UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación Web para la recopilación, tratamiento y visualización de datos públicos

Presentado por Iván Arjona Alonso
en Universidad de Burgos — 17 de mayo de 2018
Tutor: Dr. José Francisco Díez Pastor
y Dr. Jesús Manuel Maudes Raedo

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

El desarrollo del proyecto se ha llevado a cabo utilizando metodologias ágiles, basandose en la metodología *scrum* con algunas modificaciones (una sóla persona y sin reuniones diarias).

Se aplicó una estrategia de desarrollo incremental, con iteraciones que llamaremos *sprints*.

El resultado de cada iteración es un entregable, sobre el que se discute en la reunión posterior a cada sprint.

Se realizó, en principio, una reunión a la semana con los tutores para exponer las modificaciones realizadas en el sprint anterior y planificar los cambios a realizar en la siguiente iteración.

Estas tareas están priorizadas por el tiempo estimado de su realización, se puede ver esta estimación en el enlace de cada sprint a sus tareas.

A continuación se va a realizar un breve resumen de las tareas realizadas en cada una de las iteraciones, así como la duración de cada sprint y el gráfico *burndown* correspondiente.

Sprint 0 (16/02/2018 - 02/03/2018)

Primer sprint del proyecto. En la la reunión de planificación de este primer sprint se discute de forma general de lo que va a tratar el proyecto.

Las tareas realizadas durante este sprint fueran la creación y configuración del repositorio y sobre todo investigar sobre las tecnologías y herramientas que se podían utilizar.

Se investigaron posibles fuentes de datos para implementar más adelante: INE, sepe, aeat.

La duración fue de dos semanas en lugar de una para poder documentarse sobre todos los aspectos relevantes del proyecto.

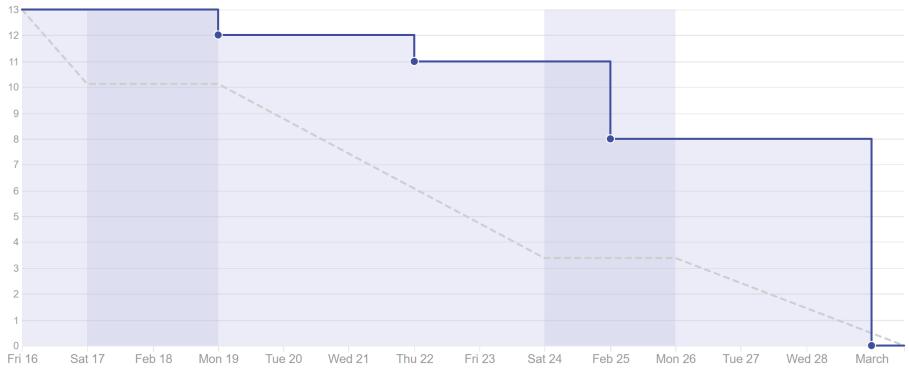

Figura A.1: Burndown del sprint 0

Tareas del sprint 0 en Github

Sprint 1 (03/03/2018 - 08/03/2018)

Un objetivo de este sprint es investigar alternativas de bases de datos no relacionades que se podrían utilizar. Se ha elegido MongoDB.

El otro objetivo es empezar a implementar prototipos con las fuentes de datos que se habían encontrado en el sprint anterior. Se implementaron prototipos del INE, de la agencia tributaria y del sepe.

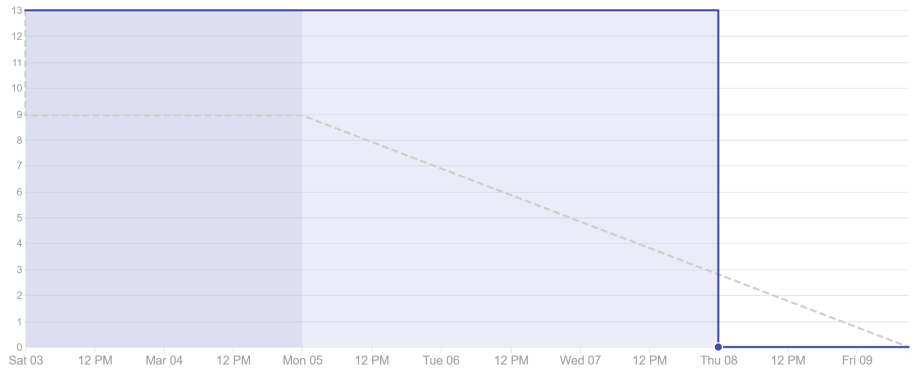

Figura A.2: Burndown del sprint 1

Tareas del sprint 1 en Github

Sprint 2 (09/03/2018 - 15/03/2018)

El primer objetivo de este sprint es crear la estructura de la página web con Flask. Utilizando un modelo vista-controlador. También se utiliza bootstrap para ahorrar trabajo en el diseño.

