir mostrar visualizaciones vistosas y con un gran aporte informativo para el usuario. Aquellas que requieran de un mayor esfuerzo como los mapas interactivos, deberán realizarse con una librería auxiliar como Leaflet.js.

- Dado que todos los datos que conseguimos de la mayoría de las API de hoy en día son adquiridas frecuentemente en formato JSON, se deben almacenar estos datos en una base de datos NoSQL. Se ha elegido MongoDB por ser una base de datos orientada a documentos que almacena los datos en BSON, una representación binaria de los JSON. De esta manera, se pretende reducir drásticamente la limpieza de datos que se necesite realizar sobre los datos conseguidos de la API.

Para ilustrar mejor la arquitectura, adjunto a continuación un esquema del planteamiento de la misma en un entorno real:



1.3. Requisitos de interfaz gráfica

Para mejorar la experiencia de usuario se propone una interfaz gráfica que no cambie de subdominio o que cambie lo mínimo posible, realizando cambios en el DOM si fuera necesario para no tener que visualizar los “flashazos” de cambiar de una página a otra.

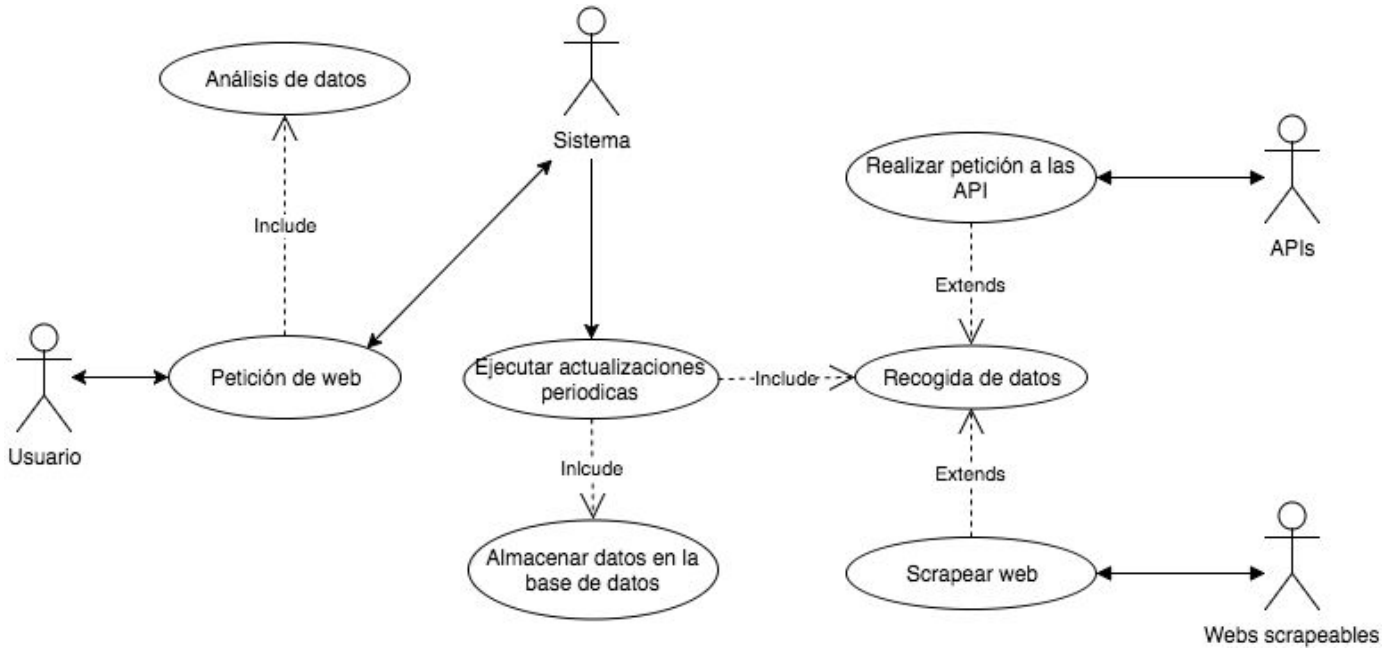
Además, las visualizaciones deberán ser acordes al esquema de colores que se use en la propia interfaz resaltando aquellos elementos que deban destacar por encima de otros.

