



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática

título del TFG  
Documentación Técnica



Presentado por nombre alumno  
en Universidad de Burgos — 7 de enero de 2016  
Tutor: nombre tutor

---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Apéndice A Manuales</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	10
<b>Apéndice B Especificación de Requisitos</b>	<b>14</b>
B.1. Introducción . . . . .	14
B.2. Objetivos generales . . . . .	14
B.3. Catalogo de requisitos . . . . .	14
B.4. Especificación de requisitos . . . . .	14
<b>Apéndice C Especificación de diseño</b>	<b>16</b>
C.1. Introducción . . . . .	16
C.2. Diseño de datos . . . . .	16
C.3. Diseño procedimental . . . . .	16
C.4. Diseño arquitectónico . . . . .	16
<b>Apéndice D Documentación técnica de programación</b>	<b>18</b>
D.1. Introducción . . . . .	18
D.2. Estructura de directorios . . . . .	18
D.3. Manual del programador . . . . .	18
D.4. Instalación, compilación y ejecución del proyecto . . . . .	18
D.5. Pruebas del sistema . . . . .	20
<b>Apéndice E Documentación de usuario</b>	<b>21</b>
E.1. Introducción . . . . .	21

<i>ÍNDICE GENERAL</i>	II
E.2. Requisitos de usuarios . . . . .	21
E.3. Instalación . . . . .	21
E.4. Manual del usuario . . . . .	21
<b>Bibliografía</b>	<b>22</b>

---

# Índice de figuras

---

A.1. Gráfico Burndown Iteración 2 . . . . .	2
A.2. Gráfico Burndown Iteración 3 . . . . .	3
A.3. Gráfico Burndown Iteración 4 . . . . .	4
A.4. Gráfico Burndown Iteración 5 . . . . .	5
A.5. Gráfico Burndown Iteración 6 . . . . .	6
A.6. Gráfico Burndown Iteración 7 . . . . .	7
A.7. Gráfico Burndown Iteración 8 . . . . .	8
A.8. Gráfico Burndown Iteración 9 . . . . .	9
C.1. Esquema de conexión MVP . . . . .	17
C.2. Ejemplo aplicación Android con MVP . . . . .	17
D.1. Selección de la versión de NetBeans . . . . .	19
D.2. Selección de descarga Tesseract . . . . .	20

## Apéndice A

---

# Manuales

---

### A.1. Introducción

### A.2. Planificación temporal

A continuación se realiza una breve explicación de las tareas realizadas en cada iteración.

#### Iteración 1 [28/10/2015 - 04/11/2015]

Esta iteración dura una semana. En ella se realiza principalmente 3 tareas. Se inician los proyectos de Servidor y Cliente de manera independiente y se trabaja en la forma de mandar la imagen del cliente al servidor. Se empieza a pensar en el modelo de datos.

Historia	Horas Estimadas	Horas Reales
Creación del modelo de datos	1	10
Creación del Servidor	1	1
Creación del Cliente	1	1
Envío de imágenes Cliente al Servidor.	6	8

Cuadro A.1: Tareas de la primera iteración

#### Iteración 2 [04/11/2015 - 11/11/2015]

Esta iteración dura una semana. En ella se empieza a desarrollar el tratamiento de la imagen, se decide cambiar de librería, ImageJ a OpenCv. Se termina el desarrollo del modelo de la iteración anterior y se empieza el desarrollo de la Base de Datos.

Historia	Horas Estimadas	Horas Reales
Creación del modelo de datos	3	3
Creación de el Sqlite	8	10
Inclusión de las librerías Open CV Servidor	10	14