Se implementó un prototipo de la carga de datos hacia la base de datos y otro para la descarga de datos desde la base de datos para ser mostrados.

Se hizo una implementación de las fuentes de datos a partir de los prototipos del sprint 1 de modo que se pueda cargar todas las fuentes de manera automática.

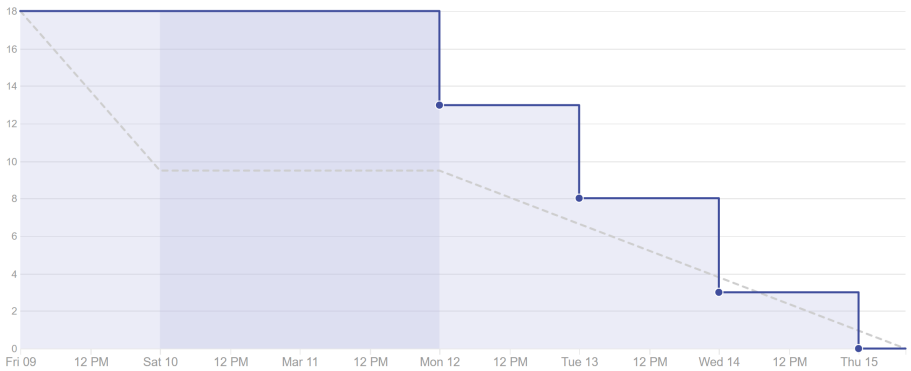

Figura A.3: Burndown del sprint 2

Tareas del sprint 2 en Github

Sprint 3 (16/03/2018 - 22/03/2018)

En este sprint se sopesaron varias plataformas para hacer el despliegue de la web. De ellas se eligió DigitalOcean y Nanobox.

Se realizó el despliegue utilizando estas plataformas. Web desplagada.

Se investigaron los posibles riesgos de seguridad como inyecciones NoSQL.

Se implementó un formulario para la consulta de datos en la página web. En esta primera aproximación se podía hacer una consulta comparando con una columna de una de las fuentes de datos.

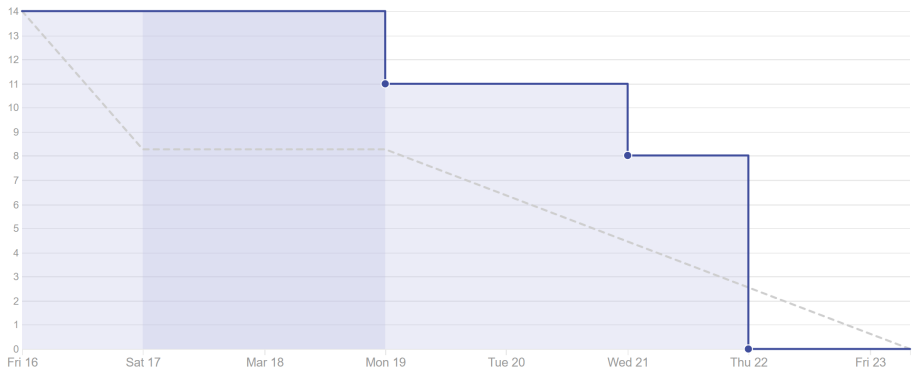

Figura A.4: Burndown del sprint 3

Tareas del sprint 3 en Github

Sprint 4 (23/03/2018 - 13/04/2018)

Este sprint coincide con semana santa, por lo que dura una semana más de lo habitual y la carga de trabajo también es mayor.

Se corrigieron errores en los tipos de las fuentes de datos al tratar con números como cadenas.

Se implementó una forma de descargar las consultas a partir del formulario.

Se mejoró interfaz gráfica y el formulario de consulta.

Se modificaron las fuentes de datos para corregir errores y añadir el código de municipio a todas ellas para más tarde poder unirlas.

