Trabajo de Final de Grado

Grado en Ingeniería en Tecnologías Industriales

Software para el procesamiento estadístico de los resultados académicos de la ETSEIB

MEMORIA

Autor: Alicia Guindulain Lebrero

Director: Lluís Solano Albajes

Convocatoria: Setiembre 2015

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resumen

En el presente proyecto se diseña y desarrolla una aplicación de escritorio que, a partir de unos datos, muestra un análisis descriptivo de los resultados académicos de los alumnos de la ETSEIB. Concretamente, se estudian las titulaciones de GETI, GEM y GEQ.

Con tal fin, en primer lugar se hizo una evaluación de antecedentes y se definió el entorno de desarrollo del software. Seguidamente, se implementó el código encargado de realizar la preparación y procesamiento de los datos. Se planteó y diseñó la estructura general de la aplicación y finalmente, se desarrolló el software.

Por un lado, la aplicación ofrece un análisis de las combinaciones de asignaturas que los alumnos suspenden en la fase inicial o selectiva, es decir, en el primer curso del grado. Por otro lado, se realiza una descripción estadística de los resultados académicos en todas las asignaturas para cada cuatrimestre, curso académico y titulación. El usuario puede realizar un filtrado de los alumnos que se estudian en este último análisis. Ambos estudios se pueden exponer en el tiempo para visualizar su evolución.

Se ha programado con el lenguaje de programación Python3.4. Para la preparación y procesamiento de los datos se ha empleado la librería de programación Pandas. Para implementar la interfaz gráfica de usuario, es decir, el entorno visual y comunicación de la aplicación, se ha utilizado principalmente la librería gráfica PyQt4.

El resultado que se obtiene es un software diseñado para funcionar en un sistema operativo OS X y fácil de implementar para otros sistemas operativos. La interfaz gráfica es elegante e intuitiva y los resultados del análisis descriptivo claros e inteligibles. Además, contiene un manual de usuario e información para garantizar la comprensión del funcionamiento y contenidos.

Se concluye que, para garantizar la futura usabilidad de la aplicación, sería recomendable ampliar su funcionalidad y continuar el proyecto mediante terceras personas.

Sumario

1. Glosario

1.1. Abreviaturas y acrónimos

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

ETSEIB: Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (Escuela Técnica Susperior de Ingeniería Industrial de Barcelona).

GEM: Grau en Enginyeria en Materials (Grado en Ingeniería de Materiales).

GEQ: Grau en Enginyeria Química (Grado en Ingeniería Química).

GETI: Grau en Enginyeria en Tecnologies Industrials (Grado en Ingeniería en Tecnologías Industriales).

GUI: Graphic User Interface (Interfaz Gráfica de Usuario).

IO: Input/Output (Entrada/Salida)

Mac: Macintosh.

POO: Programación Orientada a Objetos

UI: User Interface (Interfaz de Usuario).

1.2. Definiciones

Barplot: Diagrama o gráfico de barras. En lugar de utilizar puntos disconexos o unidos por líneas para representar los datos se emplean barras. Ampliamente utilizados cuando se quiere presentar estudios sobre un número pequeño de clases.

Biblioteca (programación): Conjunto de implementaciones funcionales que ofrece una interfaz bien definida para su uso. Habitualmente se emplea el término librería para referirse a una biblioteca.

Binding (programación): Adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

Boxplot: Diagrama o gráfico de cajas. Suministra información sobre los valores mínimo y máximo, los cuartiles Q1 (25%), Q2 (50%) o mediana y Q3 (75%), y sobre la existencia de valores atípicos y la simetría de la distribución.

Clase (programación): En programación orientada a objetos, una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Dicho modelo define un conjunto de variables (estado) y métodos apropiados para operar con dichos datos (comportamiento).

Diálogo (programación): Término utilizado a lo largo del trabajo para referirse a la ventana (*widget*) con la que interactúa el usuario final.

Diccionario (programación): En Python, un diccionario es una colección no-ordenada de valores que son accedidos a través de una clave.

Fase selectiva: Esta fase está constituida por las 5 asignaturas del cuatrimestre 1 y las 5 asignaturas del cuatrimestre 2 de las titulaciones GETI, GEM y GEQ. La normativa de la ETSEIB confiere un período de 2 años, con opción a cuatrimestre de gracia, para completar dicha fase.

Framework (programación): Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Heredar (programación): En programación orientada a objetos, mecanismo a través del cual es posible crear nuevas clases partiendo de una clase o de una jerarquía de clases preexistente, extendiendo así su funcionalidad. La clase de la que se hereda suele denominarse clase padre.

Histograma: Gráfica que muestra la frecuencia de los datos, en la que el eje horizontal representa unidades discretas, ciertos rangos, o intervalos, en tanto que el eje vertical representa la frecuencia. Frecuentemente, se dibujan barras rectangulares con sus áreas proporcionales a las frecuencias dentro de los rangos o de los intervalos.

Interfaz de usuario: Medio por el que el usuario es capaz de comunicarse con una

máquina, equipo o computador que comprende todos los puntos de contacto entre el usuario y el equipo.

Interfaz gráfica de usuario (GUI): Programa informático que actúa de interfaz de usuario utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Proporciona un entorno visual sencillo para permitir la comunicación con el sistema operativo o con el ordenador.

Librería (programación): Conjunto de implementaciones funcionales que ofrece una interfaz bien definida para su uso.

Macintosh: La línea de computadoras personales diseñada, desarrollada y comercializada por Apple Inc.

Objeto (programación): En programación orientada a objetos, es el resultado de instanciar una clase. Consta de un estado y de un comportamiento y adquiere la capacidad de interactuar con otros objetos.

Pieplot: Diagrama o gráfico de sectores. Se trata de un círculo dividido en diversos sectores que se usan para representar frecuencias de las diferentes clases mostradas dentro de un conjunto de datos.

Programación orientada a objetos (POO): Es un paradigma de programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.

Scatterplot: Diagrama o gráfico de dispersión. Los datos se disponen como puntos en una gráfica para mostrar una relación posible entre dos conjuntos de datos o variables.

Script: Lista de instrucciones o órdenes, escritas utilizando un lenguaje informático.

Shell: También conocido como terminal, intérprete de órdenes o intérprete de comandos. Es el programa informático que provee una interfaz de usuario para acceder a los servicios del sistema operativo. Permite evaluar el código y mostrar los resultados de forma interactiva, confiriendo un feedback inmediato.

Software: Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

Toolkit: Conjunto de programas, instrucciones y rutinas que se utilizan como base para facilitar la tarea de programación.

Widget: En el contexto de la programación de aplicaciones de escritorio, un widget o gadget es un elemento de interacción en una GUI. De él emanan la mayoría de componentes gráficos y su objetivo es dar acceso a funcionalidades y proveer información visual actualizada.

2. Prefacio

2.1. Origen del proyecto

El proyecto de desarrollo de software para el procesamiento estadístico de los resultados académicos de la Escuela Técnica Superior de Ingeniería Industrial de Barcelona (a partir de ahora ETSEIB) aparece en Proyecto I, asignatura de segundo curso del Grado en Ingeniería en Tecnologías Industriales (a partir de ahora GETI), del Grado en Ingeniería de Materiales (a partir de ahora GEM) y del Grado en Ingeniería Química (a partir de ahora GEQ). El extenso abanico de posibilidades que engloba dicho procesamiento hizo que el departamento LSI (Lenguajes y Sistemas de Información) de la escuela lo propusiera más adelante en la bolsa de trabajos de final de grado de la ETSEIB.

Partiendo de aquí, la idea de la funcionalidad que se pretende conseguir con el programa surgió a sugerencia del tutor de este proyecto y tras un análisis de antecedentes (de un programa desarrollado en Proyecto I).

2.2. Motivación

La principal motivación de este proyecto es el reto de crear una aplicación. El interés nace a sabiendas de la indiscutible y exponencial importancia del mundo del software y la informática en la sociedad de tecnologías actual.

Otro aliciente es el beneficio que puede suponer, tanto para la docencia como para el estudiante, el disponer de un programa informático que genere de forma automática diferentes análisis descriptivos de los resultados académicos de la escuela. Además, la existencia de una base de datos extensa, gestionada por el Servicio PRISMA de la Universidad Politécnica de Cataluña (UPC), elimina la necesidad de recoger los datos y favorece una mayor dedicación a las tareas de interés: diseño del software y análisis estadístico.

Por un lado, como estímulo se une el ser capaz de desarrollar un proyecto en su totalidad de forma óptima, resolutiva, analítica y siguiendo siempre un método iterativo que garantice la calidad y cumplimiento de los objetivos. El hecho de poder ver e interactuar con la solución del proyecto, el programa, es un factor que ayuda a involucrarse más con el

objeto del trabajo. Además, se facilita la tarea de comprobar si los objetivos se han conseguido.

Por otro lado, está el desafío de fusionar de forma práctica los conocimientos y competencias adquiridas a lo largo de la titulación. Paralelamente, ampliar dichos conocimientos de forma autodidacta como herramienta necesaria para conseguir un objetivo concreto.

2.3. Requerimientos previos

Para la realización de este proyecto se ha necesitado de conocimientos básicos y competencias en informática y programación, fundamentos de estadística, organización y gestión industrial, gestión de proyectos, impacto ambiental y normativas.

En cuanto al desarrollo de la aplicación, ha sido indispensable el uso eficiente de los sistemas de información y el conocimiento de una tercera lengua, el inglés.

3. Introducción

3.1. Objetivos del proyecto

Se presentan en este apartado del estudio los objetivos que se pretende conseguir, que se dividirán en dos grupos:

- El objetivo principal, que es el fundamental y razón de ser del proyecto.
- Los objetivos específicos y funcionales, que describen el procesamiento que se desea realizar y los requisitos que ha de cumplir la aplicación. También obedecen a las expectativas y necesidades de los usuarios razón de ser y operadores. Se define como usuario razón de ser a todo aquel que se beneficie de los servicios y funcionalidad que ofrecerá el programa: personal docente, alumnos y terceros. En cambio, se define como usuario operador aquel que se encargue del correcto funcionamiento y posibles modificaciones futuras del código interno del programa.

3.1.1. Objetivo principal

El objetivo principal de este trabajo es la creación de una aplicación de escritorio que realice y muestre un análisis descriptivo desde varios puntos de vista de los resultados académicos de los estudiantes de GETI, GEM y GEQ de la ETSEIB.

3.1.2. Objetivos específicos y funcionales

- El análisis ha de contener una evaluación de las combinaciones de asignaturas suspendidas en la fase selectiva. Por fase selectiva o fase inicial se entiende el primer curso de los tres grados impartidos en la escuela.
- El software debe proporcionar una descripción del rendimiento académico de los alumnos en todas las asignaturas del GETI, GEM y GEQ por separado.
- Se ha de poder visualizar la evolución temporal de los resultados que se obtengan en ambos estudios.
- Se ha de garantizar la comprensión de los resultados del análisis descriptivo.

- El programa debe mantener su funcionalidad al introducir nuevos datos o frente a cambios en el plan de estudios de las titulaciones.

- La aplicación ha de poseer una interfaz de usuario agradable e intuitiva. El aspecto visual de la misma deberá adecuarse al contexto en que se enmarca y a las expectativas del usuario razón de ser.

- Minimizar el tiempo de ejecución y funcionamiento de la aplicación.

- El proyecto ha de poder ser continuado en un futuro para garantizar la futura utilidad de la aplicación.

3.2. Alcance del proyecto

Este proyecto cubre, por un lado, la descripción del proceso, módulos y herramientas que se han empleado para obtener los resultados que mostrará la aplicación. Por otro lado, cubre todo el proceso de concepción, diseño, creación y programación de la aplicación. También abarca la elaboración de manuales de uso. No incluye la incorporación de la misma al mercado de aplicaciones.

3.3. Metodología

El contenido de este informe en las próximas secciones se elabora, a grandes rasgos, teniendo en cuenta el orden cronológico en que se ha desarrollado el proyecto. Se han contemplado 6 fases desde su nacimiento hasta la obtención del programa final. Cada una de ellas consta de diferentes partes, también expuestas en orden temporal:

Fase 1.- Se realiza un estudio preliminar o análisis de antecedentes.

- Documentación sobre programas ya existentes elaborados en la asignatura de Proyecto I de la ETSEIB. Análisis de la funcionalidad existente y aspectos no cubiertos.

Fase 2.- Se define el entorno de desarrollo.

- Documentación sobre los posibles sistemas operativos con los que trabajar, lenguajes de programación que utilizar y entornos de desarrollo integrado para la implementación del código del programa. Selección del entorno de trabajo.

Fase 3.- Se realizan las tareas necesarias para la preparación y procesamiento de los datos que permitan obtener la estadística descriptiva que efectuará la aplicación.

- Descripción y análisis del contenido de los ficheros de entrada de la base de datos del Servicio PRISMA. Elaboración de ficheros adicionales para la aplicación.

- Documentación sobre las librerías y módulos necesarios. En informática, una librería o biblioteca es un conjunto de subprogramas implementados por terceros y utilizados para desarrollar software. Introducen funcionalidades de interés al programador.

- Preparación. Creación de funciones predefinidas para estructurar los datos adecuadamente según el análisis que quiera realizarse en cada caso.

- Procesamiento. Creación de funciones para obtener y graficar los resultados del análisis descriptivo.

Fase 4.- Se realizan las tareas necesarias para obtener el programa, es decir, consta del diseño y desarrollo de la aplicación.

- Diseño gráfico de la Interfaz Gráfica de Usuario, a partir de ahora GUI, y diseño conceptual de funcionamiento.

- Documentación sobre las librerías y módulos necesarios para la elaboración de una aplicación de escritorio.

- Desarrollo del código del programa.

Fase 5.- Se realizan las tareas necesarias para facilitar el uso de la aplicación al usuario razón de ser y para dirigir la futura continuidad y mejora del proyecto al usuario operador.

- Redacción del manual de usuario.

- Descripción de las posibles vías de continuación.

Fase 6.- Se exponen los estudios realizados para planificar y evaluar el proyecto.

- Planificación de actividades realizada.

- Evaluación de costes derivados del desarrollo del proyecto

- Análisis del impacto ambiental y social al entorno durante y posterior a la realización del proyecto.

4. Antecedentes

Como se ha comentado anteriormente, este proyecto surge como continuación a la asignatura de Proyecto I de la ETSEIB. Es por tanto que se considera oportuno iniciar el proyecto con un análisis previo de la funcionalidad que ofrecen dichos programas. Ello permitirá identificar las áreas no cubiertas en cuanto al análisis estadístico de los resultados académicos de la escuela y justificará la necesidad de implementar un nuevo software para cubrir dichas áreas.

4.1. The Hag

The Hag es una aplicación desarrollada por un grupo de alumnos en el cuatrimestre de primavera del curso académico de 2014-2015. A continuación se realiza un estudio de su interfaz de usuario y funcionalidad; es decir, no se analizará en cuanto a su estructura interna o código. De este estudio se extrae una lista de aspectos positivos, aspectos mejorables y funcionalidad conseguida que será considerada a lo largo de la elaboración del software objeto del proyecto.

Cabe mencionar que se ha escogido *The Hag* como antecedente de análisis a recomendación del tutor de este proyecto.

4.1.1. Descripción

Al ejecutar la aplicación aparece una ventana donde se introducen dos ficheros de datos que corresponden a los datos personales de los alumnos y los resultados académicos de los mismos. Una vez seleccionados, se ejecuta la ventana principal del programa donde se introduce el filtro principal de los datos que se desea analizar; ésta puede verse en la **¡Error! No se encuentra el origen de la referencia.**Fig. 4.1. Existen 3 opciones a realizar dentro de la estadística descriptiva que ofrece *The Hag:*

1. *Rankings:* Se muestra mediante una tabla todos los datos extraídos de los ficheros de entrada. Éstos se pueden exponer en orden creciente o decreciente en función de diversas variables, que son las columnas de la propia base de datos.

2. *Posición personal:* Se selecciona el código del estudiante y las asignaturas deseadas a analizar. Se muestra en formato texto el percentil que ocupa dicho

alumno dentro de cada asignatura. Existe la opción de visualizar un gráfico de Radar (o diagrama de araña) donde aparecen las notas finales en las asignaturas seleccionadas del alumno de interés junto con las obtenidas por el “alumno media”.

3. Gráficos: Se puede obtener diagramas de barras 2D y 3D, Boxplots, Histogramas y Scatterplots 2D y 3D de las notas finales o de selectividad de los alumnos. Se puede seleccionar cualquier categoría que se desee estudiar en el gráfico dentro de las diferentes variables que incluya la base de datos.

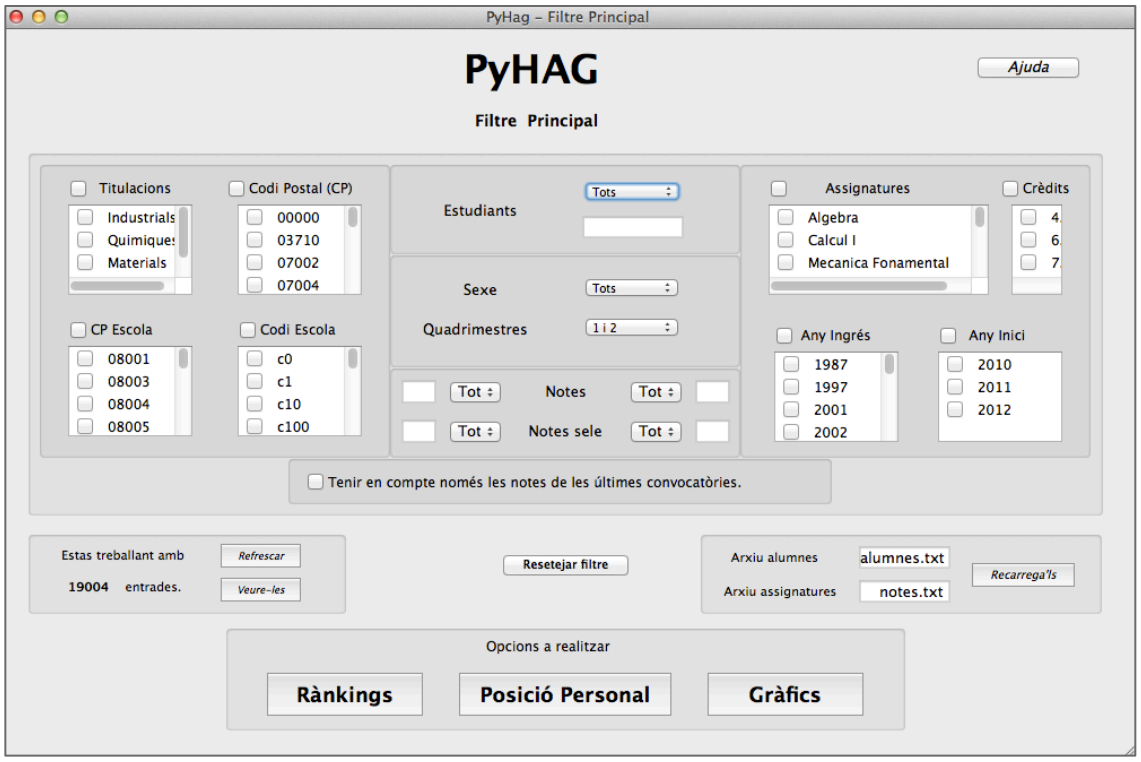


Fig. 4.1. Ventana principal del programa *The Hag.*

4.1.2. Aspectos positivos

Tras experimentar con el programa y evaluar las prestaciones que ofrece, se desprenden los siguientes puntos fuertes:

- Versatilidad. El filtro principal permite seleccionar una muestra de estudiantes específica para cualquier grupo dentro de las diferentes variables y categorías que incluya la base de datos. Entre ellas el sexo, créditos de asignaturas, código postal del alumno o de la escuela, año de ingreso, año de inicio, nota de selectividad, etc.

- Diversidad. Hay múltiples tipos de gráficos con los que representar los resultados académicos de los alumnos.

- Fluidez. El programa es rápido en ejecutarse y no se colapsa o bloquea durante su uso.

4.1.3. Aspectos negativos

El análisis permite extraer también un conjunto de puntos débiles o posibles aspectos a mejorar:

- Necesidad de uso de la terminal. Tanto la ejecución de *The Hag* como la visualización de los gráficos se realizan a través del shell del ordenador. Este aspecto puede resultar negativo para un usuario no acostumbrado al empleo del intérprete de comandos. Por terminal, shell, intérprete de órdenes o intérprete de comandos se entiende el programa informático que provee una interfaz de usuario para acceder a los servicios del sistema operativo.

- Limitación en cuanto al curso académico de análisis. A priori, la aplicación solo trata con las asignaturas del primer curso de las titulaciones de la ETSEIB. Este hecho solo puede modificarse editando el código interno del programa.

- Defectos en las tablas de Rankings. Por ejemplo, la aparición de columnas en formato float (decimal) cuando debería tratarse de números enteros, entre otros.

- Escasa información o ayuda. El programa está diseñado para un usuario razón de ser que conoce la base de datos de partida. Explícitamente, no se detalla con claridad el significado de las siglas que se atribuyen a las columnas de las tablas de datos o las categorías de diferentes variables. Como ejemplo, el código de la escuela es una variable que va desde c0 hasta c385. Podría incluirse un manual al que acudir para identificar el significado de cada grupo. Este aspecto no es primordial, pero podría tenerse en cuenta para un usuario tipo alumno o externo.

4.1.4. Funcionalidad

En concreto, *The Hag* proporciona las herramientas necesarias para visualizar los resultados académicos (notas finales y percentiles) de diferentes grupos de alumnos, en función del filtro que se aplique, y todo ello mediante eficientes gráficas y/o salidas de texto.

Dicho filtro considera cualquier combinación de categorías posible, lo cual resulta un potente instrumento de inspección y análisis. Estas competencias no se considerarán para el desarrollo de la aplicación de este trabajo, pues se considera que ya se han cubierto eficazmente.