Cuadro A.2: Tareas de la segunda iteración

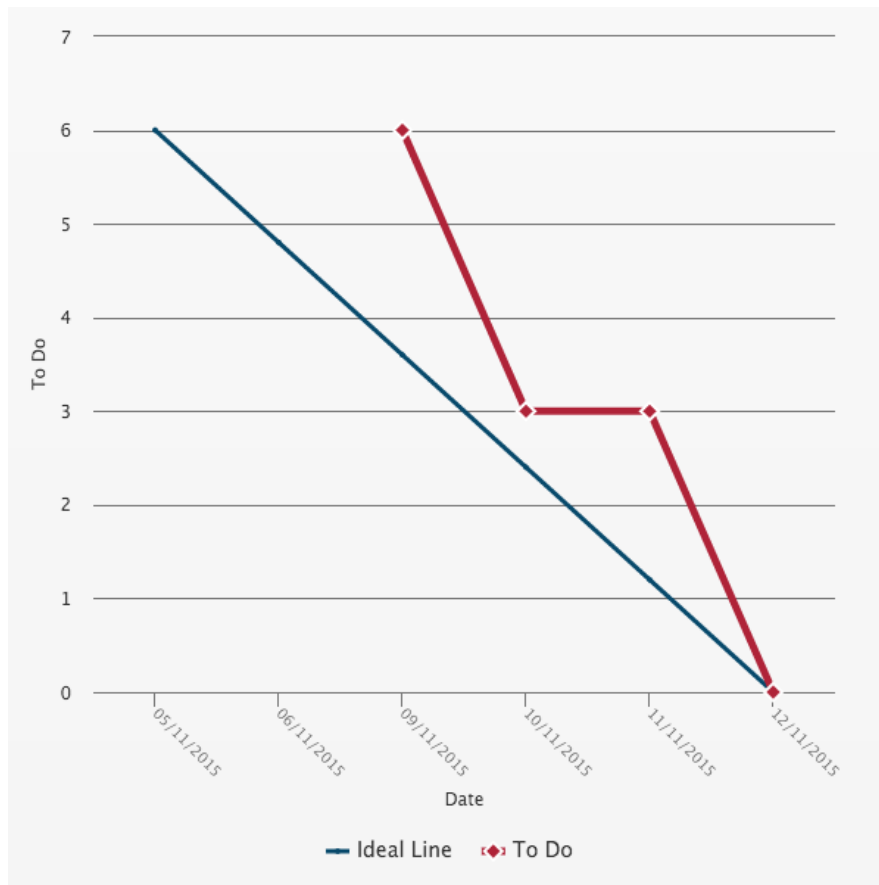


Figura A.1: Gráfico Burndown Iteración 2

**Iteración 3 [11/11/2015 - 18/11/2015]**

Esta iteración dura una semana. En ella se desarrolla el sistema para cortar la imagen antes de enviarla al servidor.

Historia	Horas Estimadas	Horas Reales
Creacion del modelo de datos	5	2. Crop de la imagen para el procesado en el ser
Diseño de layouts	10	8

Cuadro A.3: Tareas de la tercera iteración

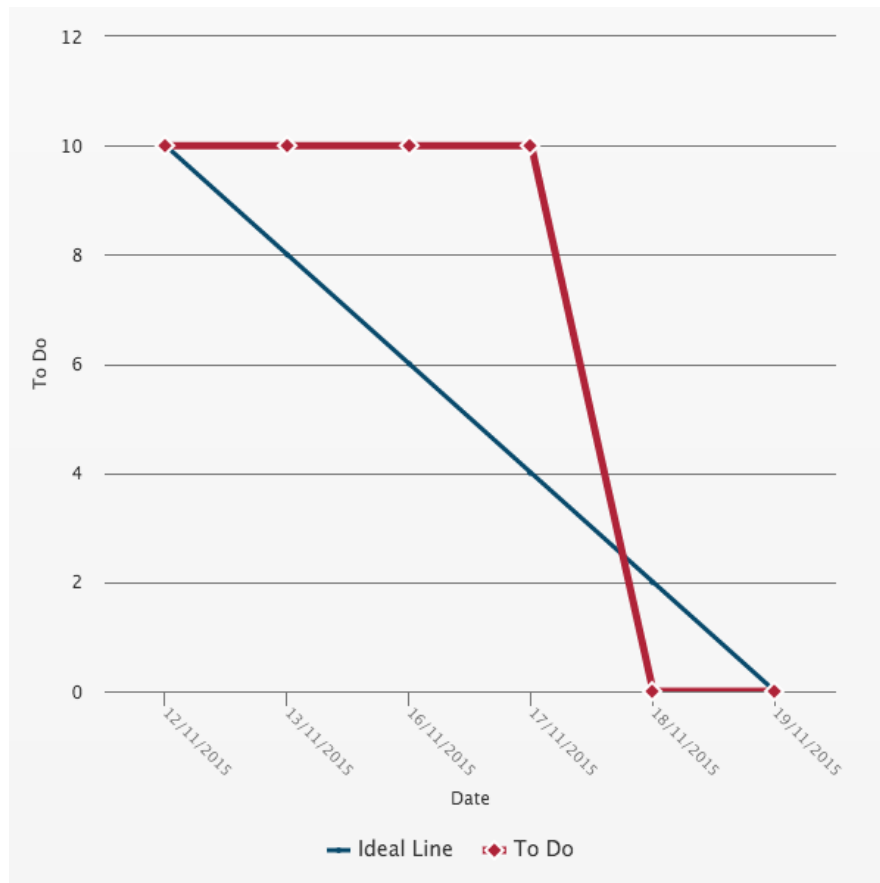


Figura A.2: Gráfico Burndown Iteración 3

**Iteración 4 [18/11/2015 - 02/12/2015]**

Esta iteración dura dos semanas. En ella se desarrolla el sistema para alinear la imagen y obtener las líneas de producto a partir de el recorte, para poder enviarlo al tesseract. El gráfico Burndown se muestra que no se añadieron las horas, por ello, es una línea plana.

Historia	Horas Estimadas	Horas Reales
Algoritmo de Deskewing	10	13
Obtención de Productos	10	10
Binarización Correcta de los productos	10	10
Inclusión Tesseract	4	20