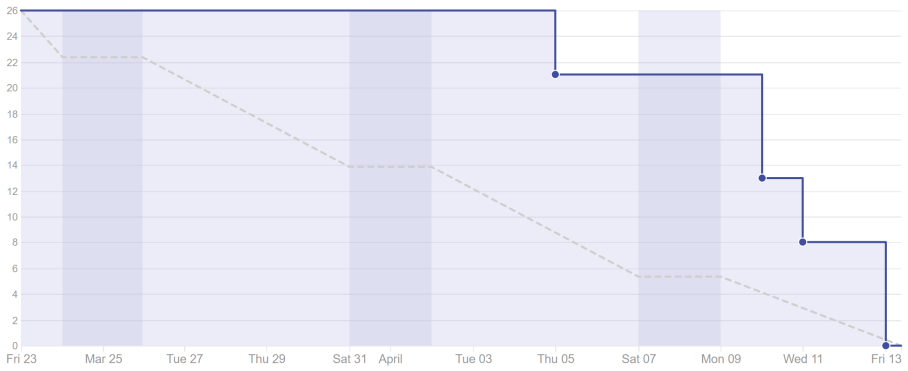

Figura A.5: Burndown del sprint 4

Tareas del sprint 4 en Github

Sprint 5 (14/04/2018 - 25/04/2018)

Este sprint se dedicó a empezar a documentar la memoria y corregir algunos errores en la página de consulta como el reenvio de formularios en firefox y la descarga de consultas en json y csv.

También se añadió una descripción a la fuente de datos para explicar de qué se trata cada una en la interfaz web.

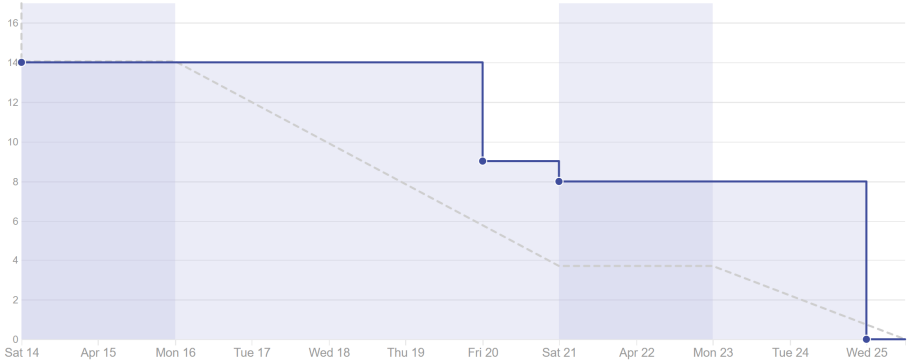

Figura A.6: Burndown del sprint 5

Tareas del sprint 5 en Github

Sprint 6 (26/04/2018 - 02/05/2018)

El objetivo principal de este sprint fue implementar la posibilidad de juntar varias subconsultas mediante un join. Otra característica implementada es la de avisar al usuario si se ha sobrepasado el límite de columnas especificado, para que pueda filtrar más fino.

También se consideró hacer cambios en el modelo de datos, pero debido a la alta dimensionalidad se ha dejado como estaba en el sprint anterior.

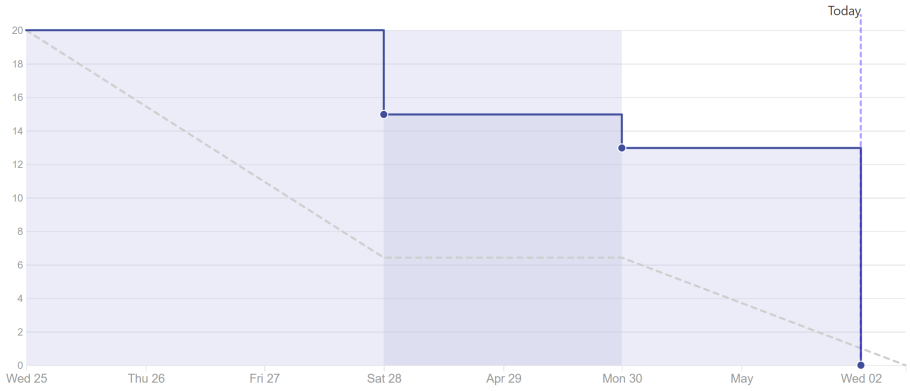

Figura A.7: Burndown del sprint 6

Tareas del sprint 6 en Github

Sprint 7 (03/05/2018 - 09/05/2018)

En este sprint se implementó un mapa coroplético para mostrar los valores de cualquier atributo en el mapa agrupando los municipios por su provincia.

Se eliminar los campos duplicados de las consultas que surgían al realizar join de varias subconsultas. Como estos campos repetidos siempre son iguales, se ha optado por eliminarlos en lugar de renombrarlos.