No obstante, The Hag no ofrece un análisis de las combinaciones o agrupaciones de asignaturas más suspendidas. Además, la descripción estadística que realiza se reduce a las asignaturas del primer curso y los gráficos que se obtienen no muestran en ningún caso la evolución temporal de los resultados. Estos aspectos deberán remediarse en el programa objeto del proyecto.

5. Entorno de desarrollo

5.1. Sistema Operativo

El Sistema Operativo, a partir de ahora OS, o software del sistema es el conjunto de programas informáticos que permite la administración eficaz de los recursos de una computadora. Estos programas comienzan a trabajar apenas se enciende el equipo, ya que gestionan el hardware desde los niveles más básicos y permiten además la interacción con el usuario. Se entiende por hardware al conjunto de los componentes que forman la parte material (física) del ordenador.

El OS con el que se ha trabajado es **OS X**, también conocido antiguamente como Mac OS X. Está basado en Unix, desarrollado, comercializado y vendido por Apple Inc. y especialmente diseñado para ser utilizado en computadoras Macintosh.

Hoy en día las computadoras vienen, básicamente, con tres alternativas de OS: Windows, OS X y Linux. A nivel de participación de mercado, Windows fue el más popular y extendido en la gran mayoría de computadoras. Tan es así, que un estudio realizado en 2009 estimó que el 80% de las computadoras del mundo utilizaban una u otra versión del mismo. Respecto a OS X y Linux, aquel estudio afirmó que se mantenían en aproximadamente un 11% y un 4%, respectivamente, de las computadoras vendidas en el mundo. [1]

No obstante, OS X ha crecido espectacularmente gracias a un constante flujo de innovación y actualizaciones que impactan al mercado en general. De hecho, un estudio realizado en 2014 por *VMWare* acerca del uso de ordenadores Mac en las empresas a nivel mundial reveló que los usuarios de equipos informáticos prefieren, de una manera abrumadora, el uso de los equipos de Apple frente a PCs que operan bajo Windows. La encuesta muestra que un 71% de las empresas son capaces de soportar el uso y administración de equipos Mac, y el 66% de éstas posee empleados haciendo uso de estos equipos en sus entornos de trabajo. [2] El hecho de que haya un porcentaje tan elevado de usuarios que quieran hacer uso de equipos Mac en la propia empresa y con fines laborales no solo refleja el gran peso que han ido adquiriendo los productos de Apple en el mercado; sino que parece ser que la tendencia muestra que el dominio de Windows está acercándose a su fin. Teniendo estos resultados en consideración, se creyó oportuno elegir este OS

como entorno de trabajo.

Adicionalmente, la ventaja estrella que ofrece trabajar con un Macintosh es que existe la posibilidad de instalar otros OS en la misma computadora. Ello se puede realizar mediante un software de virtualización o realizando una partición del disco duro. De este modo y en el caso de que fuera necesario, se puede integrar en una misma computadora toda la funcionalidad que ofrecen los diferentes OS.

5.2. Lenguaje de programación

Una vez escogido el entorno operativo de trabajo, hace falta elegir el lenguaje de programación con el que se desarrollará el código del programa. Se ha programado con Python por prescripción del director de este proyecto. Sin embargo, se exponen a continuación las características por las que resulta idóneo.

Existe una inmensa variedad de lenguajes para satisfacer distintas necesidades. Aunque muchos de ellos se pueden utilizar en diferentes ámbitos, suele haber algún lenguaje que destaque entre los demás para dicha área. Por ejemplo, para programar páginas web destacan HTML, CSS, JavaScript y PHP, mientras que para desarrollar aplicaciones móviles lo hace Objective-C para iOS o Java para Android.[3]

En el caso de creación de aplicaciones de escritorio, por cada sistema operativo pueden existir diferentes lenguajes específicos que sólo servirán para dicha plataforma. Sin embargo, uno de los objetivos es garantizar la futura usabilidad el programa. Dicha usabilidad comprende el que se pueda utilizar en diferentes sistemas operativos, con lo que se ha optado por desarrollar con un lenguaje multiplataforma.

En primer lugar, Python destaca por su sencillez, legibilidad y precisión de sintaxis. En pocas líneas permite programar algoritmos complejos y es práctico en cuanto al uso del código. A diferencia de lenguajes como C++ o Java la sintaxis de Python es bastante similar al lenguaje natural y por lo tanto mucho más intuitivo y fácil de aprender.

En segundo lugar, este lenguaje resulta muy versátil. Puede servir para aplicaciones web, interfaces de usuario, análisis de datos, estadísticas, etc. Esto es así gracias al elevado número de librerías y frameworks que lo complementan. Por framework se entiende una estructura conceptual y tecnológica de soporte definido,

normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. [4]

En tercer lugar, Python permite generar programas multiplataforma que pueden ser ejecutados en cualquier sistema en que pueda instalarse un interprete Python (Windows, Mac Linux, FreeBSD, OpenSolaris, etc). Además, es gratuito y de código abierto.

En cuarto lugar, se valora que Phyton cuenta con una comunidad muy activa, capaz de aportar tutoriales y respuestas a problemas concretos que facilitarán el aprendizaje autodidacta del uso de las bibliotecas y frameworks que se requieran para el desarrollo del software objeto del proyecto.

Finalmente, ha sido decisivo el hecho de que la ejecutora de este proyecto posee nociones básicas de este lenguaje de programación. Aprender uno diferente desde cero implica un valioso tiempo que dejaría de emplearse para el análisis estadístico y la creación de la GUI.

Se ha programado con la versión Python 3.4, que es la más reciente.

5.3. Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación como son: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Para una aplicación Python se necesita un IDE en el cual se pueda programar en dicho lenguaje. [5]

En este proyecto se utiliza un entorno de desarrollo llamado Spyder. Este IDE, antiguamente llamado Pydee, tiene capacidad multiplataforma y está diseñado para la programación científica. Integra librerías Python útiles para la investigación científica como por ejemplo: NumPy y SciPy para operaciones numéricas, Matplotlib para obtención de gráficos o Pandas para el análisis de datos [6]. Dispone de consolas Python y IPython integradas. La última de ellas añade funcionalidades extra al modo interactivo incluido con Python, como resaltado de líneas y errores mediante colores, una sintaxis adicional para el shell, autocompletado mediante tabulador de variables, módulos y atributos; entre otras funcionalidades.

Se basa en la plataforma Qt indirectamente a través de las librerías PyQt o PySide y está disponible para Python 3.4.

Se ha utilizado la versión Spyder 2.3.5.2, que es la más reciente.

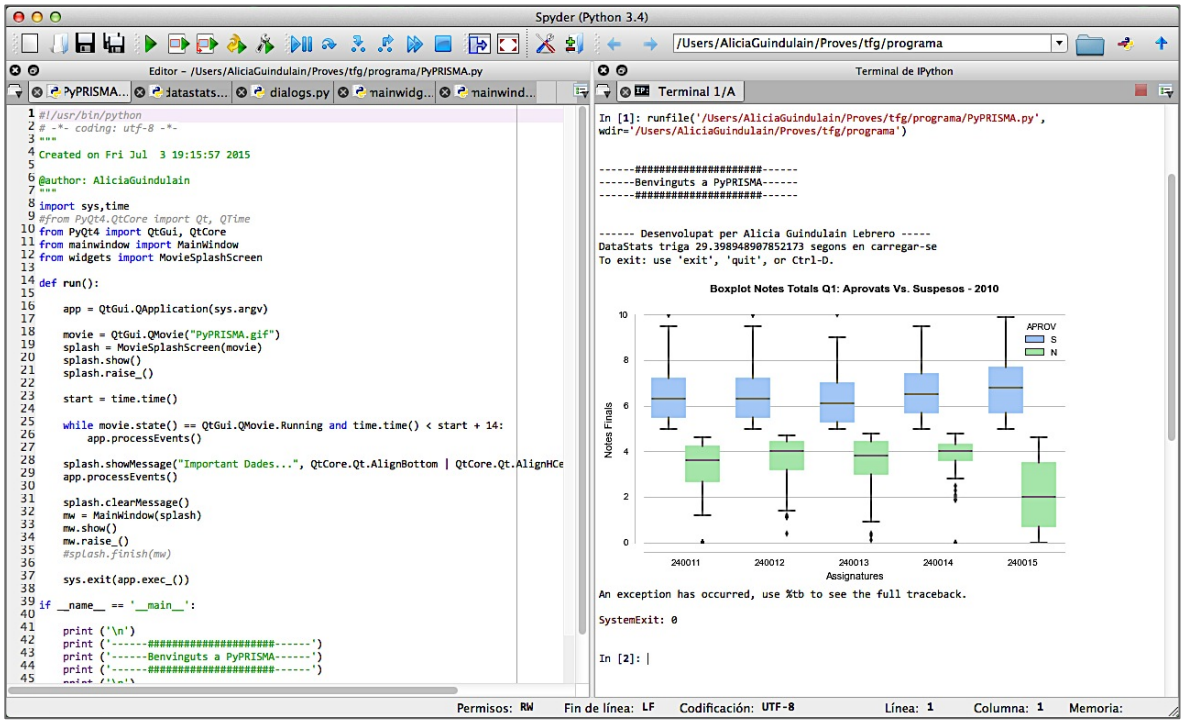


Fig. 5.1. Entorno de Desarrollo Integrado (IDE) – Spyder.

6. Preparación y procesamiento de datos

Esta sección corresponde a la tercera fase del proyecto. En ella se realizan las tareas necesarias para obtener los resultados del estudio estadístico en la aplicación.

6.1. Base de datos

6.1.1. Servicio PRISMA

Servicio PRISMA es un servicio del área de docencia que da soporte a la gestión de los estudios de la UPC, mediante el desarrollo de sistemas de información, y proporciona atención e información a sus usuarios. También se encarga de la realización de estudios estadísticos de docencia y de la elaboración de los indicadores necesarios para la toma de decisiones. [7]

Asimismo, el Servicio PRISMA se encarga de almacenar los datos personales y académicos de todos los alumnos de las distintas escuelas que pertenecen a la UPC. Es gracias a ello que se dispone de la base de datos correspondiente a la ETSEIB, requerida para el desarrollo de este proyecto. Los ficheros de datos de partida se describen a continuación.

6.1.2. Ficheros de datos

La aplicación se nutre de seis ficheros. Cabe discernir entre los ficheros de datos que se obtienen a través del Servicio PRISMA y aquellos que se han creado específicamente para la aplicación. Si se desea se puede consultar el anexo A, donde aparece un extracto de cada uno de ellos que muestra el formato y contenido de dichos ficheros.

6.1.2.1. Ficheros de partida

Se parte de dos ficheros de datos principales, obtenidos desde la base de datos del Servicio PRISMA.

6.1.2.1.1 Fichero de datos personales:

Contiene información referente a los alumnos registrados en la ETSEIB. Cada fila corresponde a un alumno distinto y se divide en ocho celdas o columnas:

Código_Expediente ; Sexo ; CP_Familiar ; Año_Acceso ; Vía_Acceso ; Nota_Acceso ; Centro_Secundaria ; CP_Centro_Secundaria

- *Código_Expediente:* Código que identifica al alumno.
- *Sexo:* Si el alumno es hombre o mujer.
- *CP_Familiar:* Código postal del domicilio del estudiante.
- *Año_Acceso:* Año de acceso a la escuela.
- *Vía_Acceso:* Código referente al tipo o vía de acceso a la escuela. Por ejemplo, desde bachillerato junto con las PAU (Pruebas de Acceso a la Universidad), desde CFGS (Ciclos Formativos de Grado Superior), cambio de universidad, convalidación de estudios parciales hechos en el extranjero, etc.
- *Nota_Acceso:* Nota de acceso a la escuela del alumno.
- *Centro_Secundaria:* Código que identifica el centro de secundaria del estudiante.
- *CP_Centro_Secundaria:* Código postal del centro de secundaria del estudiante.

La existencia de alumnos repetidos en este fichero ha forzado a realizar un filtro en que se guarda únicamente los datos personales del alumno que haya entrado a la escuela más tarde.

6.1.2.1.2 Fichero de resultados/calificaciones:

Contiene información referente a todas las convocatorias a las que se han presentado los alumnos de la ETSEIB, tanto fase inicial como fase no inicial. Cada fila corresponde a una convocatoria distinta y se divide en once columnas o celdas:

Código_Programa ; Código_Expediente ; Código_Asignatura ; Créditos ; Curso ; Cuatrimestre ; Supera ; Nota_Prof ; Nota_Num_Eval ; Nota_Num_Def ; Grupo_Clase

- *Código_Programa:* Código que identifica la titulación o programa al que corresponde la convocatoria.

- *Código_Expediente:* Código que identifica al individuo o alumno al que corresponde la convocatoria.
- *Código_Asignatura:* Código UPC de la asignatura a la que corresponde la convocatoria.
- *Créditos:* Número de créditos de la asignatura convocada.
- *Curso:* Curso académico en que se realiza la convocatoria.
- *Cuatrimestre:* Cuatrimestre en que tiene lugar la convocatoria.
- *Supera:* Si se aprueba o suspende la convocatoria.
- *Nota_Prof:* Nota o evaluación del profesor, de 0 a 10.
- *Nota_Num_Eval:* Nota curricular, de 0 a 10.
- *Nota_Num_Def:* Nota definitiva de la convocatoria, de 0 a 10.
- *Grupo_Clase:* Grupo o clase al que está matriculado el alumno en la convocatoria. Aquellos alumnos que convaliden una asignatura desde otro plan de estudios se identifican también en esta variable.

6.1.2.2. Ficheros adicionales

Como complemento a la información que se desprende de los ficheros de partida y para hacer más cómoda la lectura e interpretación de los resultados que muestra la aplicación, se ha creado cuatro ficheros adicionales con información referente a las asignaturas de la fase selectiva y de las titulaciones GETI, GEM y GEQ:

6.1.2.2.1 Fichero de asignaturas de la fase inicial

Contiene información referente a las asignaturas que forman el primer curso del GETI, del GEM y del GEQ. La fase selectiva o inicial es común para los tres grados, con lo que estas asignaturas se unifican y agrupan en un solo fichero. Cada fila corresponde a una asignatura distinta y se divide en tres columnas:

Código_Asignatura ; Nombre_Asignatura ; Cuatrimestre

- *Código_Asignatura:* Código UPC de la asignatura.
- *Nombre_Asignatura:* Nombre explícito de la asignatura.
- *Cuatrimestre:* Cuatrimestre al que pertenece la asignatura respecto al plan de estudios del grado.

6.1.2.2.2 Ficheros de asignaturas del GETI, GEM y GEQ

Contienen información referente a las asignaturas que forman las titulaciones del GETI, del GEM y del GEQ por separado. Cada fila corresponde a una asignatura distinta y se divide en cuatro columnas:

Código_Asignatura ; Nombre_Asignatura ; Cuatrimestre ; Tipo

- *Código_Asignatura:* Código UPC de la asignatura.
- *Nombre_Asignatura:* Nombre explícito de la asignatura.
- *Cuatrimestre:* Cuatrimestre al que pertenece la asignatura respecto al plan de estudios del grado.
- *Tipo:* Si es obligatoria o optativa.

Uno podría preguntarse acerca de la necesidad del fichero específico para la fase selectiva, teniendo disponible toda la información del plan de estudios de las titulaciones GETI, GEM y GEQ en los tres últimos ficheros. Esto se ha hecho para garantizar la correcta unificación de los tres grados en el estudio de la fase inicial. Actualmente en la titulación GEQ, las asignaturas de química I y II se codifican de diferente manera. Como que este hecho se conoce, es fácil de tratar. No obstante, no se conocen las posibles modificaciones del plan de estudios que pueda surgir en un futuro. Para intentar que el funcionamiento y fiabilidad de la aplicación permanezcan intactos, se toma como referencia los códigos del primer fichero y se comparan con los tres últimos, consiguiendo unificar los datos de las convocatorias de estas asignaturas.

6.2. Librerías

Para poder explorar y trabajar con los datos, principalmente ha sido necesario el uso

de una biblioteca de programación, la librería Pandas.

6.2.1. Pandas

Pandas es una librería en Python para el análisis de datos publicada bajo la licencia BSD-3 (Berkeley Software Distribution 3-Clause License).

Se ha decidido trabajar con este paquete de herramientas debido a que su funcionalidad se ajusta con las necesidades del tipo de análisis de datos que se quiere realizar. Pandas proporciona un alto rendimiento en cuanto a la manipulación de tablas de datos. Ofrece estructuras variadas, rápidas y flexibles, diseñadas para facilitar el trabajo de agregar, combinar y redistribuir conjuntos de datos. Existe la posibilidad de indexación jerárquica, que resulta muy útil para clasificarlos y agrupar variables de interés. Además, posee herramientas IO (Input/Output) robustas para cargar datos de archivos de texto (CSV y delimitados), archivos de Excel, bases de datos, y guardarlos o descargarlos desde el formato ultrarrápido HDF5. Por último, también permite obtener fácilmente gráficos con un diseño sencillo e intuitivo. [9]

Existe una alternativa a Pandas, Orange, que no se ha escogido ya que su comunidad no es tan activa y la documentación está menos desarrollada.

La versión que se ha utilizado es: Pandas 0.16.2.

6.3. La clase *DataStats*

La estructura interna de la aplicación del proyecto se divide en clases. En Programación Orientada a Objetos, a partir de ahora POO, una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Dicho modelo define un conjunto de variables (el estado) y métodos apropiados para operar con dichos datos (el comportamiento). Se entiende por métodos a las subrutinas de manipulación de los datos de la clase.

La clase *DataStats* representa la entidad que contiene los datos procedentes de los ficheros de entrada, los métodos para manipularlos y las estructuras de datos resultado de dicha manipulación. El código de la misma aparece adjunta en el anexo B.

6.3.1. Preparación de datos

Por un lado, la puesta a punto de los datos engloba la lectura de los ficheros de entrada y su transformación en un formato adecuado para trabajar con ellos. La estructura principal empleada para contenerlos es el DataFrame, que se explicará a continuación.

Por otro lado, incluye la limpieza de los datos para asegurar su coherencia y la creación de métodos que permitan obtener nuevos subconjuntos de datos apropiados para el estudio que se quiera realizar en cada caso.

6.3.1.1. La estructura DataFrame

La estructura Pandas que se ha utilizado principalmente para trabajar con los datos es el DataFrame. Se trata de una estructura bidimensional de datos etiquetados con columnas potencialmente de diferentes tipos: texto, números enteros o decimales y otros. Se puede imaginar como una hoja de cálculo o tabla de SQL. La naturaleza de los datos almacenados en los ficheros de partida también es bidimensional, lo cual hace de los DataFrames la estructura idónea para contenerlos.

De los seis ficheros de partida se obtendrán los seis DataFrames principales de la clase *DataStats*. Numerosos métodos que incorpora esta clase se encargarán de jugar con los DataFrames principales para dar cabida a otros subconjuntos de datos, agrupados mediante esta misma estructura.

6.3.1.2. Limpieza de datos

Antes de utilizar los datos para el estudio, se ha realiza una etapa de inspección de los mismos para descubrir y corregir o eliminar datos erróneos.

En los ficheros de datos se descubrieron las siguientes inconsistencias:

- En el fichero de datos personales, un mismo alumno (código de identificación del alumno) aparece múltiples veces pero con diferentes valores de sexo, vía de ingreso a la escuela, código postal, etc.

- Un alumno tiene resultados en convocatorias anteriores al año en que accede a la escuela.

- Un alumno tiene resultados en convocatorias que infringen el curso natural del

grado. Por ejemplo, un alumno entra a la escuela un año específico y en el año siguiente tiene resultados de asignaturas que corresponden al tercer o cuarto curso del grado. Este hecho se debe al cambio de plan de algunos alumnos que se cambian de la antigua titulación superior al grado.

Las correcciones que se toman para cada caso son las siguientes:

- Para los alumnos repetidos, sólo se considera el alumno que accede la escuela más tarde. En caso de empate, se escoge al individuo del que se conozcan más datos.

- Se modifica el año de acceso a la escuela al año en que se conoce el primer resultado académico del alumno.

- Un alumno tiene resultados en convocatorias que infringen el curso natural del grado. Por ejemplo, un alumno entra a la escuela un año específico y en el año siguiente tiene resultados de asignaturas que corresponden al tercer o cuarto curso del grado. Este hecho se debe al cambio de plan de algunos alumnos que se cambian de la antigua titulación superior al grado.

Por otro lado, también se tiene alumnos y convocatorias con entradas incompletas; sobretodo alumnos. Como el número de éstas es elevado, se ha decidido conservar estos datos y tratarlos cuidadosamente más adelante en el estudio.

6.3.1.3. Subconjunto fase selectiva

El estudio para la fase selectiva implica hacer un seguimiento de un bloque específico de estudiantes. El método *alumnes_interes* se encarga de identificar y agrupar a los alumnos que pertenecen a dicho bloque. Se ha realizado un estrecho filtro en la selección de estos alumnos para asegurar la coherencia y representatividad de los resultados que se obtengan. A continuación se exponen los requisitos que cumplen éstos alumnos:

- El alumno pertenece a una titulación de Grado, es decir, titulación ya adaptada al Proceso de Bolonia. No se incluyen los alumnos que se incorporan desde el plan antiguo de estudios.

- El alumno comienza la fase selectiva en un curso académico específico y cursa todas las asignaturas por primera vez.

- El alumno se matricula a todas las asignaturas del primer y segundo cuatrimestre. En otras palabras, no se consideran alumnos que abandonan el grado a mitad del primer curso.

- El alumno de nuevo acceso no convalida asignaturas, sino que cursa todas ellas.

6.3.1.4. Subconjunto global

De ahora en adelante, el estudio del rendimiento académico en todas las asignaturas se denotará por estudio *global.* Para éste no se realiza ningún filtro específico, ya que se proporcionará una ventana en la aplicación para que el usuario escoja los filtros que él desee aplicar. Las opciones a escoger para el filtro son:

- Código de identificación del/de los alumnos.

- Código postal del/de los alumnos.

- Código postal del centro de secundaria.

- Código de identificación del centro de secundaria.

- Vía de acceso de los estudiantes que se analicen.

- Año de acceso.

- Sexo. Estudiar solo hombres, mujeres o ambos.

- Intervalo de notas de selectividad.