Cuadro A.4: Tareas de la cuarta iteración

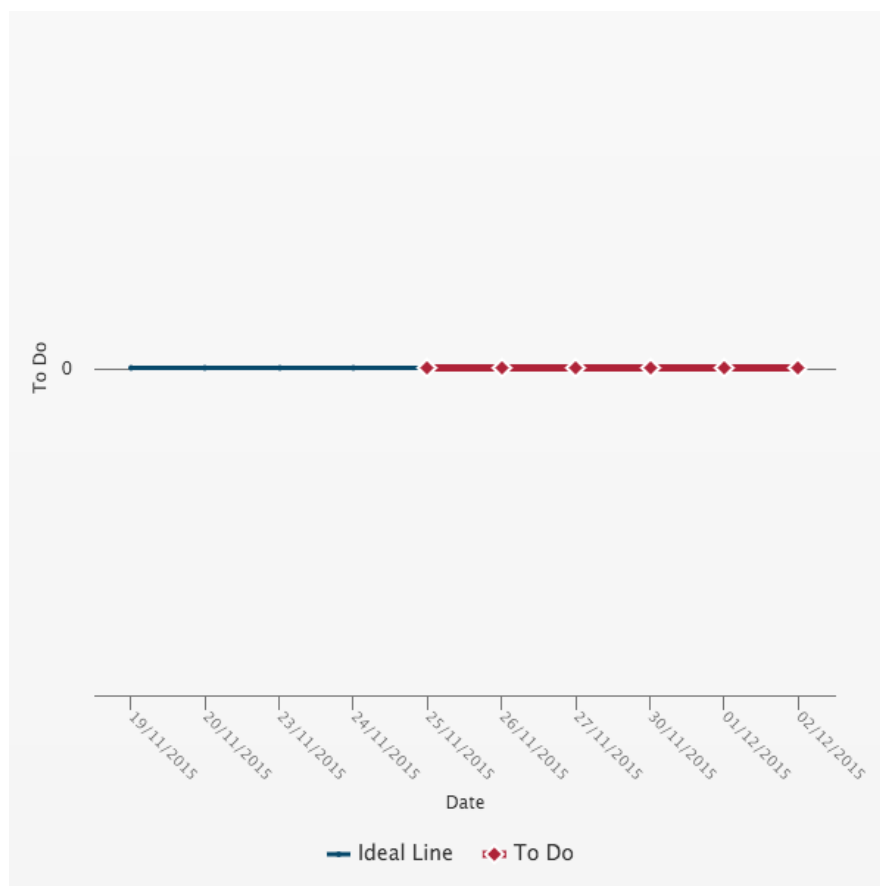


Figura A.3: Gráfico Burndown Iteración 4



**Iteración 5 [02/12/2015 - 16/12/2015]**

Esta iteración dura dos semanas. En ella se busca e implementa la forma de corregir los errores que se producen al pasar la imagen a texto por tesseract. Se empieza a trabajar en el envío de los productos al cliente.

Historia	Horas Estimadas	Horas Reales
Corrección de números linea producto	10	9
Creación de JSON	4	2
Corrección de productos	4	6

Cuadro A.5: Tareas de la quinta iteración

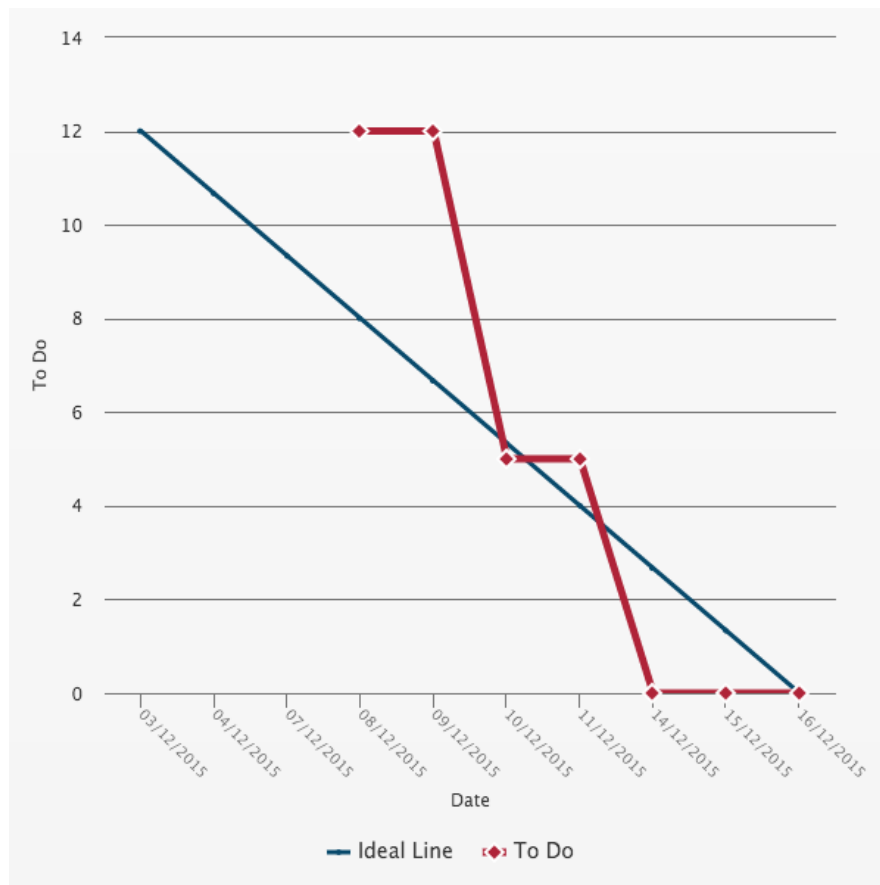


Figura A.4: Gráfico Burndown Iteración 5

**Iteración 6 [16/12/2015 - 23/12/2015]**

Esta iteración dura una semana. En ella se desarrolla trabajo en la inserción de los tiques en el cliente. Se desarrolla tanto la vista como la funcionalidad. Se empieza a trabajar en la vista de filtro de productos