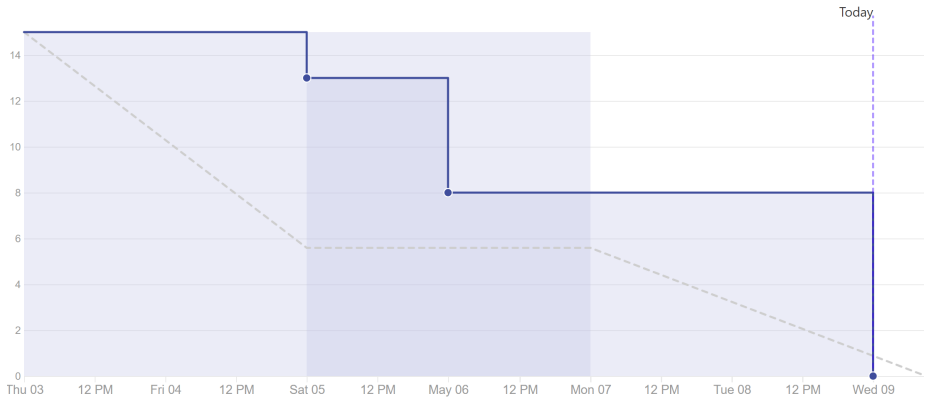

Figura A.8: Burndown del sprint 7

Tareas del sprint 7 en Github

Sprint 8 (10/05/2018 - 16/05/2018)

En este sprint se implementó una aplicación para juntar varios ficheros csv en uno sólo utilizando join. Para poder aplicar técnicas de minería de datos sobre estos ficheros.

Se añadió una funcionalidad de poder mostrar mapas coropléticos a nivel de municipio, además de a nivel de provincia.

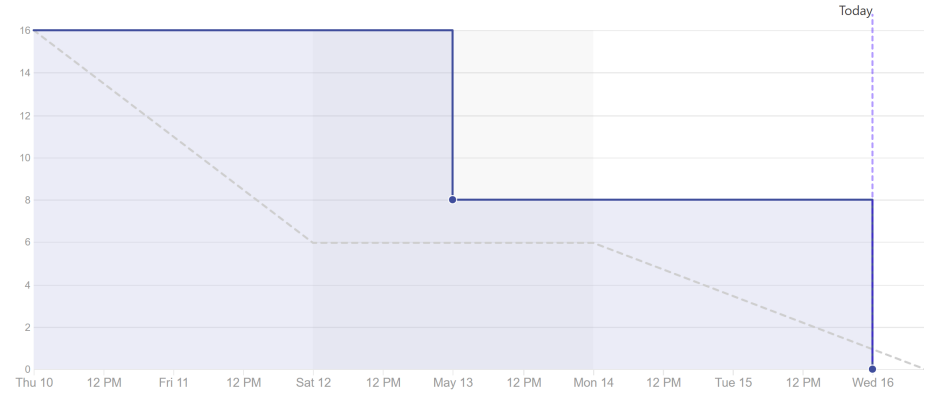

Figura A.9: Burndown del sprint 8

Tareas del sprint 8 en Github

Concepto	Coste (€)
Salario neto	1000 €
Retención IRPF (19 %)	360.53 €
Seguridad social (28,30 %)	537.00 €
Salario bruto	1897.53 €
Total 4 meses	7592.59 €

Tabla A.1: Costes de personal

A.3. Estudio de viabilidad

Viabilidad económica

Costes de personal

El proyecto se ha llevado a cabo por un desarrollador empleado a tiempo parcial durante 4 meses. Se considera un salario neto de 1000 €. (Ver tabla A.1)

Los percentajes de cotización a la seguridad social se han calculado a partir del régimen general para 2018 como horas comunes (23.6 % empresa y 4.7 % trabajadores) [5].

Costes de material

En esta sección se incluyen los costes de software y hardware.

Como hardware se incluye el coste del equipo que se ha utilizado para desarrollar la aplicación (un único pago de 1000 €) y el servidor web (5 € mensual). (Ver tabla A.2)

Concepto	Coste (€)
Ordenador portátil	1000 €
Servidor web (mensual)	5 €
Total 4 meses	1020 €

Tabla A.2: Costes de hardware

Como software incluimos la licencia de Windows como único pago y las licencias de Github developer, Codacy Pro y PyCharm como pago mensual.