- Excluir convocatorias convalidadas.

- Excluir convocatorias no presentadas.

- Evaluar únicamente alumnos que hayan suspendido al menos una asignatura.

Los métodos encargados de filtrar los datos correspondientes a los ficheros *personal.txt* y *notes.txt* son *filtre_alumnes* y *filtre_notes* , respectivamente.

6.3.2. Procesamiento de datos

El procesamiento de datos engloba la creación de funciones para obtener la descripción estadística que se quiera realizar en cada caso. Como se explica en los

objetivos del proyecto, la aplicación tendrá que cubrir dos estudios por separado. En primer lugar, el referente a la fase selectiva; y en segundo lugar, la descripción estadística de los resultados académicos en las asignaturas de las titulaciones GETI, GEM y GEQ por separado

El tipo de procesamiento de datos que requiere cada estudio es completamente diferente; es por ello que esta sección se divide en dos apartados, respectivamente.

6.3.2.1. Procesamiento fase selectiva

Ya se ha comentado que se quiere proporcionar información visual sobre las combinaciones de asignaturas que son suspendidas por los alumnos en la fase selectiva.

Este estudio se realiza sobre cada bloque de alumnos definido por el método *alumnes_interes* y el método que lo realiza es *flux*. Esta función emplea un mecanismo iterativo que evalúa a cada alumno dentro del bloque por separado y lo sitúa dentro de la combinación específica de asignaturas suspendidas que éste cumpla. Esta clasificación se realiza a modo de conteos, es decir, no se guarda al alumno en sí; sino que se quiere saber el número de estudiantes que cumplen dicha combinación.

La estructura empleada para almacenar esta información es un diccionario de formato predefinido. En Python, un diccionario es una colección no-ordenada de valores que son accedidos a través de una clave. La figura que muestra el diseño del formato de dicho diccionario puede encontrarse en el anexo A.

Este diccionario se ha creado de modo que cada clave corresponde al número de asignaturas suspendidas. A su vez, se tiene un diccionario por clave donde se almacenan las distintas posibilidades de asignaturas suspendidas. El número de posibilidades que incluye cada clave corresponde a los coeficientes del binomio de Newton para una potencia de n=5, siendo la potencia el número de asignaturas que hay en cada cuatrimestre. Esto es lo mismo que la sexta fila del triángulo de Tartaglia o Pascal. Aún así, para demostrar tal hecho se explica el procedimiento de cálculo mediante combinaciones:

Se llama combinaciones de m elementos tomados de n en n ($m \geq n$) a todas las agrupaciones posibles que pueden hacerse con los m elementos de forma que:

- No entran todos los elementos.
- No importa el orden.

- No se repiten los elementos.

Estas combinaciones se denotan por C_m^n y pueden calcularse mediante factoriales, como se indica en la ecuación 6.1.

$$C_m^n = \frac{m!}{n!(m-n)!} \qquad (Ec.\ 6.1)$$

La tabla siguiente muestra las combinaciones que existen según el número de asignaturas suspendidas para las 5 asignaturas que hay en cada cuatrimestre. Los resultados son el número de elementos que cada clave del diccionario principal contiene en su diccionario interno.

Combinaciones de asignaturas suspendidas		
Nº de asignaturas suspendidas	**Cálculo**	**Nº de combinaciones**
0	$C_5^0 = \frac{5!}{0!\,(5-0)!}$	1
1	$C_5^1 = \frac{5!}{1!\,(5-1)!}$	5
2	$C_5^2 = \frac{5!}{2!\,(5-2)!}$	10
3	$C_5^3 = \frac{5!}{3!\,(5-3)!}$	10
4	$C_5^4 = \frac{5!}{4!\,(5-4)!}$	5
5	$C_5^5 = \frac{5!}{5!\,(5-5)!}$	1

Tabla 6.1. Cálculo de combinaciones de asignaturas suspendidas según número suspensos.

Es importante mencionar que hay un diccionario para el primer cuatrimestre y otro para el segundo cuatrimestre, y no se evalúa el mismo grupo de estudiantes. Esto se ha hecho así porque la combinatoria que supone estudiar rigurosamente la evolución de los alumnos supone diccionario de elevado tamaño y complejidad y un tiempo de cálculo del flujo aún mayor.

Esta decisión no afecta al estudio del flujo del primer cuatrimestre, ya que los alumnos que se están estudiando se han matriculado precisamente a 5 asignaturas y éstas son las del primer cuatrimestre únicamente. No obstante, este método sí afecta al estudio de la evolución estudiantil una vez se inicia el cuatrimestre de primavera de la fase selectiva. Hay que tener en cuenta que los alumnos toman múltiples rutas, matriculándose a las asignaturas suspendidas del primer cuatrimestre y seleccionando las que desean cursar del segundo. Esto lleva a la necesidad de crear un diccionario como el principal para todas las combinaciones de matrícula de todas las combinaciones de asignaturas suspendidas en el primer cuatrimestre, es decir, 252 diccionarios. Y este resultado es con el supuesto de que únicamente pueden matricularse a cinco asignaturas, ni más ni menos.

Para comprender la complejidad que se desarrolla al pretender abarcar tal estudio se expone el siguiente ejemplo:

“ Un alumno suspende Fundamentos de Informática el cuatrimestre de otoño. En la matrícula de primavera, se ha de matricular nuevamente a Fundamentos de Informática y tiene que escoger cuatro de las cinco asignaturas del segundo cuatrimestre. Éste alumno tiene 5 posibilidades de matrícula. Una vez se sabe las asignaturas a las que se ha inscrito para el cuatrimestre de primavera, entonces se mira cual de las posibles combinaciones de asignaturas suspendidas y aprobadas en ese nuevo cuatrimestre es la que sigue el alumno, es decir, aplicar el método *flux*. Estas 5 posibilidades de matrícula existen para un alumno que ha suspendido Álgebra o Cálculo I o Química I o Mecánica Fundamental y son diferentes entre sí. “

Es por esto que el grupo de estudiantes que se evalúan para el cuatrimestre de primavera no es el mismo que el bloque de alumnos de partida para dicha fase inicial. Sino que de nuevo se seleccionan solamente aquellos alumnos que se hayan matriculado en el cuatrimestre de primavera a las cinco asignaturas del segundo cuatrimestre de la fase selectiva. Esto equivale a dejar atrás a aquellos alumnos que han suspendido más de una asignatura; ya que el número de créditos a matricular excedería lo permitido.

Si no se hiciese tal selección, los resultados para el segundo cuatrimestre serían engañosos, ya que el número de combinaciones posibles de los estudiantes que no se han matriculado a esas 5 asignaturas no es el que refleja el diccionario predefinido. Con todo, el número de alumnos que incluirá el diccionario para el cuatrimestre de primavera será estrictamente inferior que el diccionario de otoño; resultado que no deberá extrañarnos conocidos los criterios empleados para su construcción.

Una vez se tiene el diccionario correspondiente a cada bloque de alumnos (para cada fase inicial o curso académico) y correspondiente al cuatrimestre de otoño o primavera, esta información contenida en el diccionario se transforma a un DataFrame mediante el método *crear_df_flux.* La Tabla 6.2 es un extracto del DataFrame correspondiente al primer cuatrimestre de la fase selectiva del curso académico 2010-2011. Éste puede verse en su totalidad en el anexo A.

DataFrame Fase Selectiva Otoño 2010-2011			
Nom	**Nº Susp.**	**Susp.**	**Total**
Cap	0		216
Càlcul I.	1	[240012]	4
Mecànica Fonamental.	1	[240013]	15
Química I.	1	[240014]	4
Àlgebra Lineal.	1	[240011]	22
Fonaments d'Informàtica.	1	[240015]	54
Càlcul I, Mecànica Fonamental.	2	[240012, 240013]	4
Càlcul I, Química I.	2	[240012, 240014]	2
Química I, Fonaments d'Informàtica.	2	[240014, 240015]	9
Mecànica Fonamental, Fonaments d'Informàtica.	2	[240013, 240015]	20
Àlgebra Lineal, Càlcul I.	2	[240011, 240012]	14
Càlcul I, Fonaments d'Informàtica.	2	[240012, 240015]	8
Àlgebra Lineal, Fonaments d'Informàtica.	2	[240011, 240015]	15

Tabla 6.2. DataFrame flujo fase selectiva – Primer cuatrimestre – Curso 2010 - 2011.

Estos resultados deberán mostrarse en la aplicación de algún modo. Se ha decidido realizar una exposición en formato texto y otra mediante recursos gráficos.

En primer lugar, para obtener una salida tipo texto se ha creado la función *crear_txt*.

Este método parte directamente del diccionario para mostrar los resultados. El fichero resultante puede verse también en el anexo A.

En segundo lugar, para graficar el rendimiento y la evolución académica se ha decidido crear 8 gráficos distintos. Todos ellos deberán cumplir la condición de simplicidad y fácil comprensión, ya que constituye uno de los objetivos del proyecto. La creación y exposición explícita de los gráficos no se realiza en la clase *DataStats*, sino que se lleva a cabo en el desarrollo de la interfaz gráfica. No obstante, *DataStats* deberá proporcionar los datos ya listos para poder ser graficados. Es por ello que a continuación se definen los gráficos y citan los métodos que encargarán de proporcionar la información que requieran. Para facilitar la comprensión del resultado que se obtiene con cada gráfico, se muestra también el gráfico que expone la aplicación en cada caso. Se incide en que la creación de los mismos no se implementa en *DataStats*, sino que se explicará más adelante en el informe.

Gráfico 1.- *Notes Mitges per Assignatura: Global*

- Descripción: Se representa la distribución de las notas definitivas de los alumnos para cada asignatura. Permite comparar gráficamente el rendimiento de los alumnos entre asignaturas.

- Datos: Los datos corresponden a las notas finales de aquellos alumnos que cursan por primera vez la asignatura en cuestión. En otras palabras, sólo considera la primera convocatoria. El grupo de alumnos es diferente para cada año y procede del bloque *alumnes_interes*.

- Tipo de gráfico: Boxplot (Diagrama o gráfico de cajas). Permite ver la mediana, percentiles, dispersión y valores extremos y atípicos de la distribución de las notas finales para cada asignatura.

- Método: *notes_mitges(curso , cuatrimestre)* prepara los datos para cada curso académico y cuatrimestre de la fase inicial.

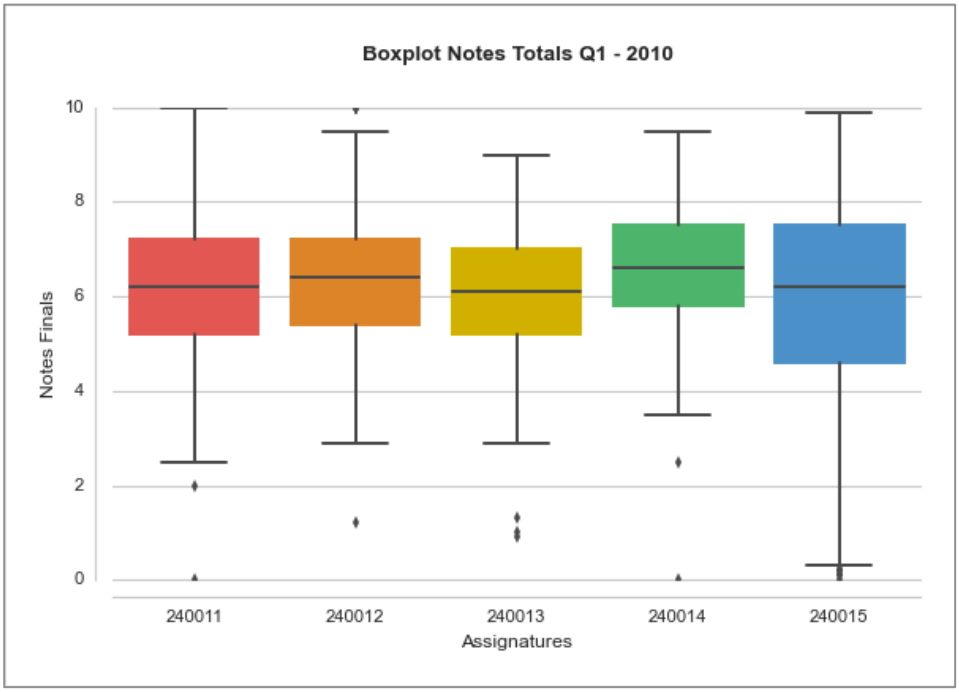


Fig. 6.1. Gráfico 1 – *Notes Mitjes per Assignatura: Global* - 2010 - Q1.

Gráfico 2.- *Notes Mitges per Assignatura: Aprovats vs. Suspesos*

- Descripción: Se representa la distribución de las notas definitivas de los alumnos para cada asignatura, separando los alumnos que aprueban la convocatoria de los que no. Permite comparar gráficamente el rendimiento de los alumnos que aprueban y de los que suspenden entre asignaturas por separado.

- Datos: Los datos corresponden a las notas finales de aquellos alumnos que cursan por primera vez la asignatura en cuestión. El grupo de alumnos es diferente para cada año y procede del bloque *alumnes_interes*.

- Tipo de gráfico: Boxplot estratificado. Igual que el Boxplot pero permite separar los datos es función de una variable categórica o niveles.

- Método: *notes_mitges_aprov(curso , cuatrimestre)* prepara los datos para cada curso académico y cuatrimestre de la fase inicial.

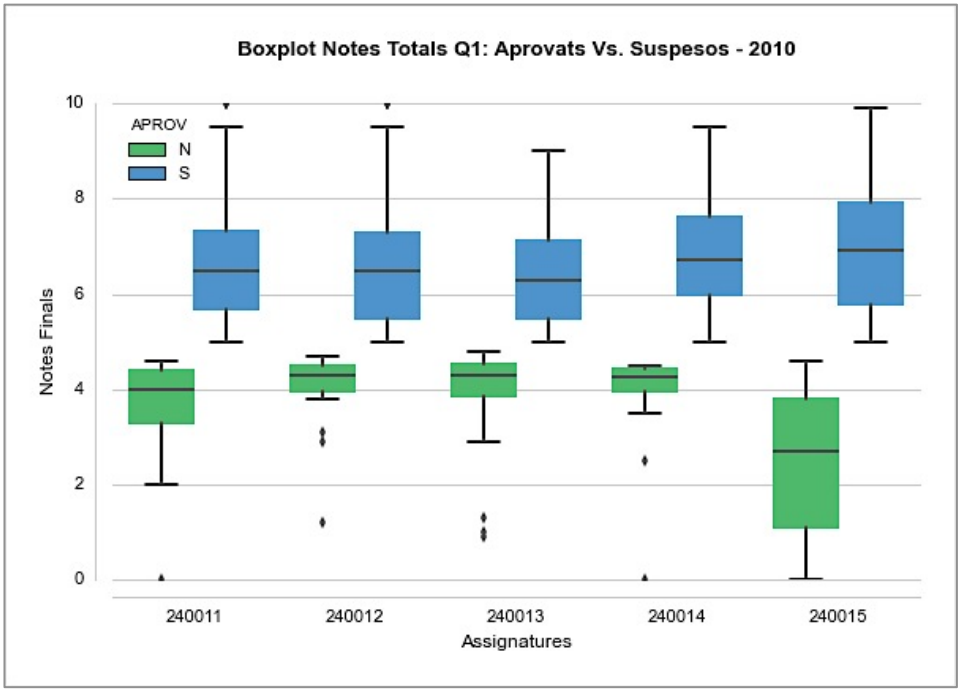


Fig. 6.2. Gráfico 2 – *Notes Mitjes per Assignatura: Aprovats Vs. Suspesos* - 2010 - Q1.

Gráfico 3.- *Total Suspesos per Assignatura*

- Descripción: Seleccionando el número de asignaturas suspendidas, se muestra el número de alumnos que han suspendido cada combinación específica de asignaturas.

- Datos: Los datos corresponden al número de suspensos de aquellos alumnos que cursan por primera vez la asignatura en cuestión y proceden del DataFrame del flujo de la fase selectiva creado mediante la función *crear_df_flux.*

- Tipo de gráfico: Barplot (Diagrama o gráfico de barras). Para representar los datos se emplean barras.

- Método: *crear_histograma(curso , cuatrimestre , numero_suspendidas)* prepara los datos para cada curso académico, cuatrimestre de la fase inicial y número de asignaturas suspendidas.

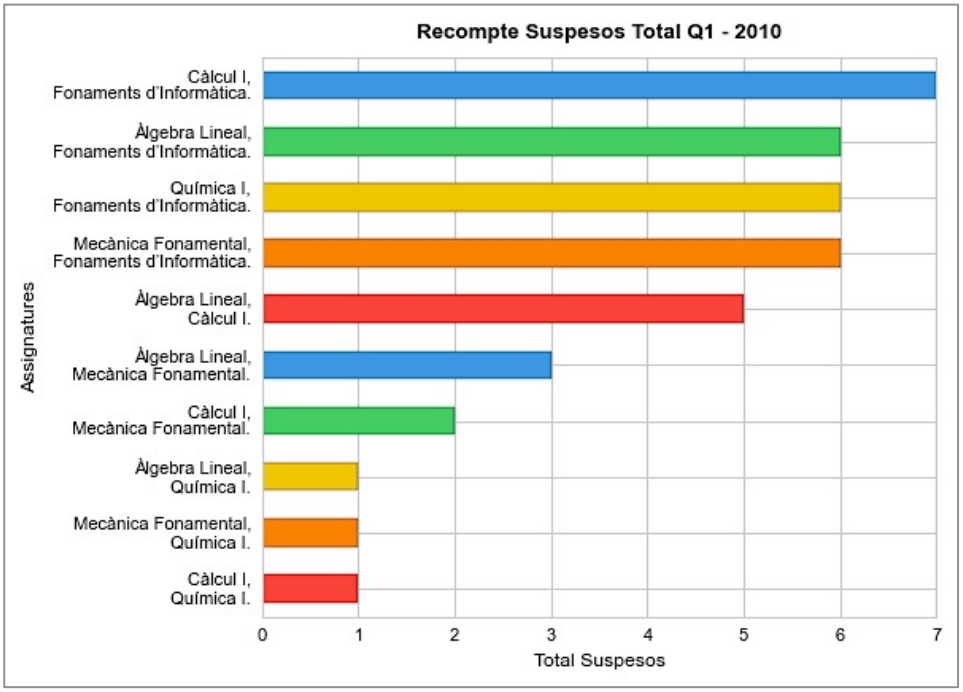


Fig. 6.3. Gráfico 3 – *Total Suspesos per Assignatura* - 2010 - Q1 - 2 suspensos.

Gráfico 4.- *Percentatge Suspesos per Assignatura*

- Descripción: Se muestra el porcentaje de suspensos de cada asignatura respecto el número total de suspensos. Nótese que no se trata de porcentajes respecto al total de alumnos.

- Datos: Los datos corresponden a los alumnos que han suspendido una o más asignaturas para cada cuatrimestre de la fase selectiva. El grupo de alumnos es diferente para cada año y procede del bloque *alumnes_interes*.

- Tipo de gráfico: Pieplot (Diagrama o gráfico de sectores). Se trata de un círculo dividido en diversos sectores que se usan para representar frecuencias de las diferentes clases mostradas dentro de un conjunto de datos.

- Método: *mes_suspesa(curso , cuatrimestre)* prepara los datos para cada curso académico y cuatrimestre de la fase inicial.

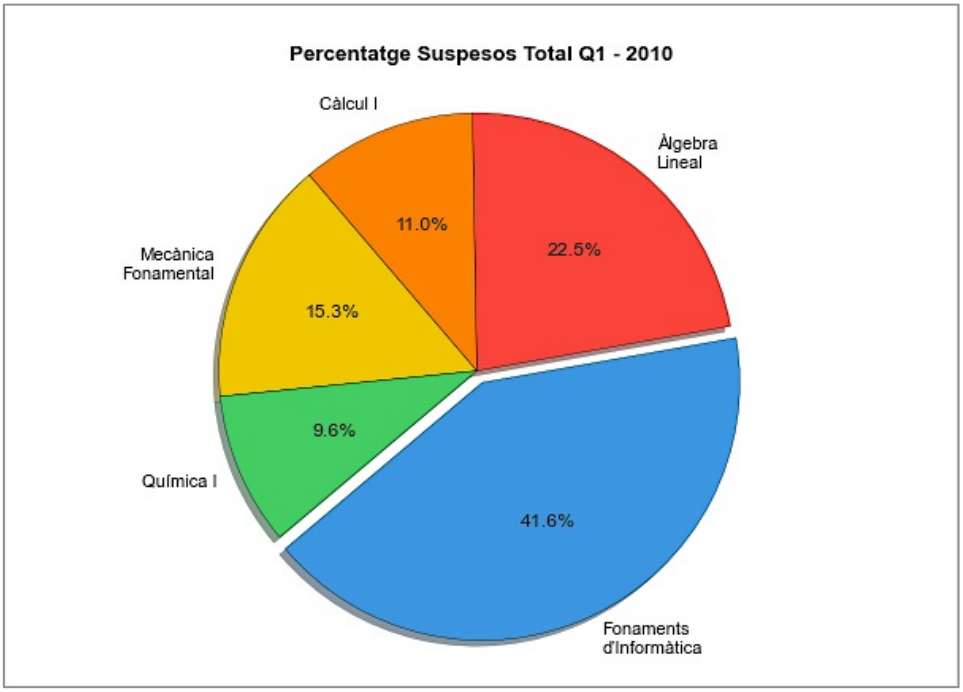


Fig. 6.4. Gráfico 4 – *Percentatge Suspesos per Assignatura* - 2010 - Q1.

Gráfico 5.- *Percentatge Suspesos per Nombre d'Assignatures*

- Descripción: Se muestra el porcentaje de alumnos que han suspendido cada número de asignaturas posible, desde 0 asignaturas suspendidas hasta el número de asignaturas que haya en el cuatrimestre de estudio. Nótese que en este caso sí se trata de porcentajes respecto al total de alumnos.

- Datos: Los datos corresponden al total de alumnos inician la fase selectiva en un curso académico en concreto. Proceden del bloque *alumnes_interes*.

- Tipo de gráfico: Pieplot (Diagrama o gráfico de sectores).