Historia	Horas Estimadas	Horas Reales
Conexión correcta con el servidor	2	2
Diseño de Guardado de productos	14	10
Diseño de Visionado de productos por filtros	15	20

Cuadro A.6: Tareas de la sexta iteración

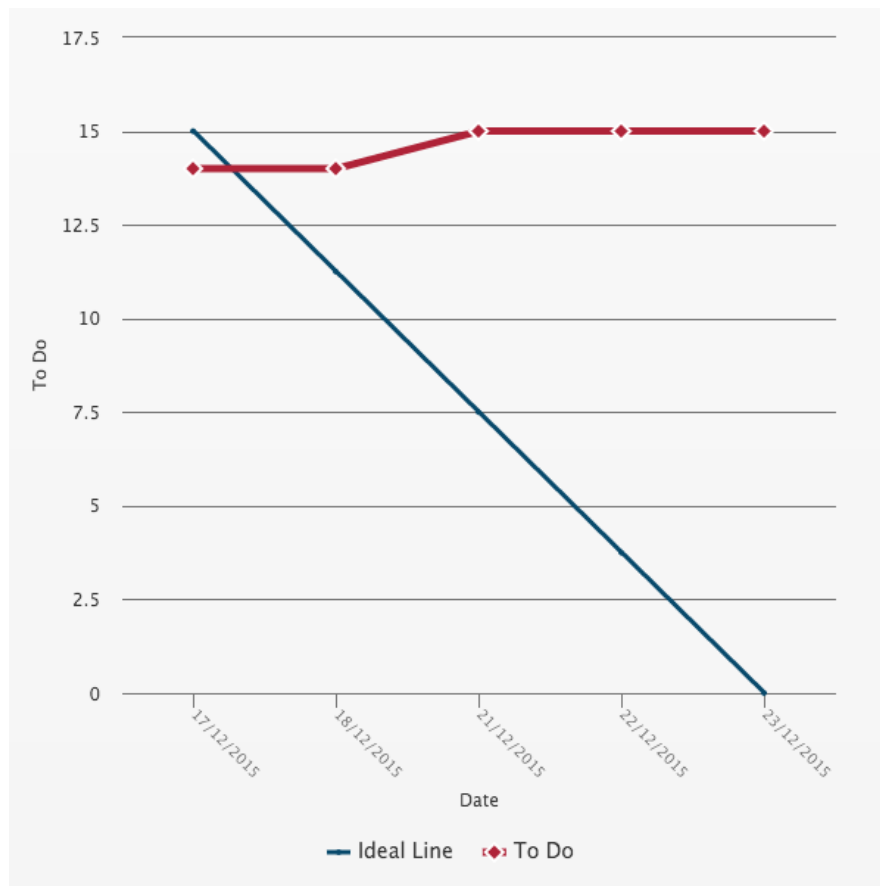


Figura A.5: Gráfico Burndown Iteración 6

**Iteración 7 [23/12/2015 - 30/12/2015]**

Esta iteración dura una semana. En ella se desarrolla tanto la vista de productos y las compras por filtro. Se añade en la ventana principal el sistema de gráficos por categoría.

Historia	Horas Estimadas	Horas Reales
Diseño de la pantalla principal	11	4
Desarrollo de pantalla de tiques	8	5