1.4. Requisitos de arquitectura física

En lo que a requisitos de arquitectura física se refiere, al tener que acceder mediante web, es obvio que el usuario accede desde un navegador lo que impone la necesidad de programar un servidor y un cliente. Por otra parte, el servidor ha de ser capaz de consultar datos a APIs exteriores para poder ofrecerlos a los clientes que hagan peticiones.

2. ANÁLISIS DE CASOS DE USO

En el siguiente capítulo se especifican y analizan los diferentes casos de uso que componen el sistema así como un estudio del diseño técnico del mismo:



En esta primera abstracción de los casos de uso del sistema, podemos observar los distintos actores que interactúan con el mismo:

- **Sistema:** Es habitual no situar al propio sistema como actor en los casos de uso pero he creído conveniente incluirlo en esta ocasión para reflejar la conexión entre las actividades automáticas de las que se encarga el servidor y las peticiones del usuario..

- **APIs:** Se trata de una representación abstracta del conjunto de webs que disponen de una API pública de la que podamos conseguir datos.

- **Web scrapeables:** Representación del conjunto de páginas webs que proporcionar una página idónea para scrapear. Esta diferenciación entre las páginas se realiza porque el método para conseguir los datos de una u otra es muy diferente.

- **Usuario:** Es el actor que representa a los usuarios finales. En definitiva todos aquellos que usarán el sistema.

Paso a detallar resumidamente cada uno de los casos de uso del diagrama anterior:

Caso de uso: 01 - Petición de web (CdU01)

a. *Actores*: Usuario, Sistema
b. *Descripción:* Un usuario ha accedido al sistema y ha solicitado un informe de comparación entre dos ciudades.
c. *Precondición*: El usuario debe entrar en la web.
d. *Flujo básico:*
    1. El usuario entra en la web;
    2. El usuario elige qué ciudades desea comparar y realiza la petición;
e. *Flujo alternativo:* -
f. *Postcondición*: El servidor empieza a realizar tareas y el usuario ha de quedar a la espera con una pantalla de carga.

Caso de uso: 02 - Análisis de datos (CdU02)

a. *Actores*: Usuario, Sistema
b. *Descripción:* Un usuario ha accedido al sistema, ha solicitado un informe de comparación entre dos ciudades y el sistema debe analizar los datos de la base de datos para enviarle los resultados al usuario.
c. *Precondición*: El usuario ha realizado una petición web (CdU01).
d. *Flujo básico:*
    1. El sistema obtiene los datos necesarios de la base de datos;
    2. El sistema realiza los análisis pertinentes y devuelve los datos al usuario;
e. *Flujo alternativo:* -
f. *Postcondición*: El servidor ha realizado los análisis oportunos y ha devuelto los datos al usuario.

Caso de uso: 03 - Ejecutar actualizaciones periódicas (CdU03)

a. *Actores*: Sistema
b. *Descripción:* Ha pasado cierto tiempo desde la última actualización de datos y se deben actualizar de nuevo.
c. *Precondición*: Ninguna.
d. *Flujo básico:*

1. El script del sistema detecta que ha pasado el tiempo suficiente para realizar una nueva petición de datos;
2. El sistema comienza a ejecutar rutinas para actualizar datos.

e. *Flujo alternativo:* -

f. *Postcondición*: El servidor comienza a ejecutar rutinas para actualizar los datos..

Caso de uso: 04 - Almacenar los datos en la base de datos (CdU04)

a. *Actores*: Sistema

b. *Descripción:* El sistema se encarga de almacenar los datos recogido en la base de datos.

c. *Precondición*: Se ha tenido que ejecutar la rutina de actualización de datos(CdU03) y se deben de haber recogido los datos (CdU05).

d. *Flujo básico:*

1. El sistema estandariza los datos que ha recogido;
2. El sistema comprueba que los datos no están duplicados;
3. El sistema introduce los datos en la base de datos;

e. *Flujo alternativo:* -

f. *Postcondición*: Los datos recogido no duplicados se encuentran introducidos en la base de datos.