- Método: *mes_num_suspensos(curso , cuatrimestre)* prepara los datos para cada curso académico y cuatrimestre de la fase inicial.

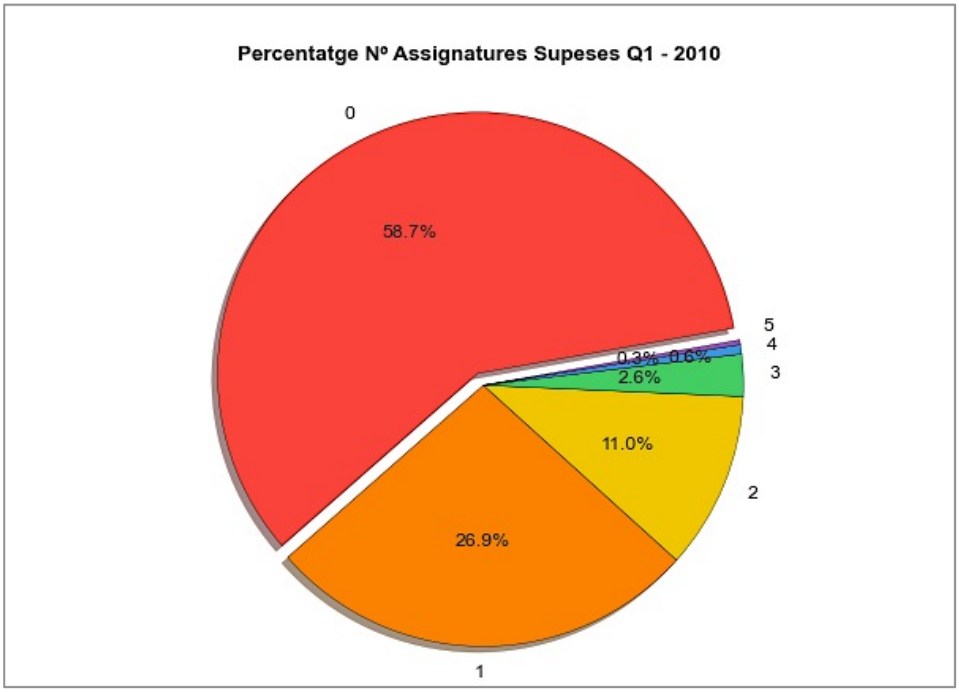


Fig. 6.5. Gráfico 5 – *Percentatge Suspesos per Nombre d'Assignatures* - 2010 - Q1.

Gráfico 6.- *Primer Curs Superat en 1 Any*

- Descripción: Se muestra el porcentaje de alumnos que superan la fase selectiva en un año respecto los que no. Ello implica tener todas las asignaturas del primer curso del grado aprobadas al finalizar el cuatrimestre de primavera del mismo curso académico en que se accedió a la escuela.

- Datos: Los datos corresponden al total de alumnos inician la fase selectiva en un curso académico en concreto. Proceden del bloque *alumnes_interes*.

- Tipo de gráfico: Pieplot.

- Método: *f_progres(curso)* prepara los datos para cada curso académico.

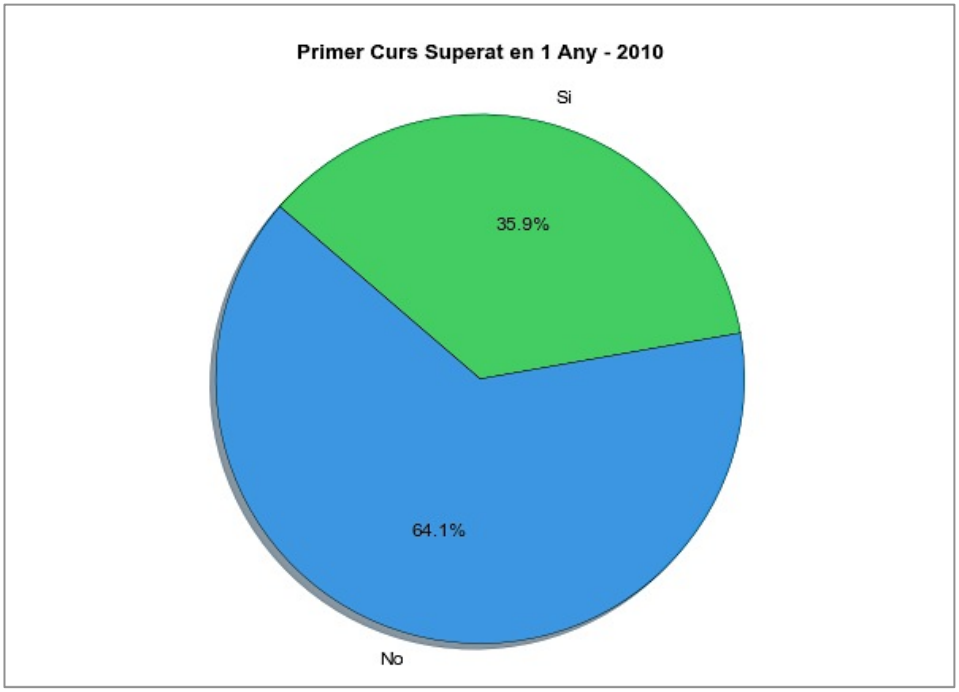


Fig. 6.6. Gráfico 6 – *Primer Curs Superat en 1 Any* - 2010.

Gráfico 7.- *Tot Aprovat i Primer Curs Superat en 1 Any*

- Descripción: Para aquellos alumnos que superan la fase selectiva en un año, se muestra el porcentaje de alumnos que han suspendido al menos una asignatura respecto los que la completan con todas aprobadas.

- Datos: Los datos corresponden al total de alumnos inician la fase selectiva en un curso académico en concreto y la superan en el periodo de un año. Proceden del bloque *alumnes_interes*.

- Tipo de gráfico: Pieplot.

- Método: *f_progres(curso)* prepara los datos para cada curso académico.

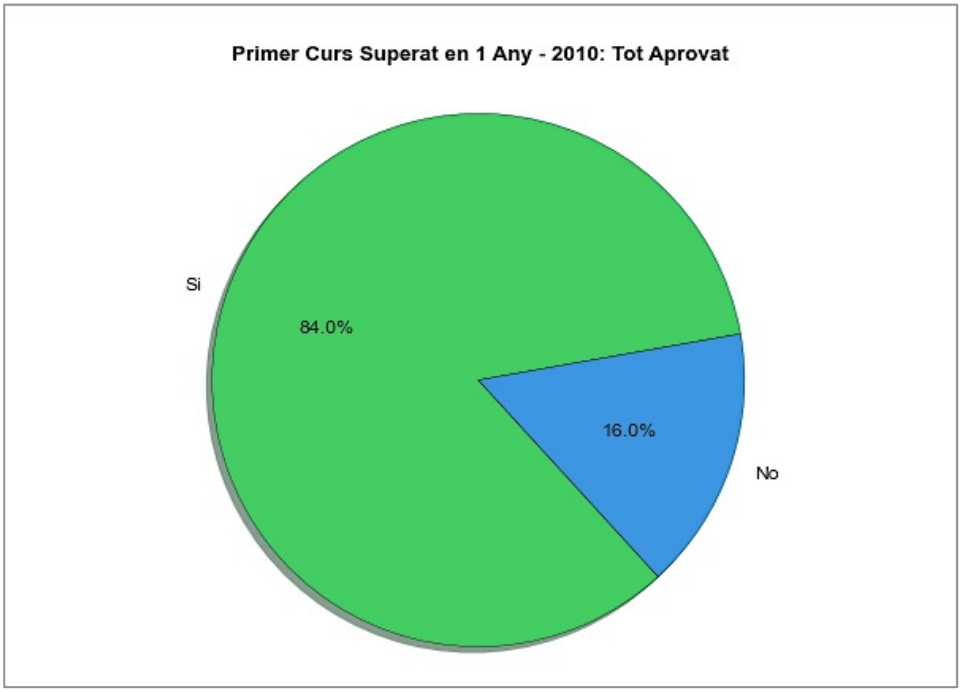


Fig. 6.7. Gráfico 7 – *Tot Aprovat i Primer Curs Superat en 1 Any* - 2010.

Gráfico 8.- *No Nets i Primer Curs Superat en 1 Any*

- Descripción: Se muestra el número de alumnos que suspenden cada asignatura del primer cuatrimestre de la fase selectiva, pese a haber conseguido finalizarla en un año. Este gráfico muestra únicamente las asignaturas del primer cuatrimestre ya que el suspenso de las asignaturas del segundo cuatrimestre, cursadas generalmente en primavera, implica directamente no finalizar la fase selectiva en un año.

- Datos: Los datos corresponden al total de alumnos inician la fase selectiva en un curso académico en concreto, la superan en el periodo de un año y han suspendido al menos una asignatura. Proceden del bloque *alumnes_interes*.

- Tipo de gráfico: Barplot.

- Método: *f_progres(curso)* prepara los datos para cada curso académico.

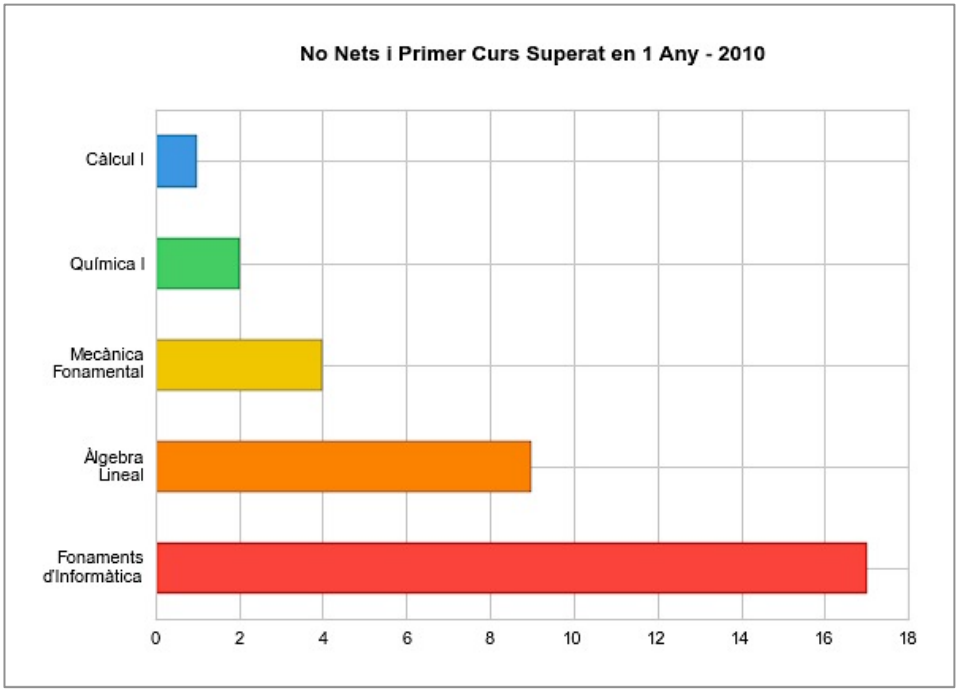


Fig. 6.8. Gráfico 8 – *No Nets i Primer Curs Superat en 1 Any* - 2010.

Para los ocho gráficos se ha creado la opción de visualizar los resultados agrupando todos los cursos que incluye la base de datos. Es decir, el resultado de sumar los resultados que se obtienen para cada año.

También se ha creado el método *crear_serie_df* que permite contrastar los resultados en el tiempo, es decir, para cada año y en orden cronológico. La figura 6.9 muestra cómo evoluciona el número de alumnos que han suspendido 3, 4 y 5 asignaturas en el primer cuatrimestre de la fase selectiva.

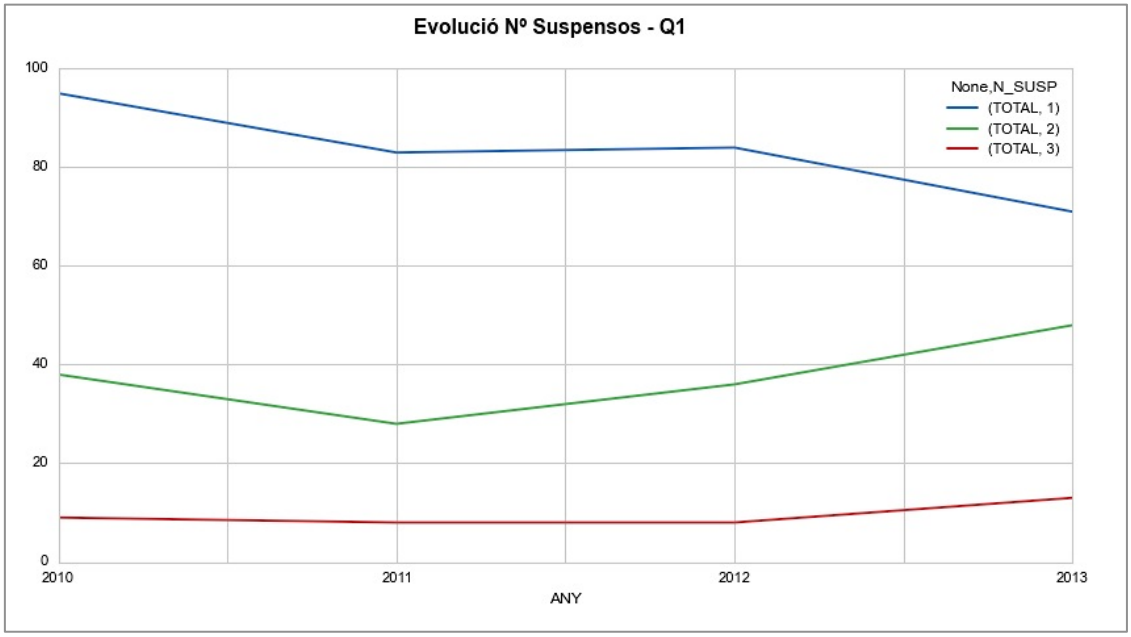


Fig. 6.9. Serie *Evolució Nº Suspensos – Q1.*

6.3.2.2. Procesamiento global

Para proporcionar la descripción estadística de las asignaturas del GETI, GEM y GEQ se ha optado por el uso de tablas. El contenido de éstas se han elaborado a imagen y semejanza de paneles de resultados expuestos en la memoria de la ETSEIB que se realiza anualmente; explícitamente en la memoria del año 2012-2013 [8].

Crear_df_taula es el método que se encarga del cómputo y creación de los DataFrames que contienen esta descripción estadística. A modo de ejemplo, se muestra el formato de las tablas resultantes en la Tabla 6.3. No obstante, el resultado real obtenido se puede encontrar en el anexo A:

				Curs 2012-2013								
				Matrícula	conv>1	Nota >=5	N entre 4 i 5	Nota <4	No Present.	% Aprovat/ Present.	% Aprovat/ Matric.	Nota mitjana
Q1	OB	240011	Àlgebra Lineal	105	104	80	9	9	7	82	76	5,7
	OB	240012	Càlcul I	126	125	91	17	10	8	77	72	5,5
	OB	240013	Mecànica Fonamental	134	133	77	22	27	8	61	57	4,8
	OB	240014	Química I	93	90	78	4	3	8	92	84	6,3
	OB	240015	Fonaments d'Informàtica	138	136	108	4	26		78	78	5,9
Q2	OB	240021	Geometria	381	31	244	82	47	8	65	64	5,3
	OB	240022	Càlcul II	382	24	178	98	94	12	48	47	4,7
	OB	240023	Termodinàmica Fonamental	394	43	272	41	81		69	69	5,4
	OB	240024	Química II	375	13	300	42	32	1	80	80	6,0
	OB	240025	Expressió Gràfica	394	20	279	45	68	2	71	71	5,3

Tabla 6.3. Resultados globales GETI – Cuatrimestre primavera – Curso 2012/13.

Los parámetros que se calculan son los siguientes:

- *Matriculats:* Número total de alumnos matriculados a la convocatoria.
- *Convalidats:* Número de alumnos que convalidan la asignatura en dicha convocatoria.
- *No Presentats:* Número de alumnos que no se presentan al examen final de la convocatoria.
- *Nota >= 5:* Número de alumnos cuya nota definitiva de la convocatoria es superior o igual a 5.0.
- *Nota entre 4 y 5:* Número de alumnos cuya nota definitiva de la convocatoria es superior o igual a 4.0 e inferior a 5.0.
- *Nota < 4:* Número de alumnos cuya nota definitiva de la convocatoria es inferior a 4.0.
- *% Aprov/Present:* Porcentaje de alumnos cuyo resultado de la convocatoria es superada respecto el total de alumnos que se presentan al examen final de la convocatoria.

- *% Aprov/Matric:* Porcentaje de alumnos cuyo resultado de la convocatoria es superada respecto el total de alumnos que se matriculan a dicha convocatoria.

- *Nota Mitjana:* Nota media calculada como media aritmética o promedio.

Nótese que, a diferencia de lo que sucede en el procesamiento de los datos para el estudio de la fase selectiva, los datos que se tienen en cuenta no realizan ningún seguimiento del alumno. No se valora o distingue aquéllos alumnos que cursan la asignatura por segunda o tercera vez.

Para la obtención de estas tablas se tiene en cuenta que el plan Bolonia se instaura en la escuela el curso académico de 2010. Esto significa que el número de asignaturas de las que se conoce los resultados académicos va aumentando progresivamente. El tamaño de las tablas aumentará paralelamente.

También se ha creado el método *crear_serie_taula* que permite mostrar los resultados de cada asignatura en el tiempo. La figura 6.10 muestra la evolución temporal del porcentaje de alumnos del GETI que superan la asignatura de álgebra lineal respecto al número de presentados al examen final y respecto a los matriculados.

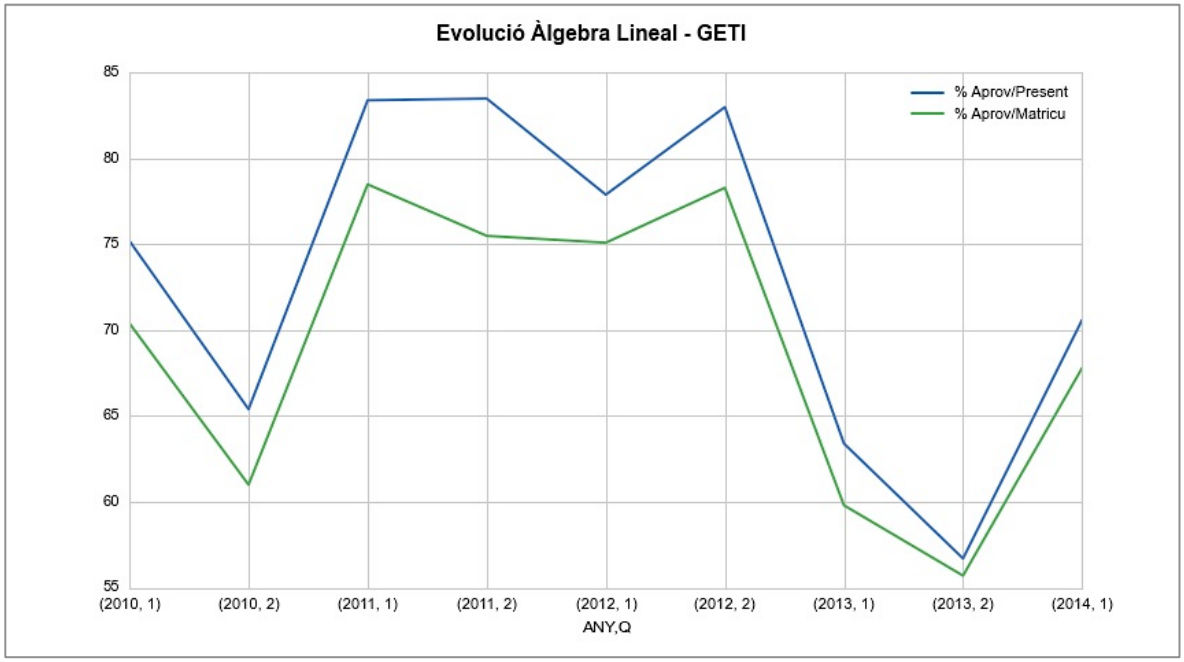


Fig. 6.10. Serie *Evolució Àlgebra Lineal – GETI.*

6.3.3. Optimización

Uno de los objetivos de este proyecto es conseguir que la aplicación tarde el mínimo tiempo posible en ejecutarse y que funcione de forma fluida. No obstante, algunos de los métodos y subrutinas que contiene la clase *DataStats* se basan en algoritmos de naturaleza iterativa. A medida que aumenta el número de datos sobre los que se aplican estos algoritmos aumenta también el tiempo requerido para el cómputo de los resultados deseados.

Se parte de un programa que tarda entorno 6 minutos y 45 segundos en ejecutarse. Teniendo el objetivo presente, se realizó una optimización de algunos de los mecanismos que emplea esta clase. Entre ellos, cabe destacar el diseño de un nuevo algoritmo más eficiente para el funcionamiento del método *flux,* encargado de analizar la evolución académica de los alumnos en la fase selectiva. También se estudió detenidamente los métodos Pandas empleados para manipular los datos en general y se encontró alternativas de cómputo más rápidas.

Pese a la primera optimización realizada sobre los métodos de *DataStats*, el tiempo que se tardaba en inicializarla era de aproximadamente 3 minutos y 20 segundos. Además, cada vez que se deseaba obtener los resultados referentes al estudio estadístico de un año en particular se debía esperar a que la clase realizase todos los cálculos pertinentes. Todo ello resulta tardío e ineficiente, con lo que se llevó a cabo una segunda optimización.

Esta mejora debía solucionar simultáneamente dos problemas: en primer lugar, el elevado tiempo de inicialización de la clase; y en segundo lugar, la espera que implicaba la obtención de ciertos datos que requeriría la aplicación en tiempo real.