Gitkraken Pro supone un coste de 41 € anual, como no hay pago mensual, se considerará el pago completo. (Ver tabla **??**)

Concepto	Coste (€)
Windows 10 Home	145 €
GitHub Developer (mensual)	7 €
Codacy Pro Plan (mensual)	15 €
PyCharm (mensual)	8.90 €
GitKraken Pro (anual)	41 €
Total 4 meses	309.60 €

Tabla A.3: Costes de software

Costes totales

Tendremos un coste total de 8922.19 €. (Ver tabla A.4)

Concepto	Coste (€)
Personal	7592.59 €
Hardware	1020 €
Software	309.60 €
Total	8922.19 €

Tabla A.4: Costes de software

Beneficios

Si se quisiera rentabilizar el proyecto se pueden considerar varias alternativas.

- Inclusión de publicidad.
- Modelo *freemium* [7]: incluir algunas características más avanzadas de pago.
- Limitar la cantidad de datos a mostrar poniendo una barrera económica.

Dependencia	Versión	Licencia
beautifulsoup4	4.6.0	MIT
branca	0.2.0	MIT
chardet	3.0.4	MIT
click	6.7	BSD
dominate	2.3.1	LGPL-3.0
Flask	1.0.2	BSD
Flask-Bootstrap	3.3.7.1	BSD
Flask-PyMongo	0.5.1	BSD
Flask-WTF	0.14.2	BSD
folium	0.5	MIT
itsdangerous	0.24	BSD
Jinja2	2.10	BSD
MarkupSafe	1.0	BSD
numpy	1.14.3	BSD
pandas	0.22.0	BSD
pymongo	3.6.1	Apache-2.0
requests	2.18.4	Apache-2.0
six	1.11.0	MIT
urllib3	1.22	MIT
WTForms	2.1	BSD
xldr	1.1.0	BSD
Bootstrap	3.3.7	MIT
jQuery	1.12.4	MIT
Dynatable	0.3.1	AGPL

Tabla A.5: Dependencias

Viabilidad legal

Con la ayuda de VersionEye [6] se han listado las dependencias del proyecto, con sus correspondientes licencias (Ver tabla A.5).

De las dependencias usadas en el proyecto, la licencia más restrictiva es LGPL-3.0 [2]. Esta licencia permite el uso de la librería con total libertad, por lo que no nos restringe en la licencia que tenemos que utilizar.

Se ha decidido utilizar la licencia GNU-3.0 [1]. Con esta licencia se permite el uso, distribución y modificación del proyecto, siempre que se mantenga la misma licencia y se acredite al autor original.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se describirán los objetivos de la aplicación y se detallarán tanto los requisitos funcionales como los no funcionales.

B.2. Objetivos generales

- Integrar varias fuentes de datos públicas en una única base de datos. Estos datos son datos de carácter sociológicos, económicos y demográfico a nivel municipal en España.
- Permitir añadir nuevos conjunto de datos de forma sencilla.
- Desarrollar un algoritmo para la carga de varias fuentes de datos en una base de datos de manera automatizada.
- Desarrollo de una aplicación web que permita la consulta de los datos de manera sencilla y visual.
- Facilitar la interpretación de los datos utilizando un mapas coropléticos interactivo.
- Desplegar la aplicación web en un servidor de forma que sea fácil de actualizar cada vez que se realice un cambio. Además de funcionar en un entorno local.

B.3. Catalogo de requisitos

Requisitos funcionales

Requisitos no funcionales

- **RNF-1 Usabilidad**: La aplicación debe ser fácil de usar e intuitiva para el usuario.
- **RNF-2 Rendimiento**: La aplicación debe cargar en un tiempo aceptable.
- **RNF-3 Mantenimibilidad**: La aplicación debe permitir añadir características de forma sencilla.
- **RNF-4 Compatibilidad**: La aplicación debe funciona correctamente en los navegadores modernos más utilizados (Edge, Chrome, Firefox, Opera y Safari).
- **RNF-5 Responsividad**: La aplicación debe funcionar en pantallas de cualquier tamaño y adaptar su interfaz a cada pantalla.
- **RNF-6 Escalabilidad**: La aplicación debe poder aumentar su rendimiento al aumentar recursos hardware.
- **RNF-7 Facilidad de despliegue**: La aplicación debe poder desplegarse en un servidor de forma sencilla.
- **RNF-8 Software libre**: Utilizar software libre siempre que sea posible.

B.4. Especificación de requisitos

Diagrama de casos de uso

Descripcion de casos de uso

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se explica todo lo que tiene que conocer el programador tanto para instalar y ejecutar la aplicación como para poder seguir con el desarrollo.