Caso de uso: 05 - Recogida de datos (CdU05)

a. *Actores*: Sistema, APIs, Web scrapeables

b. *Descripción:* El sistema debe recoger datos para todas las páginas que se hayan planteado.

c. *Precondición*: Se ha tenido que ejecutar la rutina de actualización de datos(CdU03).

d. *Flujo básico:*

1. El sistema busca las páginas de las que ha de recolectar los datos;
2. Para cada página realiza la tarea necesaria: scrapear (CdU07) o obtener datos de una api (CdU08);

e. *Flujo alternativo:* -

f. *Postcondición*: Los datos recogidos no duplicados se encuentran listos para enviar al CdU04 para ser almacenados en la base de datos.

Caso de uso: 06 - Realizar petición de datos a API (CdU06)

a. *Actores*: Sistema, APIs
b. *Descripción:* El sistema debe recoger los datos de una página que proporciona una API pública conocida.
c. *Precondición*: Se ha tenido que ejecutar la rutina de actualización de datos(CdU03) y la recogida de datos (CdU05) ha de haber seleccionado una página con API pública.
d. *Flujo básico:*
    1. El sistema realiza una petición a la API correspondiente;
    2. Se devuelve al sistema el JSON necesario para almacenar a la base de datos;
e. *Flujo alternativo:* -
f. *Postcondición*: Se han recibido los datos necesarios de la API pública de la página.

Caso de uso: 07 - Scrapear web (CdU07)

a. *Actores*: Sistema, Webs scrapeables
b. *Descripción:* El sistema debe scrapear los datos de una página que proporciona una página idónea para ser scrapeada (por ejemplo, con datos en tablas).
c. *Precondición*: Se ha tenido que ejecutar la rutina de actualización de datos(CdU03) y la recogida de datos (CdU05) ha de haber seleccionado una página con HTML scrapeable.
d. *Flujo básico:*
    1. El sistema realiza una petición a la página correspondiente;
    2. Se devuelve al sistema el HTML necesario para scrapear;
    3. Se obtienen los datos necesarios del HTML y se devuelven al sistema para poder introducirlos a la base de datos.
e. *Flujo alternativo:* -
f. *Postcondición*: Se han recibido los datos necesarios de la página scrapeable y se han conseguido los datos mediante técnicas de scrapping.

3. TECNOLOGÍAS A UTILIZAR:

Para desarrollar el proyecto se proponen las siguientes tecnologías:

- **Desarrollo del servidor - Django (https://www.djangoproject.com/):**
  Para empezar Django proporciona un servidor web incluido para el desarrollo lo que facilita el desarrollo del de producción. Django también proporciona un sistema de plantilla con lenguaje propio que permite construir vistas más rápidamente e intuitivamente. Debido al largo recorrido del framework cuenta con una extensa comunidad y una gran cantidad de documentación en diversos idiomas. Además al ser un framework de Python permite utilizar los distintos paquetes del mismo facilitando tareas de análisis.

- **Análisis de datos- Python (https://www.python.org/):**
  Con una curva de aprendizaje muy sencilla y unas funcionalidades orientadas a la buena escritura de código Python se ha hecho muy popular durante los últimos años para el análisis de datos. La comunidad es más grande y más documentada que otros lenguaje como R y esto incluye a los paquetes que se generan para el lenguaje.

- **Visualizaciones - D3.js(https://d3js.org/), Mapbox.js(https://www.mapbox.com/):**
  Dado que en mi trabajo habitual estoy acostumbrado a usar D3.js esta es la librería más acertada a mi modo de ver. Permite desarrollar interacciones más detalladas y visualizaciones más personalizadas que cualquier otro paquete de visualización existente hoy en día. Además, para construir mapas, Mapbox permite realizar visualizaciones vistosas y fácilmente integrables con D3.js.

- **Base de datos - MongoDB(https://www.mongodb.com/es):** La mayoría de APIs de hoy en día nos devuelven los datos en formato JSON. Es debido a ello que considero que la opción más acertada para almacenar los datos es un sistema NoSQL como MongoDB en ved de un sistema relacional tradicional.

4. CAMBIOS REALIZADOS:

Se han realizado una serie de cambios respecto a la primera versión de los requisitos, entre ellos:

- Se ha introducido un índice en la primera página para facilitar la búsqueda de los apartados.

- Se ha introducido un cajetín que especifica la versión del documento así como el autor y la fecha de realización del mismo.

- Se ha añadido un diagrama de arquitectura lógica para facilitar la comprensión global del sistema en la página 4.

- Se han añadido enlaces para referenciar las tecnologías propuestas.