La segunda optimización ha consistido en la modificación del mecanismo de inicialización de la clase *DataStats* y de los métodos que incorpora. Lo que se ha hecho es conferir a los métodos más lentos de un sistema que almacena sus resultados en ficheros internos. Al inicializar la clase, en vez de empezar directamente con el proceso de preparación de los datos, ésta consultará la existencia de los ficheros que contienen los datos ya tratados y los descargará en caso afirmativo. Respecto al procesamiento de los datos, cada vez que se consulte unos resultados en concreto, el método que los genera buscará si el fichero ya existe. De lo contrario, llevará a cabo su función habitual, almacenando en un nuevo fichero tales resultados una vez calculados.

Nótese que cada vez que se cambia la base de datos de partida habrá que recalcular todos los ficheros. Para ello se ha dotado al programa de un sistema para comprobar que los ficheros internos de los que se dispone concuerdan con la base de datos que se introduzca.

Con todo, lo que se consigue es aislar el tiempo de espera a la primera ejecución y asegurar que la aplicación pueda acceder a los datos de la clase de forma fluida. Se ha creado dos pequeños programas para obtener los datos estadísticos referentes a los tiempos de ejecución de la clase *DataStats.* Uno de ellos simula la primera ejecución y el otro simula el resto de ejecuciones, es decir, una vez se tienen los datos tratados. Tales programas pueden encontrarse en el anexo A. Los resultados obtenidos para una simulación de 100 ejecuciones se muestran en la siguiente tabla:

Tiempos de ejecución de *DataStats*			
Parámetro	**Cálculo**	**Primera ejecución [s]**	**Resto ejecuciones [s]**
Tiempo medio	$\bar{t} = \frac{1}{100} \cdot \sum_{i=1}^{100} t_i$	198,01	18,66
Tiempo máximo	$t_{max} = max(t_1, t_2, \dots, t_{100})$	376,94	23,41
Tiempo mínimo	$t_{min} = min(t_1, t_2, \dots, t_{100})$	179,06	17,18
Desviación Estándar	$\sigma = \sqrt{\frac{\sum_{i=1}^{100}(t_i - \bar{t})^2}{100 - 1}}$	31,69	0,96

Tabla 6.4. Descripción estadística del tiempo de ejecución de la clase DataStats*.*

Para la primera ejecución, se puede observar que el tiempo medio ronda los 3 minutos y 20 segundos. Para el resto de ejecuciones, la espera media es de 19 segundos, aproximadamente. Estos resultados no son alarmantes, sino que resultan una gran mejora en comparación con los 6 minutos y 45 segundos que se tenía al principio para cualquier ejecución.

El tiempo máximo para una primera inicialización parece indicar que es el doble que la media. Sin embargo, se teme que este resultado puede estar distorsionado por la ralentización de la computadora debido a la “sobrecarga”. Esto se cree así por la no normalidad de la distribución de los resultados; en que la media parece ir aumentando progresivamente. Los gráficos de las distribuciones de estos tiempos aparecen en el anexo A.

7. Diseño y desarrollo de la aplicación

Una vez se tienen los datos procesados, es necesario crear un conjunto de herramientas que permitan mostrar al usuario los resultados de dicho procesamiento. Con este objetivo se crea la GUI. Para ello lo primero que se hizo fue un diseño conceptual a modo de croquis de las ventanas que debía contener la aplicación. Se procedió elaborando un diseño funcional del software, también conceptual, que permitiese comunicar la clase *DataStats* con la interfaz gráfica y el usuario mismo. A partir de este punto se inició la fase de desarrollo y programación, que dio lugar a la aplicación objeto del proyecto: PyPRISMA.

7.1. Diseño y funcionamiento

En esta sección se muestran las partes que forman la GUI, el diseño de las mismas y cómo funcionan desde el punto de vista del usuario. También se mostrará y explicará el esquema de funcionamiento interno de la aplicación. Todo ello sin entrar en detalles del software que lo implementa, que será abordado en la sección 7.3.

7.1.1. Diseño de la GUI

El programa consta de un vídeo introductorio, que se inicia nada más ejecutar la aplicación, y dos ventanas diferenciadas:

1. El diálogo de importación de archivos.
2. La ventana principal.

Cabe mencionar que estas ventanas han sido desarrolladas de modo que pueden ser redimensionadas, movidas, ocultadas, restauradas y cerradas a voluntad. Todo ello sin perjudicar la distribución de los elementos que configuran la ventana y manteniendo el aspecto de la misma. Además, todas las ventanas de la aplicación tienen un comportamiento no modal respecto al sistema, es decir, permiten alternar el foco a cualquier otra ventana presente dentro del entorno gráfico o escritorio.

7.1.1.1. Diálogo de importación de archivos

En primer lugar se tiene el diálogo de importación de archivos, donde se introducen

los datos que se desee estudiar. Por diálogo se entiende una ventana con la que interactúa el usuario final. El diseño del mismo puede verse a continuación:

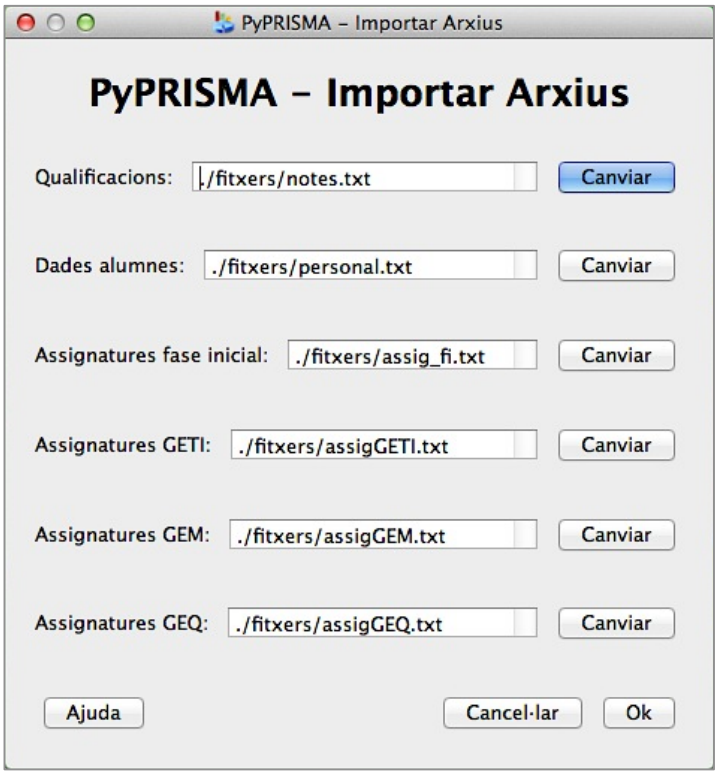


Fig. 7.1. Diálogo de inicio de la aplicación – Importación de archivos.

Como se puede observar en la figura 7.1, para cada casilla aparece la ruta relativa donde se encuentran los ficheros de datos que emplea la aplicación por defecto. Si se dispone de una nueva base de datos, mediante el botón *Canviar* aparece un cuadro de diálogo para seleccionar los nuevos ficheros que quieran ser empleados. El botón *Cancel·lar* anula la descarga de los nuevos ficheros introducidos y ejecuta la aplicación con los que dispone por defecto. En cambio, el botón *Ok* carga los ficheros introducidos y ejecuta la aplicación con éstos.

El programa dispone de una estrategia de detección y tratamiento de errores extensa. En función del error detectado, aparece un mensaje crítico informando al usuario del tipo de error que está teniendo lugar. En el proceso de importación de archivos, podemos distinguir tres casos:

1. La ruta introducida en alguna casilla no corresponde a ningún fichero y por tanto no es válida. El mensaje de alerta que aparece en la pantalla es el siguiente:

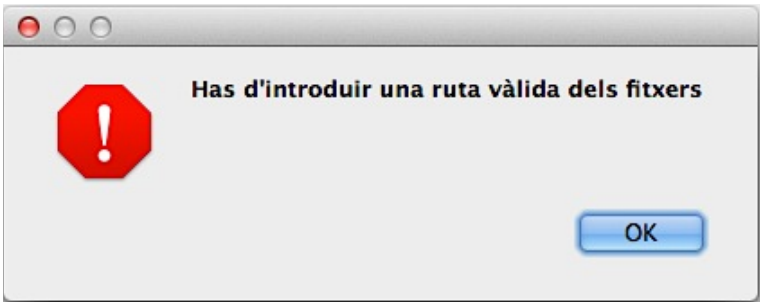


Fig. 7.2. Mensaje de alerta 1 – Ruta de ficheros inválida.

2. El contenido de alguno de los ficheros introducidos no es el esperado; ya sea porque la información que proporciona no es la correcta, faltan columnas de datos o el formato es incorrecto. El mensaje de alerta que aparece en la pantalla es el siguiente:

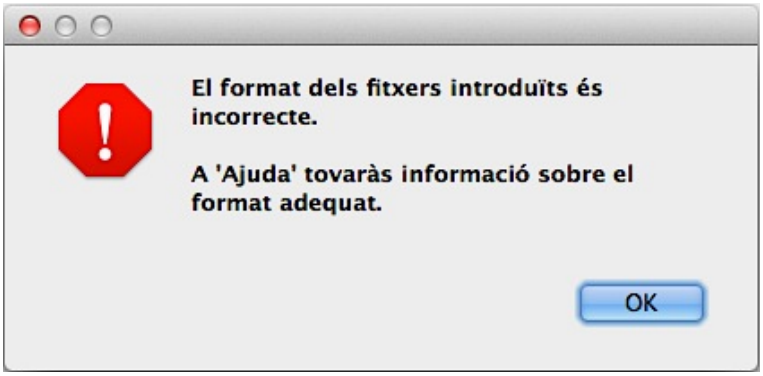


Fig. 7.3. Mensaje de alerta 2 – Formato de ficheros incorrecto.

3. Se desconoce la procedencia del error, pero los ficheros introducidos no son válidos. Ésta es una medida preventiva que se toma para asegurarse de que el programa no colapsa frente a la aparición de errores de origen desconocido. El mensaje de alerta que aparece en la pantalla es el siguiente:

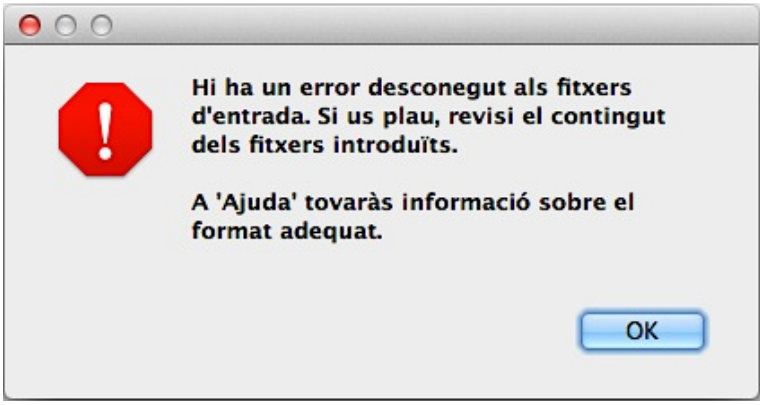


Fig. 7.4. Mensaje de alerta 3 – Error desconocido en ficheros.

Cuando aparece uno de estos mensajes de alerta, es necesario cerrarlo para reactivar la ventana de importación de archivos. Este tipo de comportamiento se llama modal respecto a la aplicación. Permite alternar el foco a otras ventanas del sistema, pero no a la ventana que le da origen (ventana madre) hasta que se toma una acción sobre ella.

Si aparece uno de los mensajes de alerta, o bien si deseamos información, el botón *Ajuda* que aparece en la figura 7.1 abre un nuevo diálogo a modo pop-up, es decir, que aparece sobre cualquier ventana que se tenga abierta en la computadora, con información referente al contenido y formato que han de tener los ficheros que toma la aplicación. Éste se ha creado para evitar que el usuario introduzca ficheros de contenido incorrecto y para que no necesite recurrir a los ficheros internos de la aplicación para conocer tal formato. El diálogo de ayuda puede verse en la figura 7.5**.** Adicionalmente, el contenido completo que muestra este diálogo puede encontrarse en el anexo B.

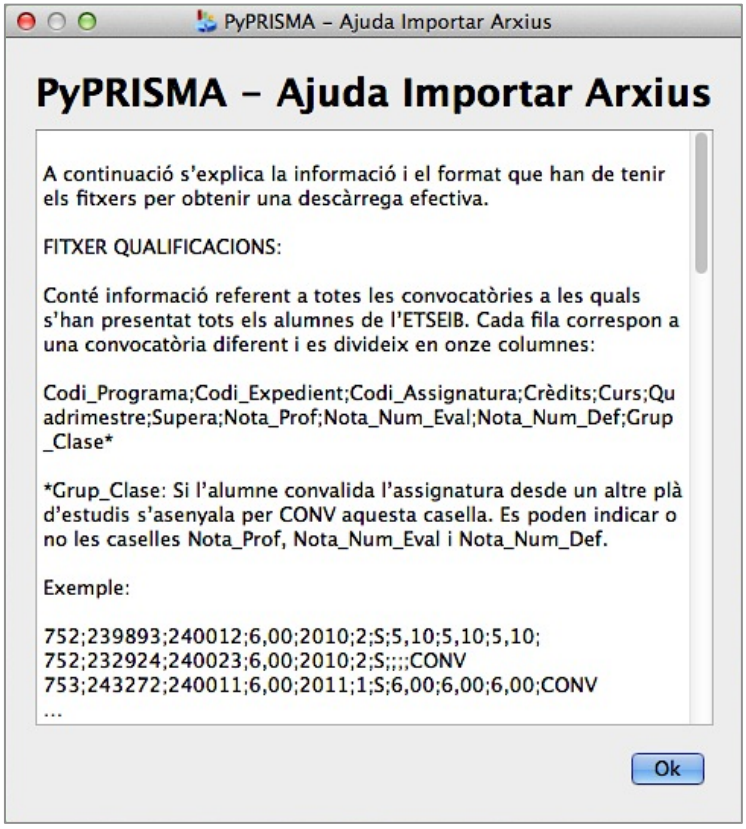


Fig. 7.5. Diálogo de ayuda para la importación de archivos.

Al igual que ocurre con los mensajes de alerta, el diálogo de ayuda es de comportamiento modal respecto a la aplicación y por tanto es necesario cerrar este diálogo de ayuda para reactivar la ventana de importación de archivos.

7.1.1.2. Ventana principal

En segundo lugar se tiene la ventana principal de la aplicación, encargada de mostrar los resultados del análisis de los datos. Ésta contiene una barra de menús en esquina superior izquierda. Se tiene tres menús diferentes: *Arxiu*, *Veure* y *Ajuda*. Si se selecciona cada uno de ellos, aparecen los siguientes desplegables:

Menú 1.- *Arxiu*

- *Tancar:* Se cierra la aplicación al seleccionarlo y muestra el atajo de teclado que puede emplearse para llevar a cabo esta acción (⌘ T).

- *Recarregar Fitxers:* Se abre el diálogo de importación de archivos que se ha explicado anteriormente y muestra el atajo de teclado que puede emplearse para

llevar a cabo esta acción (⌘F).

La figura 7.6 muestra el menú *Arxiu* con las opciones que incorpora. Cabe mencionar que al colocar el cursor sobre alguna de estas dos casillas se mostrará en la esquina inferior izquierda un mensaje, él elemento que lo contiene es la barra de estado. Adicionalmente y por descontado, el cierre de la aplicación también puede llevarse a cabo mediante los botones superiores izquierdos que aparecen en cualquier ventana de una aplicación de escritorio. Con ellos también se puede ocultar o redimensionar la aplicación.

Menú 2.- *Veure*

- *Barra d'eines:* Si se activa, se muestra la barra de herramientas con las opciones que ofrece el menú *Arxiu* i el menú *Ajuda*. De lo contrario, la barra de herramientas se esconde. Se presenta activada por defecto.

En la figura 7.6 muestra el menú *Veure* con las opciones que incorpora:

Menú 3.- *Ajuda*

- *Tutorial PyPRISMA:* Se abre el diálogo con una presentación a modo de diapositivas. Mediante las flechas izquierda y derecha del teclado se retrocede o avanza una diapositiva, respectivamente. El diálogo puede cerrarse mediante la tecla de escape o bien con el aspa rojo habitual situado en la parte superior izquierda de la ventana.

- *Sobre PyPRISMA:* Se abre un diálogo con información general de la aplicación.

Ambos diálogos tienen un comportamiento no modal respecto a la aplicación, con lo que por ejemplo se puede utilizar la aplicación mientras se va siguiendo el tutorial. Tanto el segundo diálogo como el manual de usuario pueden verse en el anexo C. En la siguiente figura se puede ver el menú *Ajuda* con las opciones que incorpora:

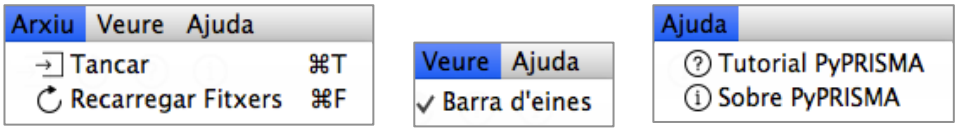


Fig. 7.6. Menú *Arxiu* a la izquierda, *Veure* en el centro y *Ajuda* a la derecha.

Por otro lado, a partir de la ventana principal se puede acceder a siete vistas centrales que se han creado para presentar los resultados del estudio estadístico en función de la tipología de los mismos. Éstas vistas forman parte de la propia ventana principal. Se accede a ellas a través de unas pestañas que aparecen en la parte superior, de modo que al seleccionar una pestaña, la vista central correspondiente se actualiza.

Vista central 1.- *Gràfics Fase Inicial*

En ella se muestran los resultados gráficos del procesamiento de datos de la fase inicial. Está dividida en dos partes simétricas que funcionan por separado pero ofrecen las mismas opciones. Esto se ha hecho así para permitir al usuario comparar resultados de diferentes años, cuatrimestres o atributos de los que depende el gráfico en cuestión.

En la figura 7.7 se muestra el diseño global de esta ventana y cómo aparecen expuestos los resultados gráficos en ella. En la figura se ha seleccionado el gráfico de *Percentatge Suspesos per Assignatura* para el curso académico de 2010 en ambos lados, pero el primer cuatrimestre en el gráfico 1 y el segundo cuatrimestre en el gráfico 2.

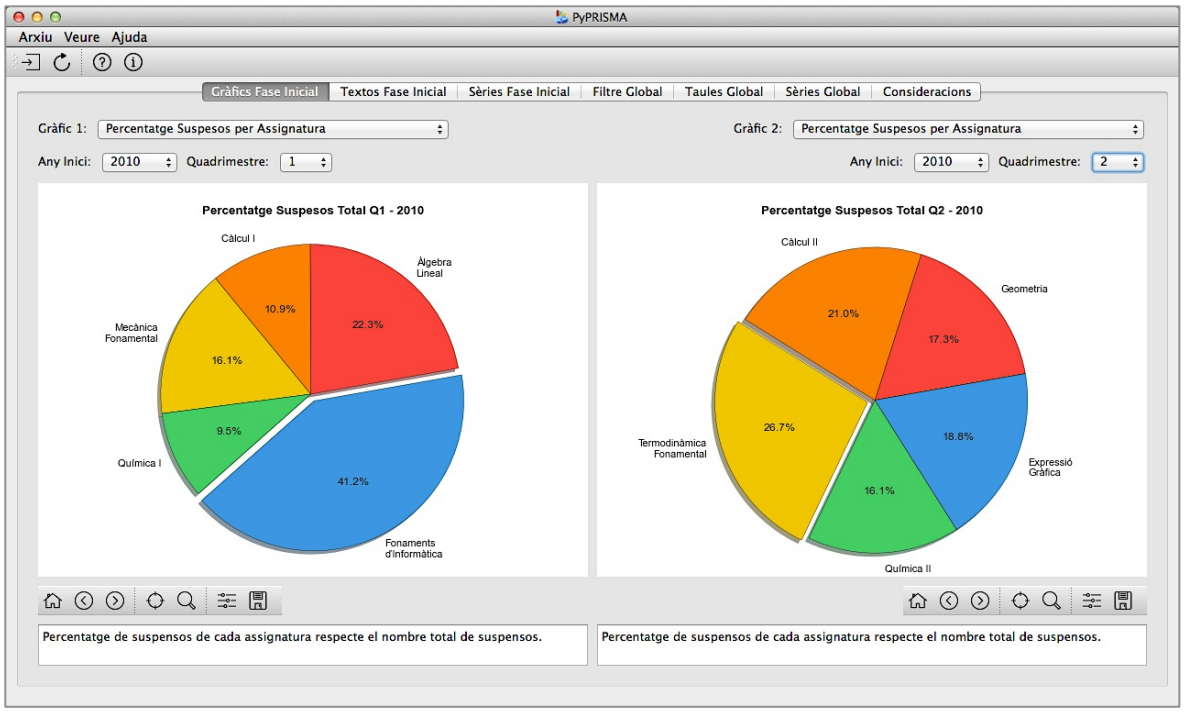


Fig. 7.7. Ventana *Gràfics Fase Inicial*.

La ventana *Gràfics Fase Inicial* se encarga de mostrar los ocho gráficos que han sido comentados en la sección 6.3.2.1 de procesamiento de datos para la fase inicial. Como las dos partes que la forman funcionan igual pero por separado, se explicará el funcionamiento del gráfico 1 directamente. La figura 7.8 muestra los elementos que dispone cada mitad.