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Instalación y ejecución del proyecto

Instalación

MongoDB

Antes de empezar con la instalación de la aplicación tenemos que instalar la base de datos. Se ha utilizado una base de datos no relacional MongoDB.

Concretamente se instaló la versión *3.6.4 Community Server* [3] para windows.

También se ha utilizado *MongoDB Compass* [3] para visualizar el contenido de la base de datos. Esta herramienta es opcional.

Python

Esta aplicación se ha desarrollado utilizando la version 3.6.4 [4], por lo que se recomienda utilizar esta versión o una posterior. En cualquier caso debe de instalarse Python 3 o superior para evitar incompatibilidades.

Todos los paquetes utilizados en la aplicación están listados en en ficheros requirements.txt junto con sus correspondientes versiones.

Habrá que instalar todas las dependencias utilizando pip[1]:

```
pip install -r requirements.txt
```

Ejecución

Actualización de la base de datos

Antes de ejecutar la aplicación tendremos que construir la base de datos con el contenido de todas nuestras fuentes por primera vez.

Para ello simplemente hay que ejecutar el fichero *actualiza-fuentes.py*. Puede tardar un rato debido a la gran cantidad de datos que se van a cargar.

Este paso puede repetirse cada vez que se quiera actualizar las fuentes de datos. Puede ser utilizar ejecutarlo una vez al mes para mantener al día las fuentes con datos mensuales.

```
python actualiza-fuentes.py
```

Servidor web

El servidor Flask ya se ha instalado como un paquete, por lo que no necesita más instalación. Para ejecutarlo hay un script *run.py* que nos lanza el servidor en el puerto 5000.

Una vez lanzado podremos acceder a la página web desde localhost:5000.

```
python run.py
```

D.5. Despliegue

Para desplegar la aplicación se ha obtado utilizar como servidor en la nube DigitalOcean y Nanobox como microservicio para facilitarnos la instalación de la máquina en la nube y la configuración del servidor.

[1] Gestor de paquetes de Python [**?**]

Instalación

Digital Ocean

Primero tendremos que registrarnos en DigitalOcean y obtener el token de nuestro usuario. Tendremos que poner una tarjeta de crédito de la que nos cobrarán el importe del servidor.

Podría utilizarse otro proveedor de hosting como *Amazon Web services* o *Google Compute*.

Nanobox

Después nos registraremos en Nanobox y creamos una nueva aplicación utilizando el token que hemos obtenido antes en DigitalOcean.

Ahora elegiremos el plan que queramos contratar. Para este proyecto será suficiente con el plan más basico de 5$, 1 CPU y 1GB de ram.

Tras crear la aplicación instalamos el cliente de nanobox. La primera vez que lancemos un comando nos pedirá los datos para iniciar sesión en nuestra cuenta.

En el fichero *nanobox.yml* tenemos la configuración con los componentes que se van a instalar y los comandos que se ejecutan al desplegar la aplicación.

Despliegue

Una vez todo instalado y configurado, cada vez que queramos actualizar la aplicación del servidor nos situamos en la carpeta del proyecto y lanzamos el siguiente comando:

```
nanobox deploy
```

Con esto, de forma transparente para nosotros, se crea una máquina virtual con nuestro proyecto en la que se instalan la base de datos, el entorno de ejecución y los paquetes y se ejecuta la aplicación en el servidor.

D.6. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] GNU. Gnu general public license. https://www.gnu.org/licenses/gpl-3.0.en.html, junio 2007. [Internet; descargado 14-mayo-2018].

[2] GNU. Gnu lesser general public license. https://www.gnu.org/licenses/lgpl-3.0.en.html, junio 2007. [Internet; descargado 14-mayo-2018].

[3] MongoDB. Mongodb download center. https://www.mongodb.com/download-center?jmp=nav#community, abril 2018. [Internet; descargado 29-abril-2018].

[4] Python. Python 3.6.4. https://www.python.org/downloads/release/python-364/, diciembre 2017. [Internet; descargado 29-abril-2018].

[5] Seguridad Social. Seguridad social: Bases y tipos de cotización 2018. http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm#36538, 2018. [Internet; descargado 14-mayo-2018].

[6] VersionEye. Versioneye tfg-datos-publicos dependencies. https://www.versioneye.com/user/projects/5ad84cd30fb24f5450e020ce#tab-dependencies, mayo 2018. [Internet; descargado 14-mayo-2018].

[7] Wikipedia. Freemium — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Freemium&oldid=106331717, marzo 2018. [Internet; descargado 16-mayo-2018].