Cuadro A.7: Tareas de la séptima iteración

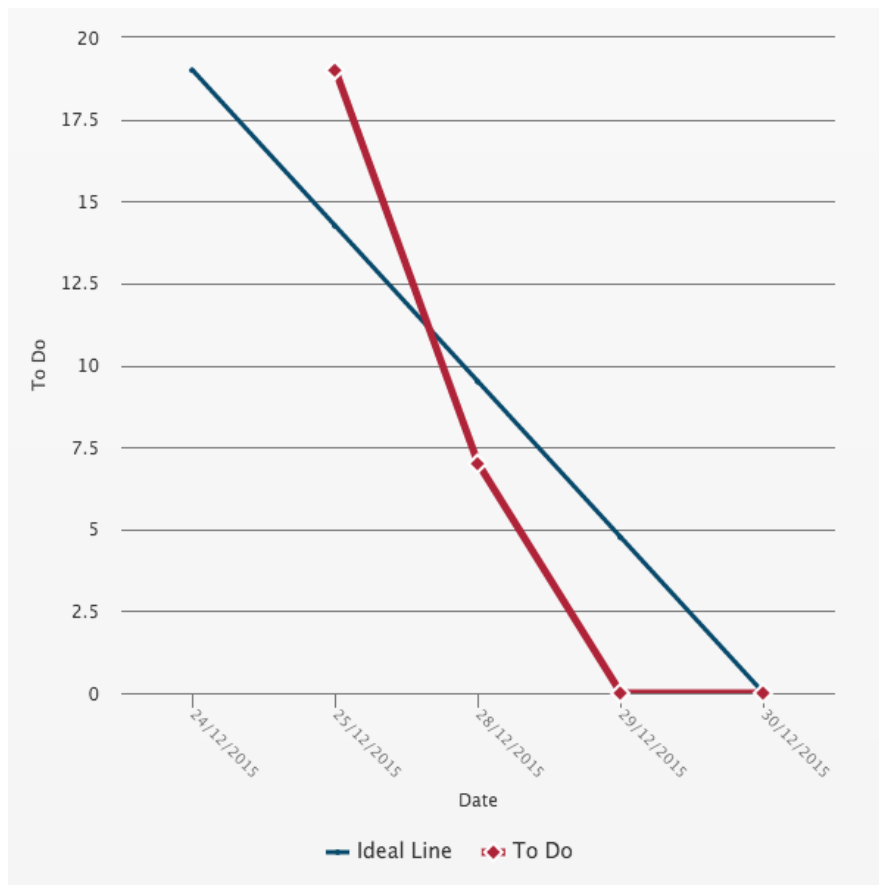


Figura A.6: Gráfico Burndown Iteración 7

**Iteración 8 [31/12/2015 - 06/01/2016]**

Esta iteración dura una semana. Esta iteración tiene poca carga de trabajo dado que se encuentra durante las fiestas de navidad , año nuevo y reyes.

Historia	Horas Estimadas	Horas Reales
Implementación vista Detalle de tique	4	3

Cuadro A.8: Tareas de la octava iteración

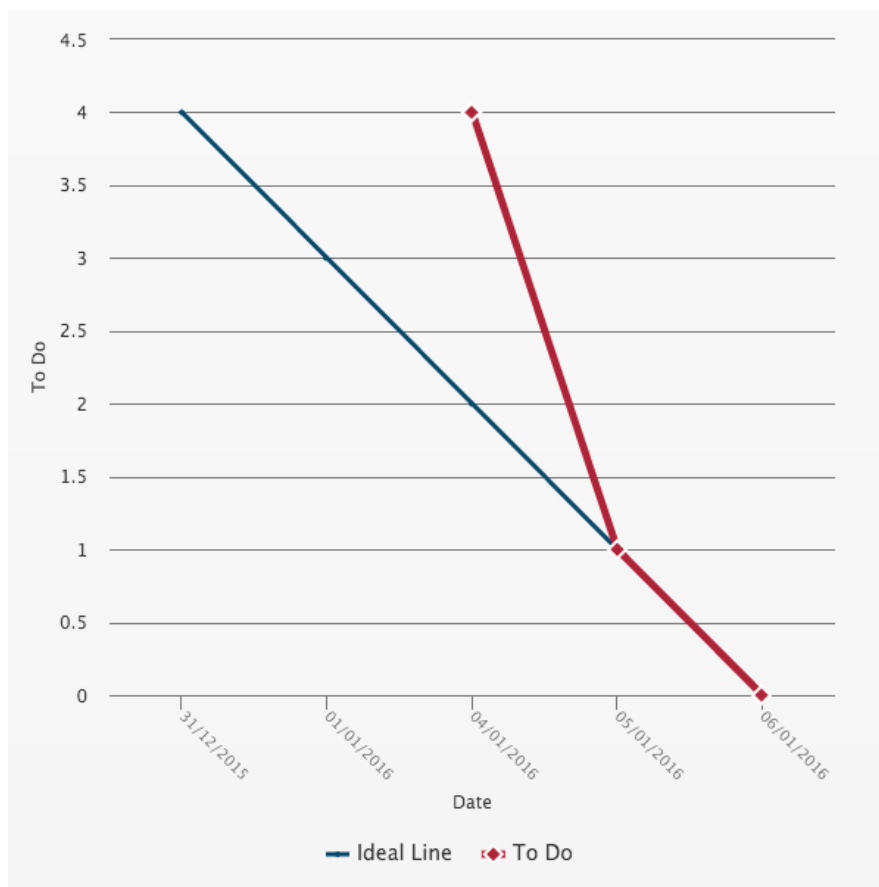


Figura A.7: Gráfico Burndown Iteración 8

**Iteración 9 [06/01/2016 - 13/01/2016]**

Esta iteración dura una semana. Esta iteración se desarrolla el envío y guardado de las correcciones del usuario en el servidor.