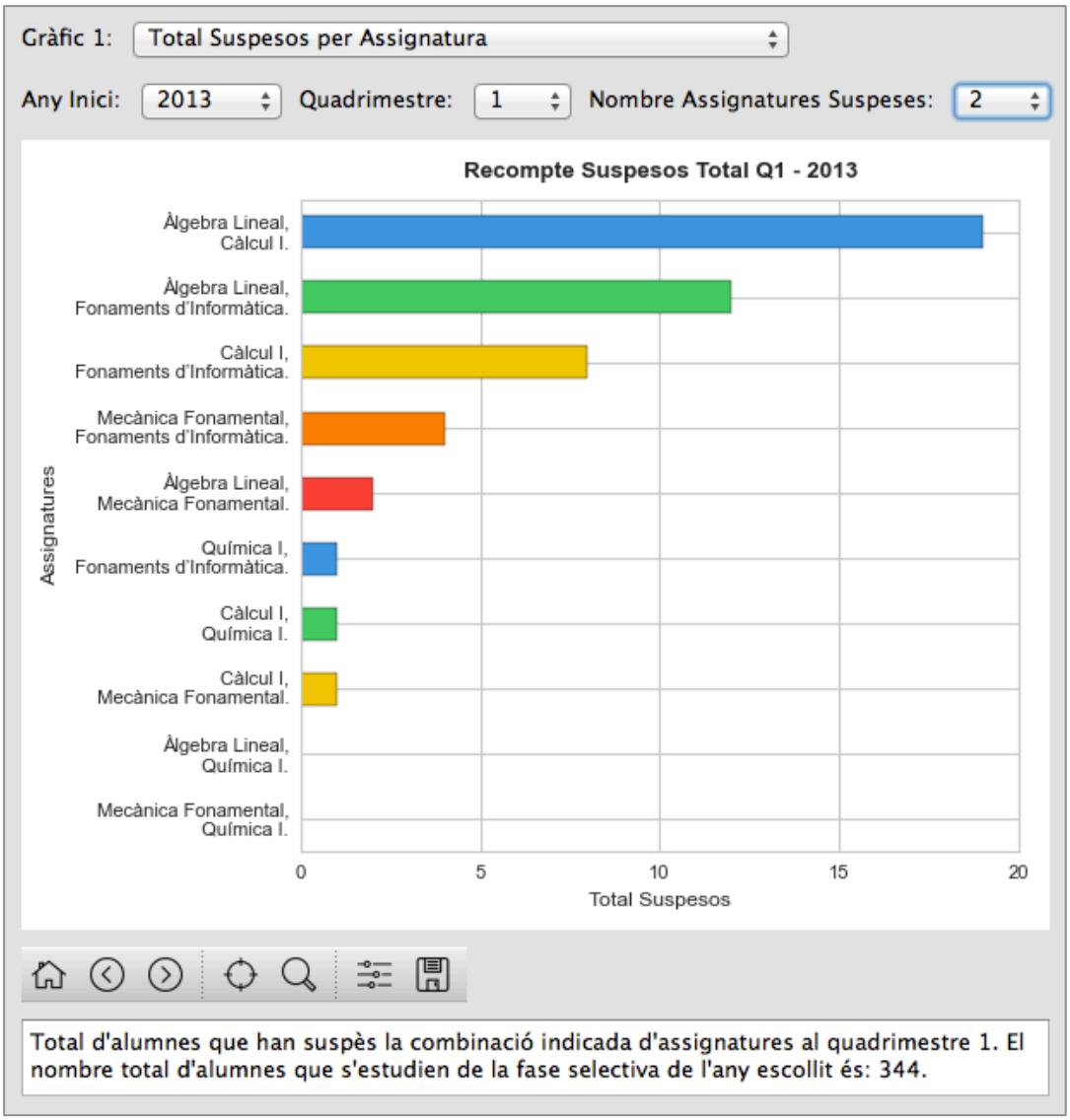


Fig. 7.8. Ventana *Gràfics Fase Inicial* – Elementos de control del gráfico 1

En primer lugar, para escoger el gráfico que desee visualizarse se selecciona el desplegable que aparece en la parte superior. Como hay gráficos que dependen de diferentes parámetros, en función del gráfico escogido aparecerán debajo del primer desplegable otro conjunto de éstos. En el ejemplo que aparece en la figura 7.8 se muestra el gráfico *Total Suspesos per Assignatura*; ya que éste gráfico depende del año de inicio de la

fase selectiva, del cuatrimestre y del número de asignaturas suspendidas, aparecen tres desplegables para poder seleccionar el valor deseado de cada uno de ellos respectivamente.

En segundo lugar, se tiene el canvas (lienzo) donde se expone el resultado gráfico. Debajo de éste se dispone de una barra de herramientas para navegar dentro del canvas y cambiar los parámetros de diseño del gráfico definidos por defecto. La funcionalidad concreta que ofrece cada botón se expone en la tabla 7.1. Por otro lado, la exposición completa de los atajos de teclado para la barra de navegación se encuentra en el anexo B.

Por último, en la parte inferior se muestra un recuadro con una explicación del gráfico y muestra algunos datos que puedan ser de interés para su mayor comprensión.

Botones Navigation Toolbar		
Icono	**Comando**	**Funcionalidad**
	Inicio	Volver a la vista por defecto del gráfico.[1]
	Atrás	Retroceder a la vista anterior. [1]
	Adelante	Avanzar a la vista siguiente. [1] [2]
	Panorámica	Desplazar el enfoque de la vista a una región específica o escalar.
	Zoom	Magnificar una región rectangular concreta del gráfico.
	Configuración	Modificar parámetros del gráfico. [3]
	Guardar	Abrir un diálogo para guardar el gráfico. [4]

Tabla 7.1. Funcionalidad botones barra de navegación del canvas.

[1] Esta funcionalidad sólo tiene sentido si se ha utilizado alguna de las demás herramientas, como por ejemplo vista panorámica o zoom, de modo que se ha representado el gráfico con más de una vista.

[2] Esta funcionalidad solo tiene sentido si previamente se ha utilizado la herramienta Inicio o Atrás.

[3] Configuración del espacio entre filas y columnas, longitudes de los ejes, etc.

[4] Formatos disponibles: png, ps, EPS, SVG o PDF.

Vista central 2.- *Textos Fase Inicial*

En ella se muestran los resultados tipo texto que se obtienen mediante el método *crear_txt* de la clase *DataStats* mencionado en la sección 6.3.2.1 del procesamiento de datos de la fase inicial. Del mismo modo que ocurre con la ventana *Fase Inicial*, está dividida en dos partes simétricas que funcionan por separado pero ofrecen las mismas opciones, permitiendo al usuario comparar resultados. Cada mitad dispone de unos desplegables que permiten seleccionar el año de inicio y el cuatrimestre del que se desea conocer la distribución en cuanto a asignaturas suspendidas de los alumnos.

En la figura 7.9 se muestra el diseño de esta ventana. Los resultados que muestra corresponden a la fase selectiva iniciada en 2010 en ambos lados, pero el primer cuatrimestre en el lado izquierdo y el segundo cuatrimestre en el derecho.

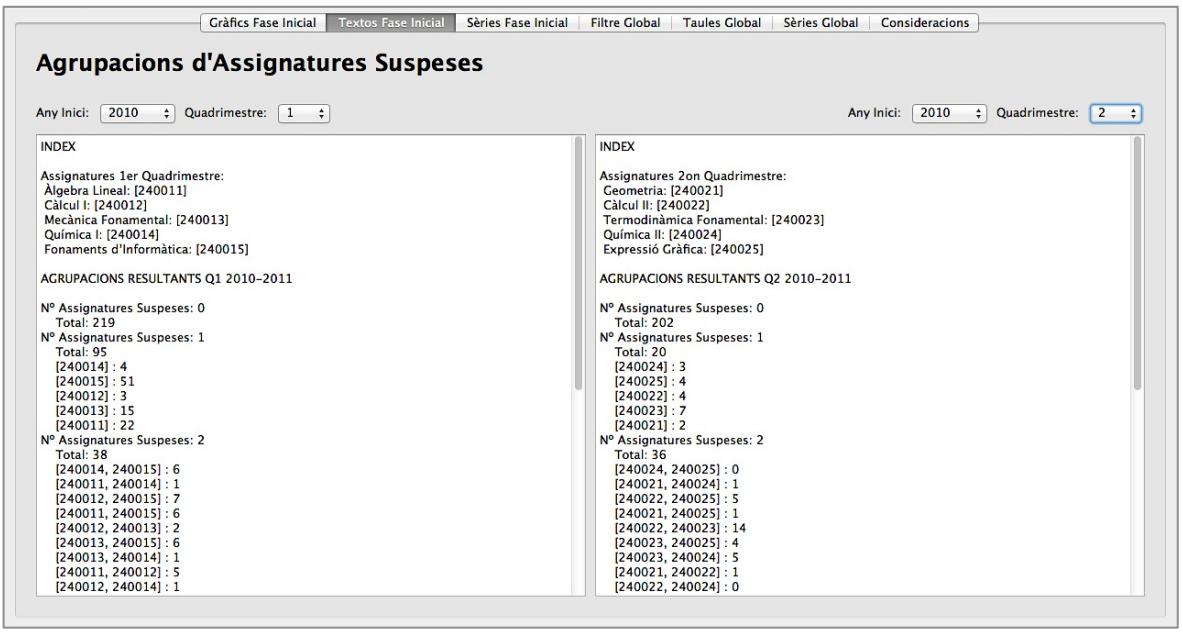


Fig. 7.9. Ventana *Textos Fase Inicial*.

Vista central 3.- *Sèries Fase Inicial*

En ella se puede visualizar la evolución temporal de los resultados del estudio de la fase inicial. Por un lado, se puede ver como cambia el número de alumnos que suspende un número de asignaturas determinado. Por otro lado, existe la opción de ver la tasa de suspensos que tiene cada combinación de asignaturas suspendidas cada año.

En esta vista también se tiene un desplegable para seleccionar el cuatrimestre del que se quiere visualizar los resultados.

Nótese que si se selecciona la opción del cuatrimestre de primavera, las combinaciones de asignaturas suspendidas se actualizan con los códigos de las asignaturas que se cursan en ese cuatrimestre de la fase selectiva: 240021, 240022, 240023, 240024 y 240025.

En la figura 7.10 aparece el número total de alumnos que suspenden únicamente Álgebra Lineal (240011), únicamente Fundamentos de Informática (240015) y los que suspenden solamente ambas. Estos resultados se muestran para cada curso académico.

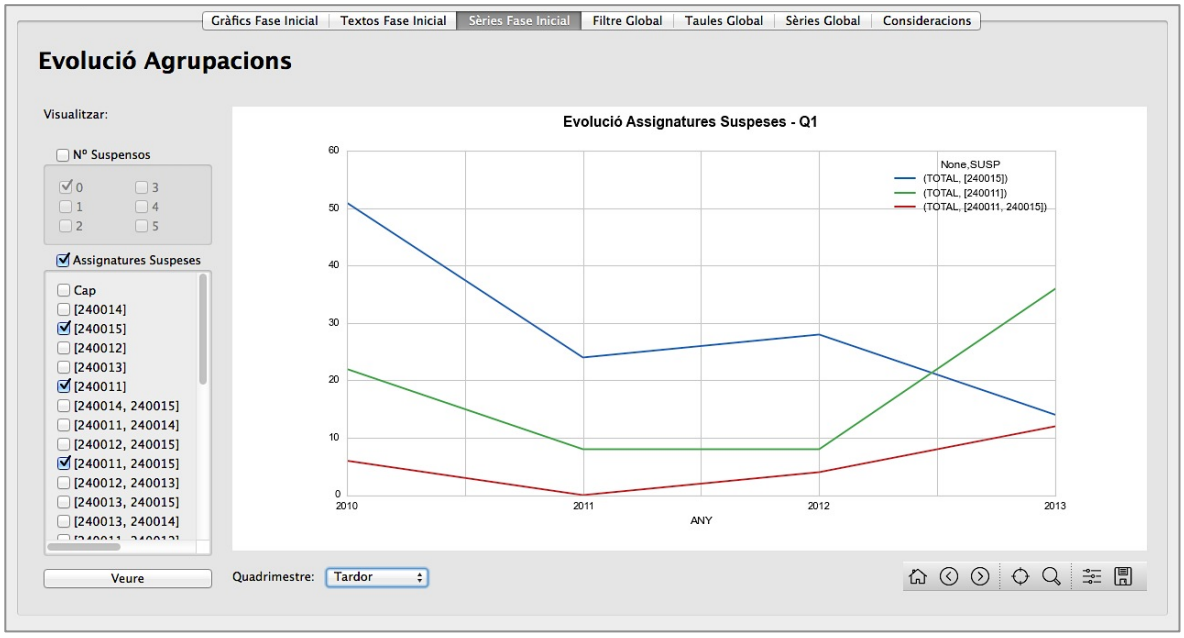


Fig. 7.10. Ventana *Sèries Fase Inicial*.

Vista central 4.- *Filtre global*

En esta vista se introducen los criterios de selección del subconjunto de datos que se estudia en el procesamiento global. Los resultados que se expongan en las vistas cinco y seis dependerán de este filtro. En la figura 7.11 se muestra el diseño de esta ventana.

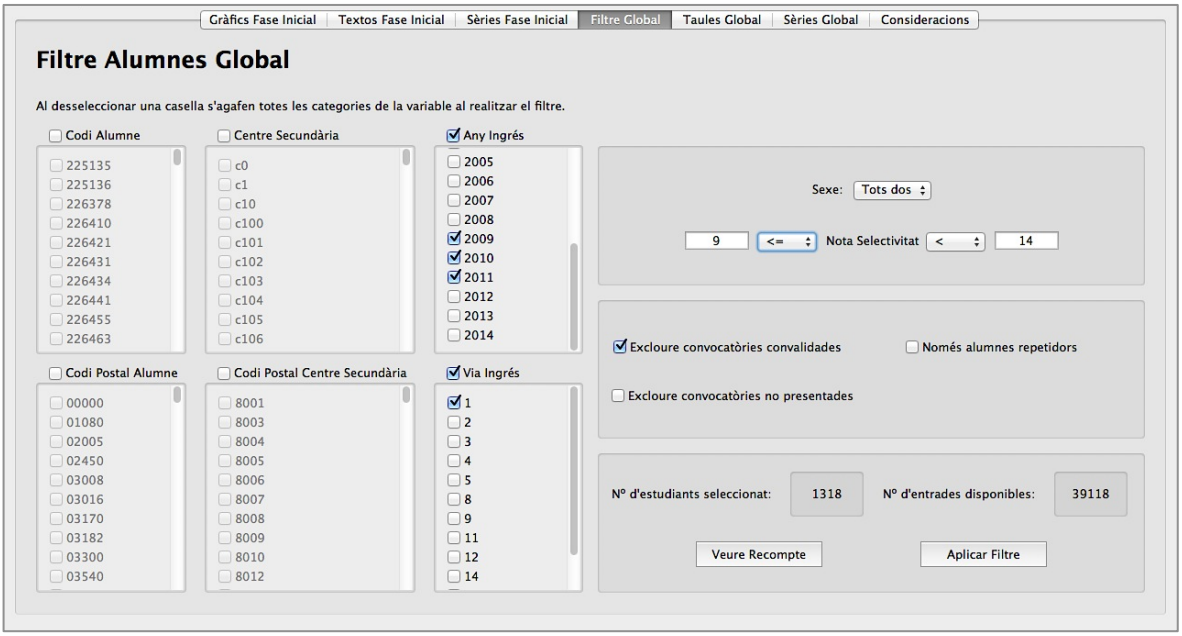
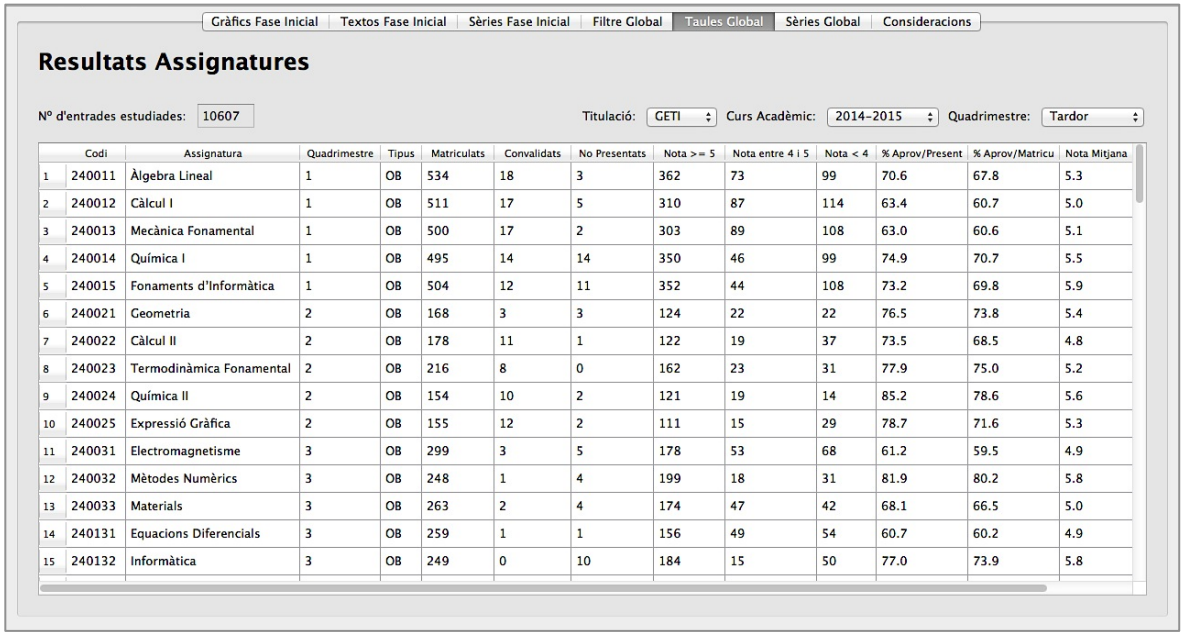


Fig. 7.11. Ventana *Filtre Global*.

Vista central 5.- *Taules Global*

En ella se muestran los resultados tipo tabla que se obtienen mediante el método *crear_df_taula* de la clase *DataStats* mencionado en la sección 6.3.2.2 del procesamiento de datos globales. Se recuerda que éstos representan una descripción estadística de todas las asignaturas para cada titulación, curso académico y cuatrimestre, una vez instaurado el Plan Bolonia. Es por ello que sobre la tabla principal aparecen tres desplegables en los que seleccionar tales parámetros. Las diferentes columnas de la tabla pueden modificarse en tamaño según desee el usuario.

En la figura 7.12 se muestra el diseño de esta ventana y cómo aparecen expuestos los resultados tipo tabla en ella. En la figura se ha seleccionado la titulación GETI, el curso académico de 2011-2012 y el cuatrimestre de primavera.



	Codi	Assignatura	Quadrimestre	Tipus	Matriculats	Convalidats	No Presentats	Nota >= 5	Nota entre 4 i 5	Nota < 4	% Aprov/Present	% Aprov/Matricu	Nota Mitjana
1	240011	Àlgebra Lineal	1	OB	534	18	3	362	73	99	70.6	67.8	5.3
2	240012	Càlcul I	1	OB	511	17	5	310	87	114	63.4	60.7	5.0
3	240013	Mecànica Fonamental	1	OB	500	17	2	303	89	108	63.0	60.6	5.1
4	240014	Química I	1	OB	495	14	14	350	46	99	74.9	70.7	5.5
5	240015	Fonaments d'Informàtica	1	OB	504	12	11	352	44	108	73.2	69.8	5.9
6	240021	Geometria	2	OB	168	3	3	124	22	22	76.5	73.8	5.4
7	240022	Càlcul II	2	OB	178	11	1	122	19	37	73.5	68.5	4.8
8	240023	Termodinàmica Fonamental	2	OB	216	8	0	162	23	31	77.9	75.0	5.2
9	240024	Química II	2	OB	154	10	2	121	19	14	85.2	78.6	5.6
10	240025	Expressió Gràfica	2	OB	155	12	2	111	15	29	78.7	71.6	5.3
11	240031	Electromagnetisme	3	OB	299	3	5	178	53	68	61.2	59.5	4.9
12	240032	Mètodes Numèrics	3	OB	248	1	4	199	18	31	81.9	80.2	5.8
13	240033	Materials	3	OB	263	2	4	174	47	42	68.1	66.5	5.0
14	240131	Equacions Diferencials	3	OB	259	1	1	156	49	54	60.7	60.2	4.9
15	240132	Informàtica	3	OB	249	0	10	184	15	50	77.0	73.9	5.8

Fig. 7.12. Ventana *Taules Global*.

Vista central 6.- *Sèries Fase Inicial*

En ella se muestran los resultados para cada asignatura expuestos en el tiempo. Como ejemplo, en la figura 7.13 se puede ver la evolución del porcentaje de alumnos aprobados respecto a los matriculados y presentados para Fundamentos de Informática.

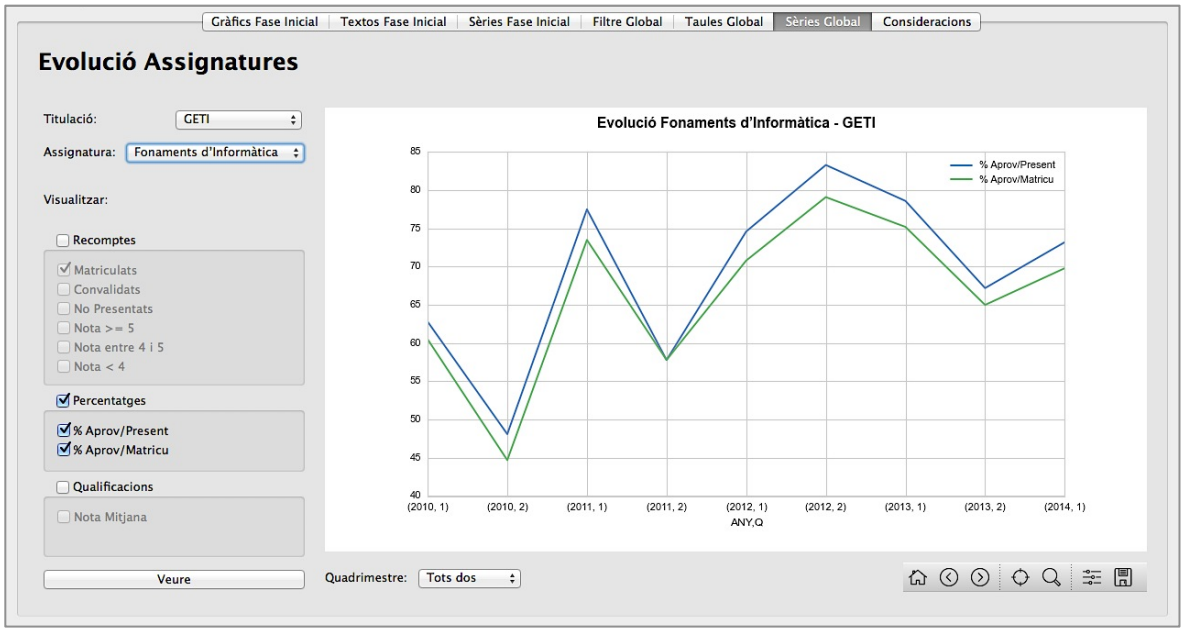


Fig. 7.13. Ventana *Sèries Global*.

Vista central 7.- *Consideracions*

Esta vista es puramente informativa, en el sentido de que no expone ningún resultado específico del procesamiento de los datos. En ella se explica el propósito de la aplicación y las consideraciones que se han tenido a la hora de seleccionar el conjunto de alumnos que se estudian para cada bloque de la fase selectiva. En parte, el objetivo de esta pestaña es sacar de dudas al usuario. Finalmente, se muestra una imagen de la escuela. En la figura 7.14 se muestra el diseño de esta ventana.

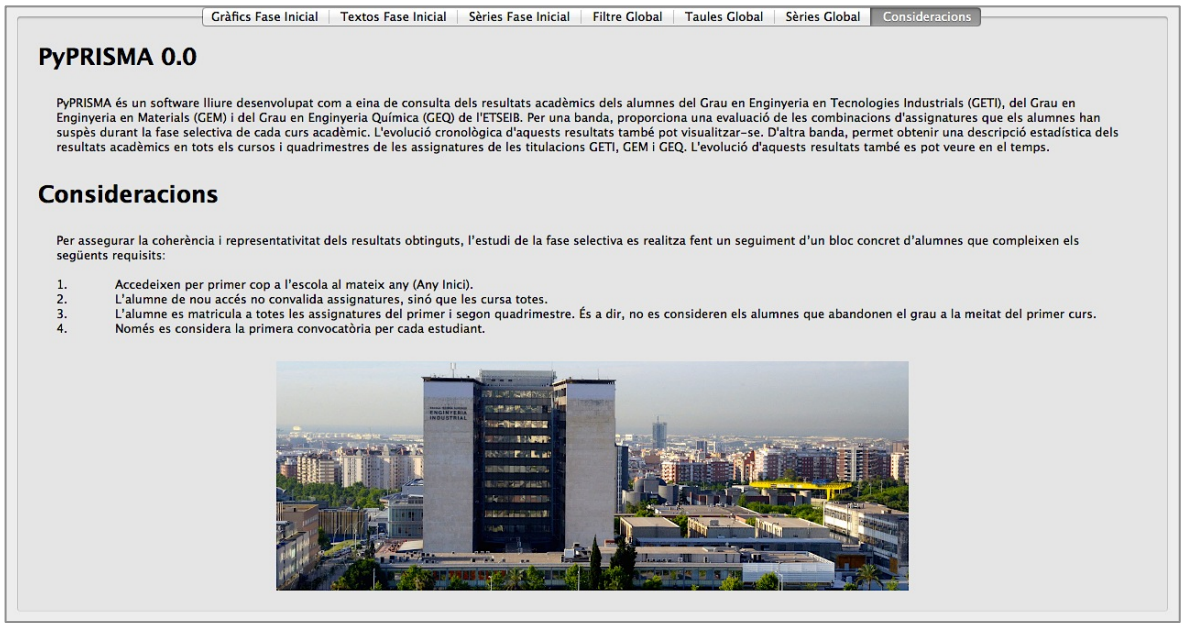


Fig. 7.14. Ventana *Consideracions*.

7.1.2. Diseño funcional

En este apartado se desarrolla una descripción del funcionamiento del programa desde el punto de vista de la programación, es decir, cómo funciona el programa internamente y la arquitectura que posee.

Para poder plasmar de una manera gráfica el funcionamiento lógico del programa se ha recurrido a los diagramas de flujo, a través de los cuales se puede reflejar el flujo de control y de datos entre las distintas actividades que tienen lugar desde que se inicia la aplicación hasta que se cierra.

Se ha dividido la estructura en dos niveles:

1. Flujograma global
2. Flujogramas específicos

7.1.2.1. Flujograma global

Muestra un esquema básico y global de la secuencia de ventanas que aparecen desde que se ejecuta hasta que se cierra la aplicación. Los rectángulos de color morado son subprocesos o subrutinas, representados a modo de caja negra y que se desglosan en el apartado de flujogramas específicos. Los romboides amarillos representan el valor de un dato almacenado. Éstos últimos son simbólicos, no se han introducido como tal, pero simplifican y optimizan el diagrama con la lógica de la aplicación.

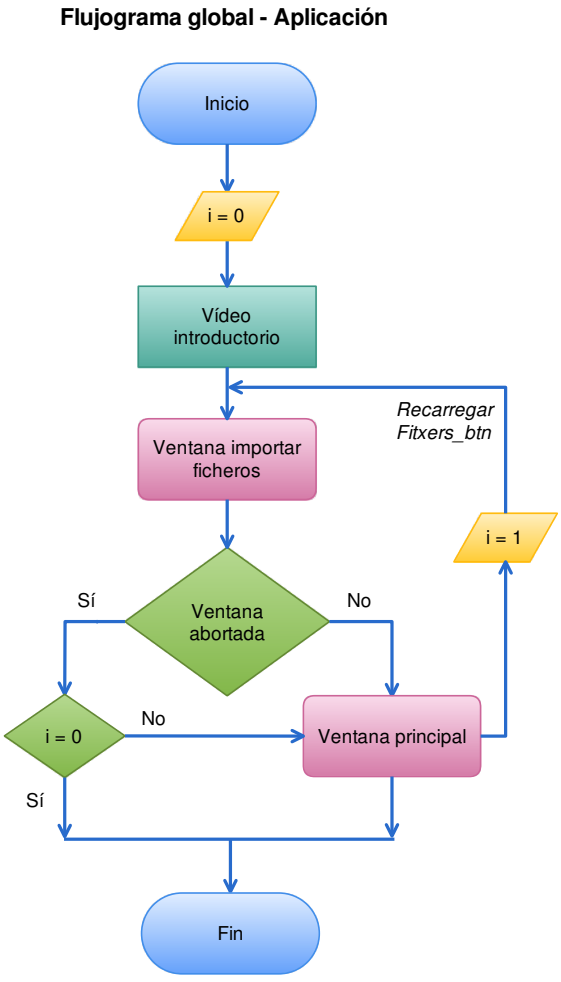


Fig. 7.15. Flujograma global – Aplicación.

7.1.2.2. Flujogramas específicos

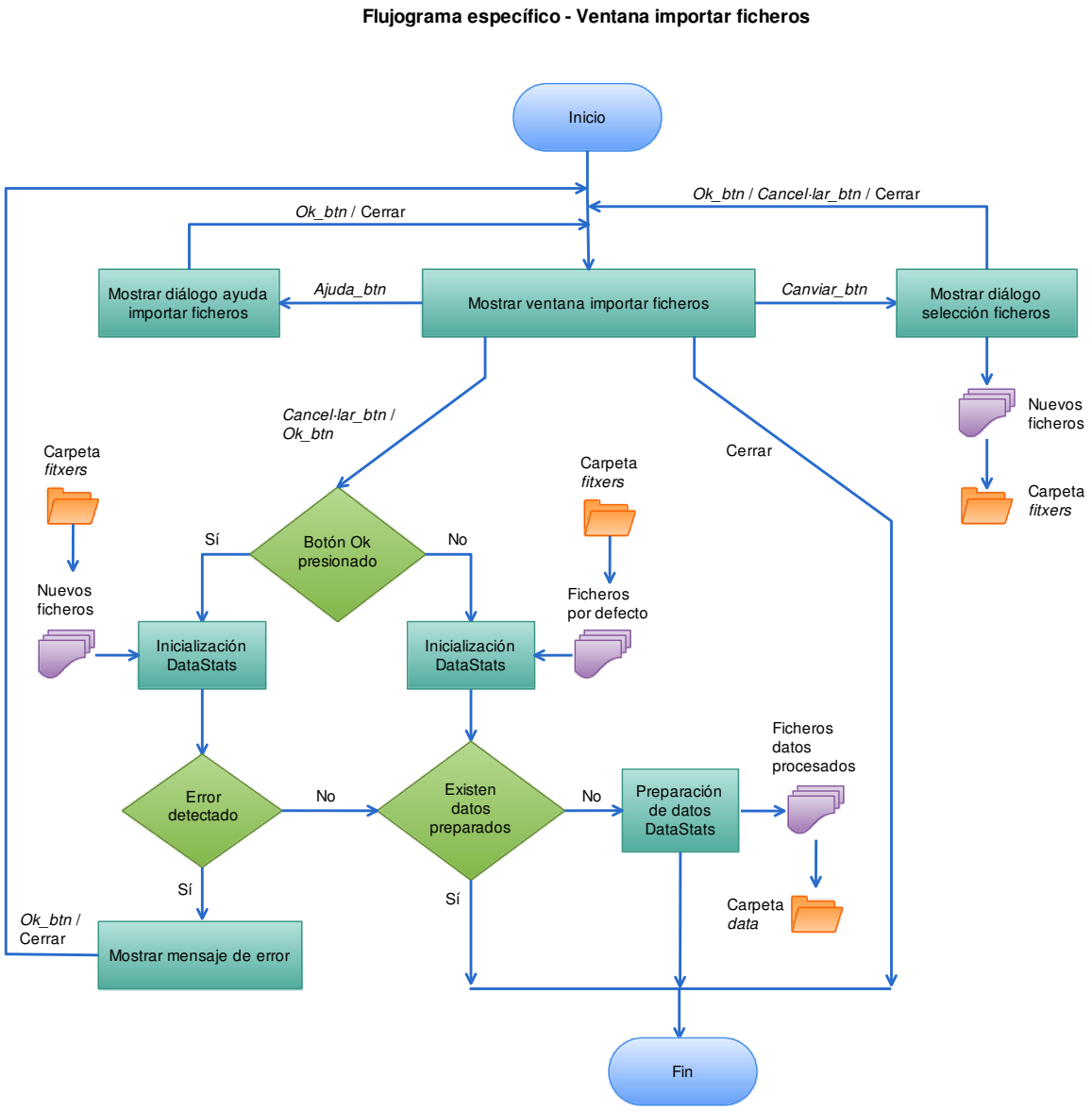


Fig. 7.16. Flujograma específico - Ventana importar ficheros.

A continuación, en la figura 7.17 se muestra el diagrama de flujo específico para la ventana principal. Nótese que éste ha sido simplificado por cuestiones puramente estéticas. Solo se ha introducido el menú *Arxiu* de la barra de menús, pues su funcionalidad transciende más que la del resto. Se ha considerado oportuno ya que tanto el menú *Veure* como el menú *Ajuda* no comportan un flujo de información especialmente relevante a la aplicación. No obstante, la figura 7.18 representa estos elementos omitidos para hacerse

una idea de cómo influyen en el funcionamiento lógico de la aplicación. En esta figura aparecerán nuevamente dos rectángulos morados que corresponden a los diálogos de ayuda *Tutorial PyPRISMA* y *Sobre PyPRISMA.* En este caso no se desarrollan por su simplicidad y además permite reflejar su comportamiento no modal respecto a la aplicación.

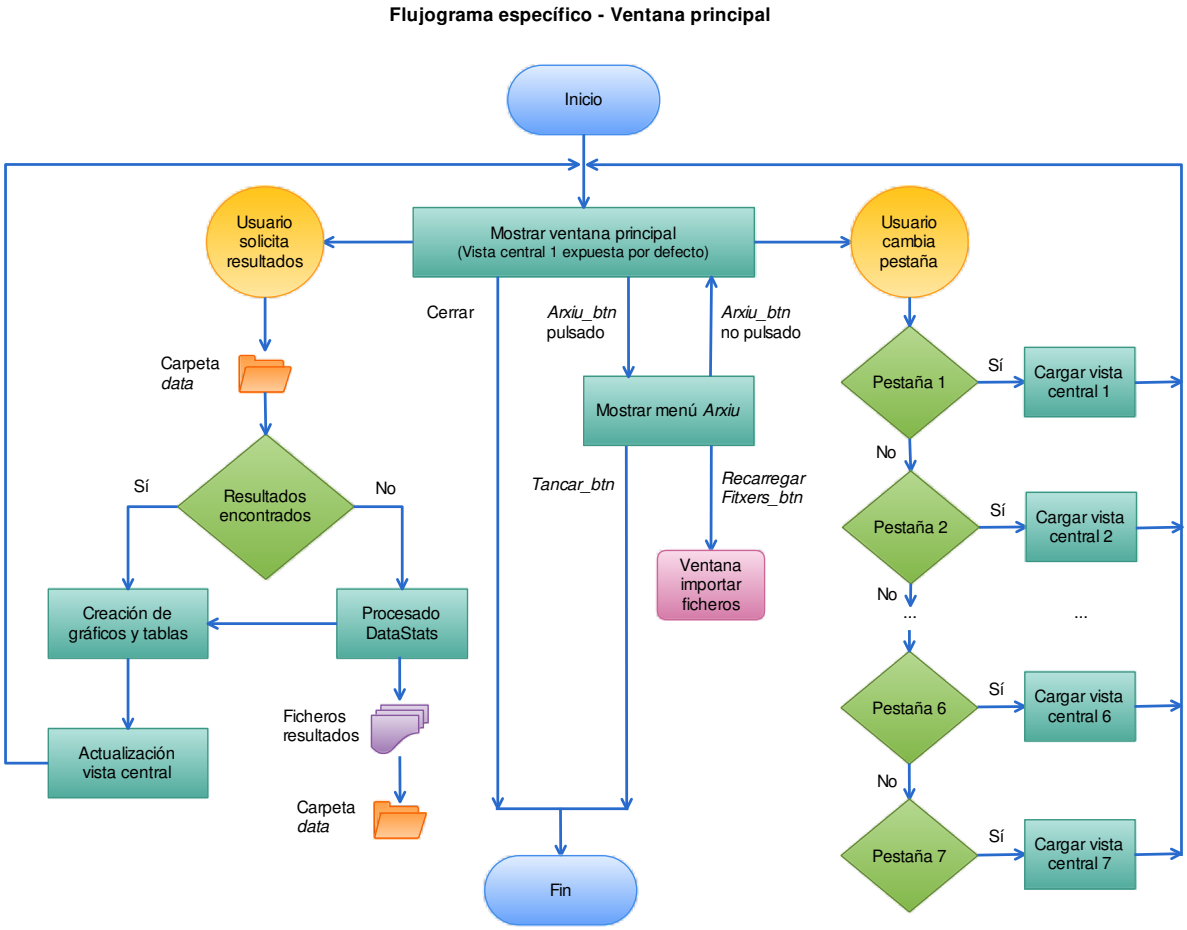


Fig. 7.17. Flujograma específico- Ventana principal.

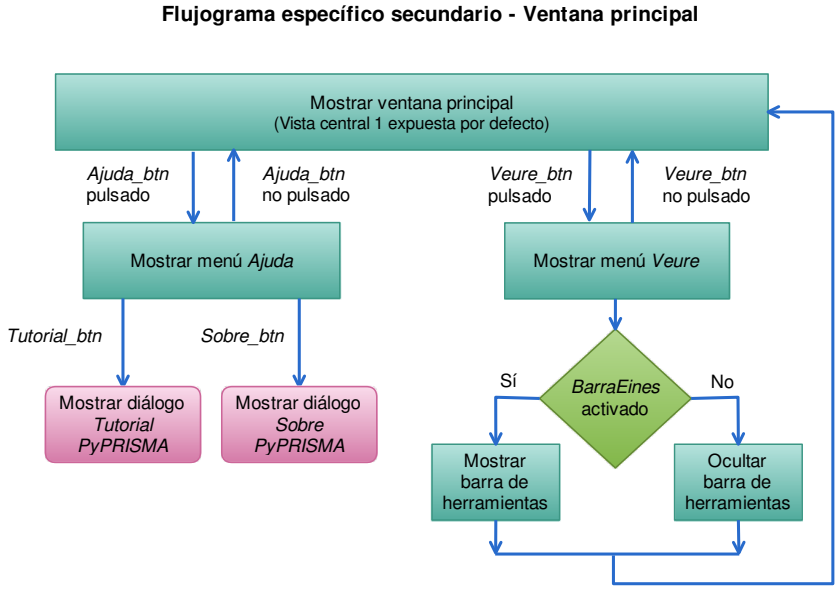


Fig. 7.18. Flujograma menús – Ventana Principal

7.2. Librerías

La implementación de la GUI de la aplicación ha requerido de diversas bibliotecas de programación. Por un lado, para la creación de los elementos y elaboración de la GUI se ha empleado PyQt. Por otro lado, para la visualización de gráficos ha sido necesario el uso de las librerías Matplotlib y Seaborn.

7.2.1. PyQt

PyQt es un binding de la biblioteca gráfica Qt para Python, es decir, una adaptación de Qt para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita; nativamente C++. Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario. Fue desarrollada por la firma británica *Riverbank Computing* y está disponible para Windows, GNU/Linux y Mac OS X bajo diferentes licencias [10]. La versión que se ha utilizado es PyQt4.

Las principales tecnologías multiplataforma en que se basan los frameworks de Python incluyen GTK, Qt, Tk y wxWidgets, aunque muchas otras tecnologías proporcionan bindings para Python activamente mantenidos. No obstante, se ha decidido trabajar con PyQt debido a ciertas características del framework Qt que se amoldan a las necesidades de este proyecto.

En primer lugar, como se ha comentado con anterioridad, se ha desarrollado la aplicación con Mac OS X. El hecho de que Qt sea una biblioteca multiplataforma disponible para el entorno operativo con el que se trabaja resulta un primer aspecto decisivo en la elección de la misma. Adicionalmente, esta característica significa que el mismo código se utiliza para generar la interfaz de usuario en Windows, Mac OS X y Linux. Esto facilitará enormemente la adaptación de la aplicación para funcionar en estos sistemas operativos; tarea que podría ser interesante en la futura continuación de este proyecto a través de terceras personas.

En segundo lugar, se ha mencionado la intención de diseñar una aplicación que disponga de una interfaz de usuario agradable e intuitiva. Qt dispone de múltiples herramientas de desarrollo visuales, editores, que permiten crear interfaces bonitas y fluidas. Estos programas facilitan la tarea de dimensionar y modificar características de los elementos que componen la UI. Entre estos programas destaca el Qt Designer, específicamente encargado del diseño de la interfaz. Si se desea un entorno más completo puede optarse por el Qt Creator. Éste es un IDE multiplataforma que dispone de un editor avanzado de código junto con un diseñador de interfaces gráficas de usuario.

Por último, Qt es gratuito y dispone de una gran comunidad online. La documentación existente para PyQt es muy extensa y dispone de múltiples tutoriales y ejemplos que aseguran el aprendizaje autodidacta del paquete de herramientas. Lo mismo sucede para las herramientas y programas que ofrece Qt para el desarrollo rápido de aplicaciones, aspecto positivo en el caso de que se hubiese optado por emplear alguna de ellas.

Cabe mencionar que pese a que se inició el proyecto con la intención de utilizar el Qt Designer para la elaboración de la GUI, finalmente se desechó esta opción. La razón de tal cambio fue el reto de desarrollar manualmente el software del programa. Esto permite un aprendizaje más profundo en cuanto a creación y programación de aplicaciones, que es la motivación principal de este proyecto. La aplicación se ha desarrollado empleando únicamente el IDE Spyder.

7.2.2. Matplotlib

Matplotlib es la librería más popular en Python para visualizaciones y gráficos. Sólo utiliza código compatible BSD (Berkeley Software Distribution), y se publica bajo la licencia

de PSF (Python Software Foundation) [11]. La versión utilizada es Matplotlib 1.4.3.

Se ha decidido trabajar con este paquete de herramientas por los siguientes motivos:

En primer lugar, es fácil de aprender y manipular. La documentación existente es extensa, repleta de tutoriales y ejemplos que agilizan dicho aprendizaje. También proporciona un gran control sobre cada uno de los elementos de las figuras, como puede ser su tamaño, el trazado de sus líneas, títulos y leyendas.

En segundo lugar, permite crear gráficos y figuras de gran calidad que pueden ser guardados en varios formatos, por ejemplo: PNG, PDF, SVG, EPS y PGF [12]. Esto puede interesar al usuario razón de ser a la hora de almacenar los gráficos que proporcione la aplicación.

Por último, se integra cómodamente con IPython, lo que proporciona un ambiente confortable para las visualizaciones y la exploración de datos interactiva.

7.2.3. Seaborn

Seaborn es una librería de visualización de datos en Python basada en Matplotlib. Se publica bajo la licencia BSD-3 [13]. La versión que se ha utilizado es Seaborn 0.6.0.

Se ha decidido trabajar con este paquete de herramientas ya que brinda funcionalidad para hacer gráficos estadísticos atractivos y explicativos. Seaborn emplea Matplotlib y dispone de herramientas para la elección de paletas de colores y temas integrados para mejorar el diseño de sus gráficos. Además, ofrece soporte para graficar estructuras de datos en Pandas. Con todo, resulta el complemento perfecto a ambas librerías.

7.3. Desarrollo

En esta sección se explican aspectos básicos de los scripts y las herramientas que se ha empleado para obtener el diseño de la GUI y el funcionamiento de la aplicación explicados en la sección anterior. Por script se entiende un fichero que contiene una lista de instrucciones o órdenes, escritas utilizando un lenguaje informático; en este caso Python.

En el anexo B se puede encontrar más información acerca de la estructura del

directorio que contiene los ficheros y subcarpetas necesarias para el funcionamiento de la aplicación. Además, en él aparece el código de cada script desarrollado.

7.3.1. Estructura y contenido general scripts

Uno de los objetivos específicos del proyecto es crear una estructura interna del programa ordenada y el código del software claro e inteligible. Todo ello para facilitar la modificación o mejora de la aplicación al futuro usuario operador. Consecuentemente, todos los scripts presentan una estructura similar y las variables que definen los elementos de la GUI se han definido según el objeto que representan. Los nombres de los ficheros Python también se escogen en concordancia con su contenido o clases que implementan.