Historia	Horas Estimadas	Horas Reales
Envío de correcciones	8	-

Cuadro A.9: Tareas de la novena iteración

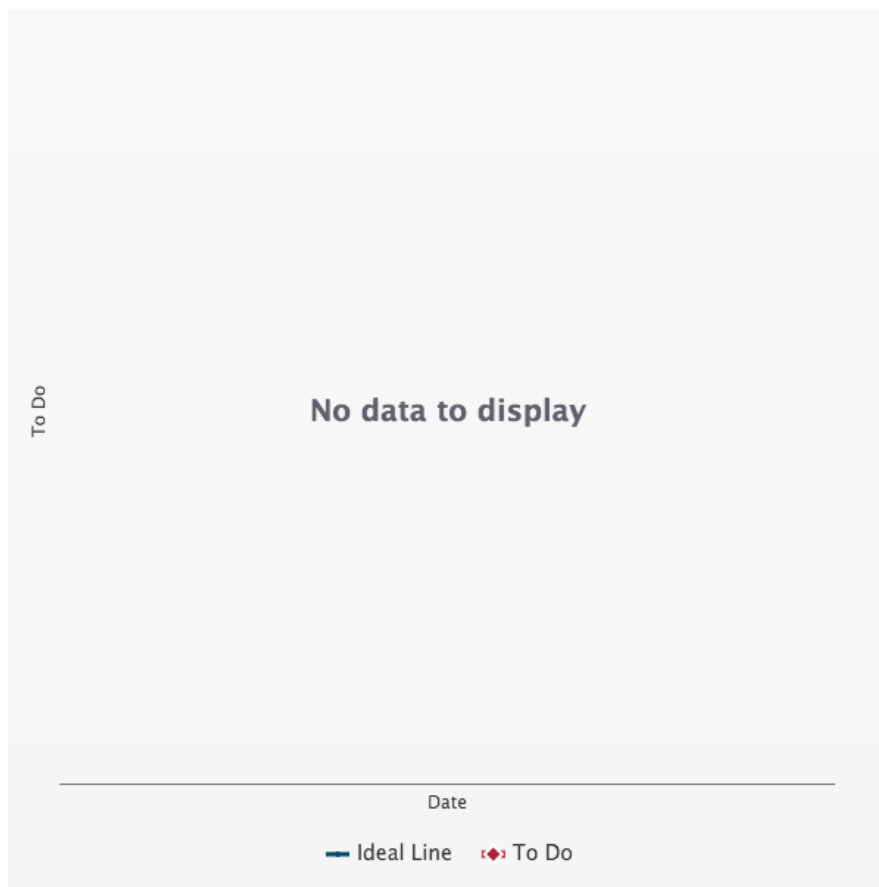


Figura A.8: Gráfico Burndown Iteración 9

### A.3. Estudio de viabilidad

En esta sección del proyecto, se comprueba si el proyecto es rentable para ello, se tendrán en cuenta varios aspectos:

- La viabilidad económica.
- La viabilidad legal.

#### Viabilidad económica

##### Costes de Personal

El presente trabajo se ha desarrollado durante 3 meses, una jornada laboral consta de 8 horas diarias de trabajo, y cada mes se trabajan de media unos 20 días, el salario que se determina para un programador en este proyecto es de 13€/hora, dado que se necesitan conocimientos que no se ven en la carrera por lo tanto coste para el personal sería de :

$$11\text{€/hora} * 8 \text{ horas/día} = 88\text{€/día}.$$

$$88\text{€/día} * 20 \text{ días/mes} = 1760\text{€/mes}.$$

$$1760 \text{ €/mes} * 3 \text{ meses} = 5280\text{€}.$$

##### Costes de Seguridad Social

La ley de Seguridad Social marca que un porcentaje de sueldos deben ser abonado al estado en concepto de impuestos, se calcula un 23,60 % por contingencias comunes, mas un 6,70 % en concepto de desempleo en contrato de duración determinada a tiempo completo mas un 0,60 % en concepto de formación profesional. Todo ello hace un total de 30,9 % [6] . Los gastos de la Seguridad Social se calcula multiplicando el salario bruto por el porcentaje del tipo de cotización.

$$5280e * 30,9 \% = 1631,52e$$

##### Costes en Hardware

Este coste viene dado por los elementos físicos necesarios para el desarrollo del proyecto, en este caso se puede agrupar en :

- Ordenador portátil : 1200€.
- Dispositivo Android: 350€.
- Ordenador para servidor: 500€.

Se ha considerado que a partir de los 4 años el hardware está obsoleto y que el coste residual es 0 por lo tanto la amortización es:

$$CA = \text{Valor de adquisición} - \text{Valor residual} / \text{Meses de vida útil}$$

$$CA = \frac{(1200e + 350e + 500e - 0)}{48\text{meses}} * 3\text{meses} = 128,125e \quad (\text{A.1})$$

### Costes en Software

Los costes software se determinan por el coste de las licencias de cada herramienta por el número de estas adquiridas. En la tabla A.10 se puede ver un desglose del precio herramienta/licencia.

Herramienta	Numero de Licencias	€/licencia	Total
Windows 7	1	120€	120€
NetBeans	1	Gratis	0€
Android Studio	1	Gratis	0€
SourceTree	1	Gratis	0€
Tesseract	1	Gratis	0€
OpenCv	1	Gratis	0€
GlassFish Free	1	Gratis	0€
Jersey	1	Gratis	0€
OrmLite	1	Gratis	0€
SimpleCropView	1	Gratis	0€

Cuadro A.10: Tabla de licencia de herramientas

El coste de amortización del software se calcula igual que el de hardware, sin embargo la vida útil del software la consideraremos como 2.

$$CA = \frac{(120e - 0)}{24\text{meses}} * 3\text{meses} = 15e \quad (\text{A.2})$$

Si sumamos todos los costes que se han calculado:

Coste	Importe(€)
Costes de Personal	5280€
Costes de Seguridad Social	1631.52€
Costes en Hardware	128.125€
Costes en Software	15€

Cuadro A.11: Tabla de costes



**Viabilidad legal**

---

## Especificación de Requisitos

---

### B.1. Introducción

El objetivo de esta sección es definir las funcionalidades de la aplicación que se está desarrollando.

### B.2. Objetivos generales

### B.3. Catalogo de requisitos

### B.4. Especificación de requisitos

#### Requisitos funcionales

- **RF1. Registrar tique:** Permite al usuario cargar una imagen, ya sea a través de la cámara del dispositivo o de la galería. Esta imagen se envía al servidor, que una vez procesada devuelve un JSON con los productos encontrados, o devuelve un código de excepción si se ha producido algún error. Los productos devueltos se deben poder editar antes de guardarse en la base de datos del dispositivo.
- **RF2. Consultar tiques:** El usuario puede realizar diferentes consultas a la base de datos en relación a los tiques que ha registrado. En ellas puede seleccionar los tiques que se encuentren en un rango de fechas, o entre un rango de precio, que viene dado por el importe total del tique.
- **RF3. Consultar Productos:** El usuario puede realizar diferentes consultas a la base de datos en relación a los productos de los tiques que ha registrado. En ellas puede seleccionar los productos que se hayan re-

gistrado entre dos fechas, el precio unitario de producto en un rango de precio y filtrar productos asociados a una categoría. Siendo estas:

- Bebidas Alcohólicas.
- Carnes.
- Comida Rápida.
- Ensaladas.
- Otros.
- Pasta.
- Pescados.
- Platos Calientes.
- Platos Combinados.
- Postres.
- Raciones.
- Refrescos.

### Requisitos no funcionales

- **RNF1. Facilidad de uso:** Se debe implementar una navegación entre las vistas sencilla e intuitiva.
- **RNF2. Extensible:** Debe estar pensado para que se pueda añadir nuevas funcionalidades, y nuevas vistas de la forma mas fácil posible.

## Especificación de diseño

---

### C.1. Introducción

### C.2. Diseño de datos

### C.3. Diseño procedimental

### C.4. Diseño arquitectónico

#### Cliente Android

En el caso del cliente para el desarrollo de la aplicación se ha elegido un patrón denominado *MVP o Model-View-Presenter*. Este patrón permite la separación de la aplicación en componentes o lo que es lo mismo separar las vistas de la lógica de la aplicación. La principal ventaja de este patrón es que permite tener la misma lógica en varias vistas totalmente distintas [3].

- **El modelo:** El modelo son los datos que se visualizan en la vista. O las clases que implementan la lógica del negocio y acceden a un sistema de datos persistente, un claro ejemplo son los DAO
- **La vista:** Es la interfaz que muestra los datos del modelo al usuario, generalmente implementada por un Activity o un Fragment, contiene una referencia a un presenter que se encargara de comunicar los eventos de la vista al modelo.
- **El Presentador:** Es el intermediario entre la vista y el modelo, se encarga de recuperar los datos del modelo y se los devuelve a la vista, por lo que tiene referencias tanto de la vista como del modelo.

Podemos ver un esquema en la imagen C.1, el modelo y la vista quedan completamente separados. Un pequeño ejemplo de como se desarrolla el ciclo de una aplicación se puede ver en la imagen C.2.

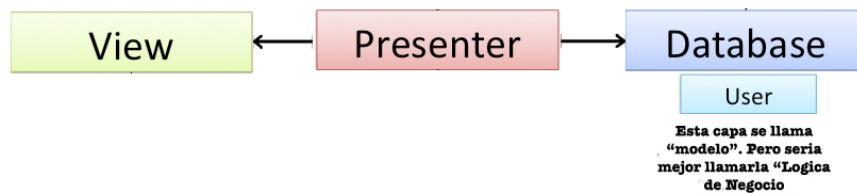


Figura C.1: Esquema de conexión MVP

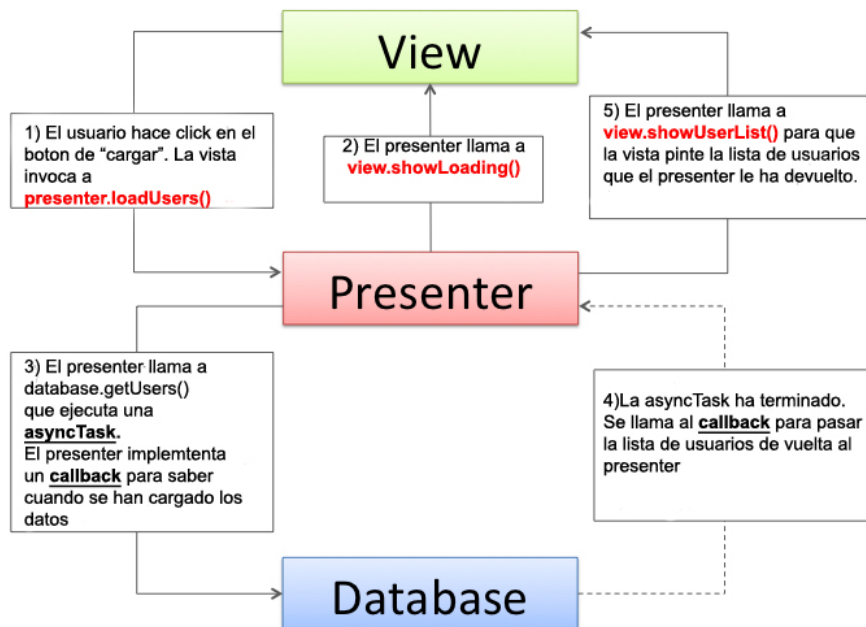


Figura C.2: Ejemplo aplicación Android con MVP

## **Documentación técnica de programación**

---

### **D.1. Introducción**

### **D.2. Estructura de directorios**

### **D.3. Manual del programador**

En esta sección se pretende explicar paso a paso como instalar, compilar y ejecutar el proyecto, como referencia a futuros programadores que usen este proyecto como base para desarrollar nuevas funcionalidades o nuevos proyectos. Esos manuales se han desarrollado en un sistema operativo Windows 7 de 64 bits pero también se puede usar la configuración de 32 bits.

### **D.4. Instalación, compilación y ejecución del proyecto**

#### **Instalación del Entorno**

##### **Java JDK 7**

Para la ejecución del proyecto, es necesario el uso de Java, por ello se debe realizar una descarga desde la web [2], eligiendo la versión del sistema operativo en la que ejecutemos el proyecto.

##### **NetBeans**

El servidor, se ha desarrollado bajo el IDE de NetBeans, para instalarlo, se debe ir a la web [4], y seleccionar la version Java EE(ver figura D.1). Una

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN<sup>19</sup>

vez descargado instalar, y asegurarse de que se selecciona la versión *GlassFish Server OpenSource Edition*, que instalará de forma automática GlassFish en nuestro ordenador.

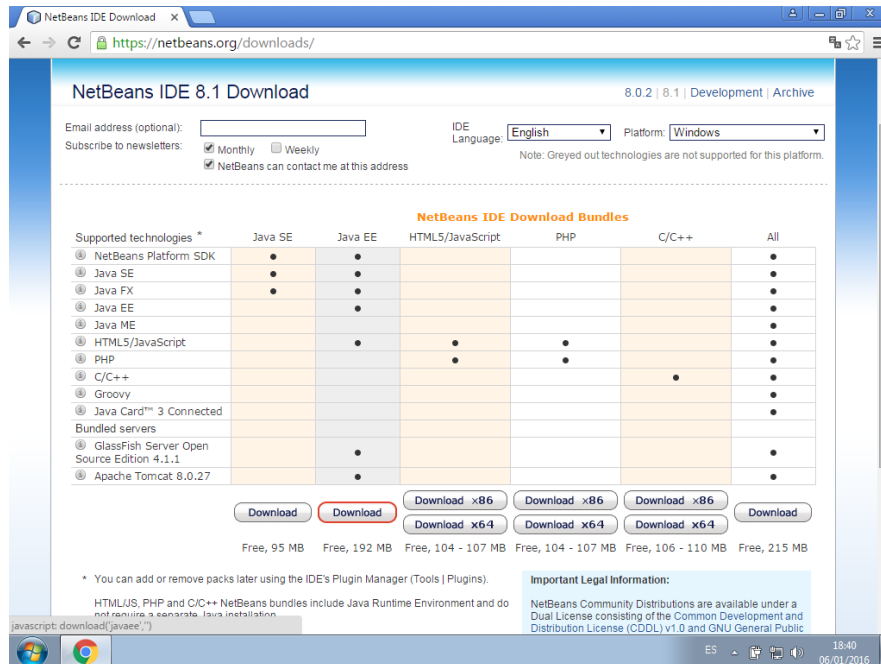


Figura D.1: Selección de la versión de NetBeans

### Tesseract

Para el uso de Tesseract en el servidor, es necesario su instalación, pero **previamente** se debe instalar el paquete de **Visual C++ para visual Studio 2013**, que se puede descargar de [5]. Se debe elegir la versión en función de la configuración del sistema operativo siendo:

- **vcredist\_x64.exe** para la versión de 64 bits.
- **vcredist\_x86.exe** para la versión de 32 bits.

Una vez instalado procedes a la descarga de Tesseract de [1] donde elegimos el archivo **tesseract-ocr-setup-3.02.02.exe** (ver figura D.2)





 <a href="#">tesseract-ocr-3.02_grc.tar.gz</a>	Ancient Greek Language data for Tesseract 3.02.02
 <a href="#">tesseract-ocr-3.02_epo_alt.tar.gz</a>	Esperanto alternative language data for Tesseract 3.02
 <a href="#">tesseract-3.02.02-win32-lib-include-dirs.zip</a>	VC++ libraries of Tesseract OCR 3.02.02 (32bit) <i>Featured</i>
 <a href="#">tesseract-ocr-setup-3.02.02.exe</a>	Windows installer of tesseract-ocr 3.02.02 (including English language data) <i>Featured</i>

Figura D.2: Selección de descarga Tesseract

## Android Studio

Para el desarrollo del cliente, se ha utilizado el IDE de Android Studio, para su instalación, se debe ir a la web [?] y elegir la versión de sistema operativo.

## D.5. 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] Download android studio and SDK tools | android developers.
- [2] Downloads - tesseract-ocr - an OCR engine that was developed at HP labs between 1985 and 1995... and now at google. - google project hosting.
- [3] Java SE development kit 7 - downloads | oracle technology network | oracle.
- [4] MVP en android: cómo organizar la capa de presentación.
- [5] NetBeans IDE download.
- [6] Paquetes redistribuibles de visual c++ para visual studio 2013.
- [7] Seguridad social:trabajadores.