Por lo que respecta a la estructura de los scripts, en todos ellos se puede distinguir dos partes:

1. *Cabecera*: En ella se importan las librerías o los ficheros que serán necesarios en función de los elementos que se definan en el cuerpo del script.
2. *Cuerpo*: En él se desarrolla el código de los programas o las clases que definen la vista y funcionamiento de la aplicación.

Se ha mencionado anteriormente que se empleará PyQt4 para implementar la UI. De hecho, la gran mayoría de los elementos que forman la interfaz gráfica provienen de esta librería. Es por esto que en todos los scripts aparecerá en la cabecera la siguiente importación:

```
from PyQt4 import QtGui, QtCore
```

Las clases disponibles en PyQt4 están divididas en siete módulos según la naturaleza de la funcionalidad que introduzcan. En el programa objeto del proyecto se ha empleado dos de éstos:

1. El módulo QtGui, que contiene componentes gráficos y sus respectivas clases. Por ejemplo incluye botones, ventanas, barras de estado, barras de herramientas, sliders (controles deslizantes), bitmaps (mapas de bits), colores, fuentes, etc.
2. El módulo QtCore, que contiene el corazón de funcionalidades no gráficas. Se usa para trabajar con tiempos, archivos y directorios, tipos de datos variables, señales,

streams (flujos), procesos y otros. Por stream, o flujo, se entiende a una conexión entre un programa y una fuente o destino de datos.

Muchos de los elementos que emplea la aplicación son instancias de clases derivadas de las que proporcionan estos módulos. Una instancia es un objeto con unas características específicas que pertenece a una clase en particular, es decir, comparte el prototipo y métodos con otras instancias de la misma clase, pero el valor que toman sus variables internas está definido en particular. En cuanto a las clases de las que proceden estas instancias, se ha comentado que son clases derivadas de las que proporcionan los módulos QtGui y QtCore. Una clase derivada o clase hija se define como una clase que hereda de otra clase, generalmente llamada clase base o clase madre. Éste procedimiento se denomina herencia simple. En cambio, si una clase derivada proviene de más de una clase base, se hablará de herencia múltiple. No obstante, por lo que respecta a el programa objeto del proyecto, únicamente aparecerá el concepto de herencia simple.

La propiedad de herencia que incorporan las clases permite crear otras nuevas a partir de clases existentes, conservando las propiedades de la clase original y añadiendo otras nuevas. Esto ha resultado imprescindible cuando se ha querido personalizar el funcionamiento o atributos de las clases que ofrecen los módulos de PyQt4.

En las siguientes secciones se empleará el formato *Módulo.Clase* para indicar el módulo del que proviene la clase que quiera explicarse. Por ejemplo: *GtGui.QWidget* indica la clase QWidget del módulo QtGui. Por otro lado, aparecerá repetidamente el concepto de widget ya que es uno de los componentes más utilizados para implementar la interfaz gráfica de usuario. En el contexto de la programación de aplicaciones de escritorio, un widget o gadget es un elemento de interacción en una GUI, como por ejemplo un botón o una barra deslizante. De él emanan la mayoría de componentes gráficos y su objetivo es dar acceso a funcionalidades y proveer información visual actualizada.

Por último, conviene explicar el mecanismo de señales y slots con el que se construye el sistema de procesamiento de eventos en PyQt4.

Las señales y los slots se usan para las comunicaciones entre los objetos de la interfaz. En general, la señal (signal) parte de un objeto emisor y llega a un objeto receptor. El objeto receptor decide si ejecuta un slot y finaliza el proceso, o emite una nueva señal que propaga el evento hacia otro objeto receptor. Estos slots pueden ser slots predefinidos

de PyQt o cualquier método Python creado. Para establecer una conexión entre una señal y un slot se utiliza el método *connect()*. La sintaxis que se empleará es *signal.connect(slot).*

7.3.2. PyPRISMA.py

Por medio de este script se ejecuta o arranca la aplicación. Contiene el mainloop o bucle principal del programa.

En primer lugar se crea el objeto aplicación, importado del módulo QtGui e imprescindible en toda aplicación PyQt4. Seguidamente se crea el vídeo introductorio de la app y se muestra en pantalla. El vídeo es una instancia de la clase *MovieSplashScreen*, definida en el fichero *widgets.py.* Ésta es una clase derivada de *QtGui.QSplashScreen* que describe la ventana que aparece mientras se cargan los elementos de un programa; ya sea para indicar al usuario que se están cargando y darle información durante el tiempo de espera o por pura autopromoción. Una vez se finaliza la reproducción del vídeo, se crea una instancia de la clase derivada *MainWindow,* que describe la ventana principal de la aplicación. Acto seguido se muestra en pantalla a modo pop-up mediante los métodos predefinidos *show()* sucedido de *raise_().*

Finalmente, se crea el mainloop de la aplicación y se inicia la gestión de eventos. El bucle principal es el encargado de recibir las señales y eventos que provengan de la ventana de escritorio y enviarlos a los widgets de la aplicación. Éste finaliza cuando se llama al método *exit()* o se cierra la ventana principal. Se ha empleado el método *sys.exit()* para asegurar un cierre limpio y seguro.

7.3.3. dialogs.py

Este script contiene la implementación de los diálogos propios de la aplicación. En este sentido, existen diálogos que aparecerán en la misma pese a que su código no se presente explícitamente en este fichero. Esto es así gracias a la existencia de diálogos PyQt4 prediseñados. Un ejemplo de éstos es la clase *QtGui.QFileDialog*, que describe y abre una ventana para abrir o guardar un archivo.

Hay cuatro diálogos creados a medida para el programa y son clases derivadas de *QtGui.QDialog*.

Diálogo 1.- *SelectFilesDialog*

Éste representa la ventana de inicio de la aplicación y sirve para introducir los 6 ficheros principales de datos sobre los que la clase *DataStats* realiza el análisis estadístico.

Para cada fichero hay un recuadro con la ruta de los datos que se emplean por defecto. El recuadro es un *QtGui.QTextEdit*, widget empleado para mostrar o capturar una información tipo texto del usuario. A su derecha se dispone de un botón que al ser presionado reconduce a un *QtGui.QFileDialog* donde seleccionar los nuevos ficheros de datos. Cada botón es un widget o instancia de la clase *QtGui.QPushButton.*

Se tiene tres botones más: *Ajuda_btn*, *Cancel_btn* y *Ok_btn*. Al presionar *Ajuda_btn* se envía una señal que inicializa y muestra en la pantalla el diálogo *AjudaDialog.* Seleccionando *Cancel_btn* o *Ok_btn* se inicializa la clase *DataStats*, con la diferencia de que *Ok_btn* modifica la localización de los archivos de datos principales a la nueva ruta introducida por el usuario dentro de la clase *DataStats*. Además, en esta clase tiene lugar la estrategia de detección de errores en la ruta o en el formato de los ficheros introducidos. El código que se presenta a continuación se encarga de esta tarea:

```
try:
    self.dataStats = self.DataStats()
    self.hide()

except (pd.parser.CParserError, ValueError):
    #Usuario introduce ficheros incorrectos --> Error 1
    QtGui.QMessageBox.critical(self, 'Crític',
    '''El format dels fitxers introduïts és incorrecte.
    A 'Ajuda' tovaràs informació sobre el format adequat.
    ''', QtGui.QMessageBox.Ok)

    self.restauraFitxers()

except OSError:
    #Usuario introduce una ruta inválida --> Error 2
    QtGui.QMessageBox.critical(self, 'Crític',
    '''Has d'introduir una ruta vàlida dels fitxers''',
    QtGui.QMessageBox.Ok)

    self.restauraFitxers()

except Exception as e:
    #Se produce un error desconocido --> Error 3
    print(e)
    QtGui.QMessageBox.critical(self, 'Crític',
    '''Hi ha un error desconegut als fitxers d'entrada.
    Si us plau, revisi el contingut dels fitxers introduïts.
    A 'Ajuda' tovaràs informació sobre el format adequat.
    ''', QtGui.QMessageBox.Ok)

    self.restauraFitxers()
```

Cuando la inicialización de la clase *DataStats* no se completa es debido a la aparición de un error. Lo que se hace es interceptarlo, enviar un mensaje de alerta y restaurar la localización de los ficheros por defecto. Lo segundo se consigue con un widget tipo *QtGui.QMessageBox.critical,* que permite mostrar mensajes de notificación críticos al usuario. El pseudocódigo siguiente traduce el contenido del fragmento de código recién expuesto para facilitar la comprensión del mismo.

Intenta:
    Inicializar la clase DataStats.
    Ciérrate.

Excepto error tipo 1:
    Muestra mensaje de error tipo 1.
    Restaura los ficheros de la clase DataStats.

Excepto error tipo 2:
    Muestra mensaje de error tipo 2.
    Restaura los ficheros de la clase DataStats.

Excepto otro error:
    Escribe el tipo de error en la terminal.
    Muestra mensaje de error tipo 3.
    Restaura los ficheros de la clase DataStats.

Como puede observarse, el cierre del *SelectFilesDialog* se realiza únicamente una vez se ha completado la inicialización y preparación de datos en *DataStats* y se está listo para iniciar la ventana principal de la aplicación. De lo contrario, se vuelve al diálogo de importación de ficheros.

Diálogo 2.- *AjudaDialog:*

Éste representa el diálogo que aparece al presionar *Ajuda_btn* en la ventana de importación de ficheros. Mediante un *QtGui.QTextEdit* muestra el contenido del fichero *ajuda.txt,* almacenado en la carpeta *fitxers* del directorio desde el que se ejecuta la aplicación.

Diálogo 3.- *HowToDialog:*

Éste representa el diálogo que aparece al presionar la opción *Tutorial PyPRISMA* en el menú *Ajuda* de la ventana principal de la aplicación. Muestra un conjunto de fotografías a modo de presentación, que el usuario controla mediante las flechas derecha e izquierda del

teclado para navegar a la siguiente o anterior diapositiva, respectivamente.

La imagen se imprime en la pantalla mediante la creación de un *QtGui.QLabel,* que representa una etiqueta. En su interior, se agregar un *QtGui.QPixmap*, que constituye una imagen o mapa de píxeles, que contiene la imagen, o diapositiva, que se desea mostrar.

La actualización de las diapositivas se ha conseguido definiendo una función que captura la señal que emite una tecla al ser presionada. Si esta tecla es la flecha derecho o izquierda, se refresca la imagen expuesta en el diálogo consecuentemente.

Las imágenes están almacenadas en la carpeta *icons/tutorial* y se han almacenado con un nombre correspondiente a su posición en la exposición del tutorial. Este pequeño hecho minimiza y simplifica el código.

Diálogo 4.- *AboutDialog:*

Éste representa el diálogo que aparece al presionar la opción *Sobre PyPRISMA* en el menú *Ajuda* de la ventana principal de la aplicación. Mediante varios *QtGui.QLabel* se organiza y expone la información que se quiere ofrecer con el diálogo.

7.3.4. mainwindow.py

Este script contiene la implementación de *MainWindow*, que es una clase derivada de *QtGui.QMainWindow* y describe la ventana principal de la aplicación.

Está compuesta por 4 áreas o secciones:

1. *MenuBar*: corresponde a la barra de menús explicada en la sección de diseño. Se crea mediante el método propio de *self.createMenuBar(),* siendo *self* la propia *QtGui.QMainWindow.*
2. *Status Bar:* corresponde a la barra de estado explicada en la sección de diseño. Se crea mediante el método predefinido de *self.statusBar().*
3. *Central Widget*: corresponde a la vista central explicada en la sección de diseño. Para ello se crea el widget central *self.mw* que es una instancia de la clase derivada *MainWidget,* definida en el fichero *mainwidget.py,* y se fija mediante el método predefinido *self.setCentralWidget(self.mw).*
4. *ToolBar*: pese a que no se ha incluido en las imágenes de la sección de diseño para optimizar el espacio, se ha implementado una barra de herramientas con las

acciones que incorpora el menú. Se crea mediante el método propio de *self.createToolBar()*. Puede modificarse su posición a gusto del usuario, siempre y cuando sea un margen de la ventana principal (superior, derecho, inferior o izquierdo).

En la siguiente imagen se señalan estas 4 partes de la *MainWindow*.

Fig. 7.19. Partes *MainWindow* - Ventana principal.

Se ha mencionado anteriormente que una vez se finaliza la reproducción del vídeo introductorio, se crea una instancia de *MainWindow,* Esto es así porque la inicialización de la MainWindow incluye la del diálogo de importación de archivos, su aparición en pantalla y, una vez éste ha sido cerrado, la inicialización de la propia ventana principal.

7.3.5. mainwidget.py

Este script contiene la implementación de *MainWidget*, que es una clase derivada de *QtGui.QWidget* y describe el widget central de la ventana principal de la aplicación.

El *MainWidget* ha de incluir las cuatro vistas centrales explicadas en la sección de diseño. Esto se ha conseguido mediante la clase *QtGui.QTabWidget,* que permite crear diferentes pestañas. A cada pestaña le corresponde un *QtGui.QWidget* que contiene todos los elementos de la respectiva vista central.

Pese a que solo se ha hablado de cuatro vistas, en realidad se ha implementado cinco. La primera de ellas, *Basics,* está desactivada. Esto se ha hecho considerando que uno de los objetivos es facilitar la tarea de continuación del proyecto. El usuario interesado en añadir funcionalidad a la aplicación tendrá una pestaña lista para introducir el contenido que desee. A modo de ejemplo, esta vista contiene un botón central. Para reactivar la vista basta con eliminar el símbolo # que se encuentra frente al métodos *ciertos métodos.*

Antes de explicar las herramientas empleadas para las cuatro vistas restantes, es preciso comentar la estrategia con la que se define la apariencia de estos widgets. Se ha mencionado en el informe que se puede modificar el tamaño de las ventanas sin perjudicar la organización y apariencia de los elementos que contiene, es decir, son escalables. Esta característica se consigue mediante layouts o esquemas de disposición.

Es posible diseñar una interfaz de usuario utilizando distintas combinaciones de layouts, es cuestión de tiempo y experiencia poder encontrar aquella que sea más fácil y rápida de crear o la que se adapte mejor a las necesidades de la aplicación. Para mantener el código uniforme y facilitar su comprensión, los elementos de todas las ventanas o vistas emplean el mismo esquema de diseño, basado en la anidación de layouts.

Se emplan layouts verticales y horizontales, que son instancias de *QtGui.QVBoxLayout* y *QtGui.QHBoxLayout,* respectivamente. En todas se creará un layout vertical principal que organiza el espacio que ocupan los componentes verticalmente. Si se quiere disponer los widgets horizontalmente, entonces se creará un layout horizontal secundario y éstos se situarán en su interior. Acto seguido, el layout horizontal secundario se introducirá en el layout vertical como si se tratase de un solo widget. El código y la figura que se muestran a continuación son un ejemplo de la anidación de layouts empleada en el diseño de la UI.

```
def initUI(self):

    layout_principal = QtGui.QVBoxLayout()

    texto = QtGui.QLabel('Escriba su nombre:')
    recuadro = QtGui.QTextEdit()
    cancelButton = QtGui.QPushButton('Cancelar')
    okButton = QtGui.QPushButton('OK')

    hbox1 = QtGui.QHBoxLayout()
    hbox1.addWidget(texto)
    hbox1.addStretch(1) #añade un espacio
```

```
hbox2 = QtGui.QHBoxLayout()
hbox2.addStretch(1) #añade un espacio
hbox2.addWidget(cancelButton)
hbox2.addWidget(okButton)

layout_principal.addLayout(hbox1)
layout_principal.addWidget(recuadro)
layout_principal.addLayout(hbox2)

self.setLayout(layout_principal)
```

El resultado gráfico del código anterior es el siguiente:

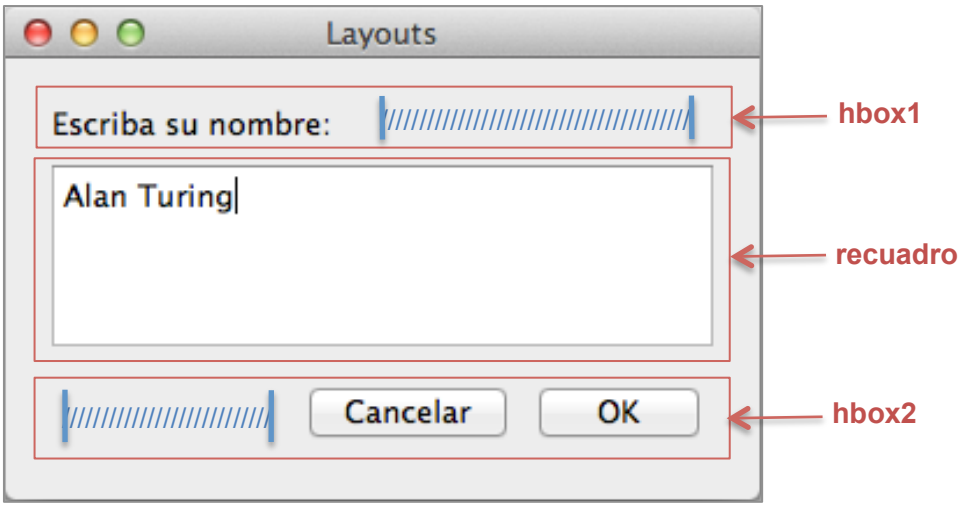


Fig. 7.20. Anidación de layouts.

Vista central 1.- *Gràfics Fase Inicial*

Debido a la simetría de esta vista, tanto en apariencia como en funcionamiento, todos los widgets y métodos necesarios que se explican a continuación aparecen por duplicado.

Para escoger el gráfico que se desee se emplea un *QtGui.QComboBox,* que es una lista desplegable de opciones. Cada gráfico dependerá de ciertas variables que se podrán escoger en otros *QtGui.QComboBox* situados debajo del primero.

Para representar el gráfico se ha creado un *FigureCanvas* o lienzo, sobre el que se sitúa una figura (el gráfico en sí) de la librería matplotlib. Las herramientas para inspeccionar o modificar el gráfico forman parte del *NavigationToolbar* o barra de herramientas de navegación del canvas. Tanto el *FigureCanvas* como la *NavigationToolbar* se importan del módulo *matplotlib.backends.backend_qt4agg*. No obstante, la barra de herramientas se ha modificado convenientemente para homogeneizar la GUI. En concreto, se personaliza el

lenguaje, los iconos y las acciones que incorpora. Además, se introduce un nuevo algoritmo para captar los atajos de teclado de ésta. Para más información, se puede consultar el anexo B.

Por último, se tiene un recuadro de texto que proporciona una descripción del gráfico que se visualice en cada caso. Este elemento es un *QtGui.QPlainTextEdit* sobre el que se ha activado el modo de solo lectura.

Cuando se selecciona un gráfico en particular, el método *self.combo_graficXY.activated[int].connect(self.accionesXY)* intercepta la señal y ejecuta el siguiente slot:

```
def accionesXY(self,index):
    self.muestraWidgetXY(index)
    self.muestraGraficoXY(index)
    self.muestraTextoXY(index)
```

Se apunta que la notación *XY* no aparece explícitamente en el código pero se emplea aquí a modo de ejemplo genérico. En la realidad *X* corresponde al número de la vista central y *Y* corresponde a la mitad del gráfico a la que hace referencia.

El método *self.muestraWidgetXY(index)* es un slot secundario encargado de enseñar el conjunto de listas desplegables necesarias para el gráfico, en función de las variables de las que éste dependa. Esto se consigue escondiendo los desplegables que no se necesitan y enseñando los que si.

El método *self.muestraGraficoXY(index)* es un slot secundario encargado de crear y enseñar el gráfico que el usuario desee. Para ello requiere del valor que el usuario ha introducido en todos las variables (o desplegables) que se muestran en pantalla. Esta información se adquiere con el método *self.desplegableXYZ.currentIndex(),* donde Z representa una variable en concreto; ya sea año de inicio, cuatrimestre o número de asignaturas suspendidas.

El método *self.muestraWidgetXY(index)* es un slot secundario encargado de mostrar en el recuadro de texto la explicación del gráfico expuesto.

Vista central 2.- *Textos Fase Inicial*

Como ocurre en la vista central 1, debido a la simetría de esta vista todos los widgets

y métodos necesarios que se explican a continuación aparecen por duplicado.

Ya se ha comentado que esta vista muestra los resultados tipo texto del procesamiento de datos de la fase selectiva. Se tiene unos resultados diferentes por año de inicio de la fase selectiva y por cuatrimestre. Para seleccionar estas variables se crean dos *QtGui.QComboBox.* El espacio en que se disponen los resultados es un *QtGui.QPlainTextEdit* sobre el que se ha activado el modo de solo lectura.

Cuando el usuario selecciona un valor dentro de estos desplegables, el método *self.desplegableXYZ.activated[int].connect(self.muestraTextoXY)* intercepta la señal y llama al slot *self.muestraTextoXY.* Como con la inicialización de *DataStats* estos resultados ya han sido calculados y guardados en ficheros dentro de la carpeta *data*, este método solo deberá buscar el fichero pertinente, leer su contenido e introducirlo en el recuadro de texto.

Vista central 3.- *Sèries Fase Inicial*

Esta vista incluye dos *QtGui.QGroupBox,* que representan una agrupación de widgets enmarcados en una región de la ventana. Estos elementos seleccionables que contiene son instancias a la clase *QtGui.QCheckBox.* Para mostrar los resultados gráficos se tiene un canvas sobre el que se dibuja una figura; igual que ocurría en la venta *Gràfics Fase Inicial.* También se tiene una barra de herramientas de navegación del canvas para modificar el aspecto o visualización del gráfico.

Vista central 4.- *Taules Global*

Esta vista debe mostrar los resultados tipo tabla del procesamiento de datos global. Se tiene unos resultados diferentes por titulación, curso académico y por cuatrimestre. Para seleccionar estas variables se crean tres *QtGui.QComboBox.* El espacio en que se disponen los resultados es un *QtGui.QTableView,* que es una vista bidimensional que proporciona Qt derivada de la clase *QtGui,QAbstractItemView,* idónea para la presentación de tablas de bases de datos. Cómo los datos pueden estar almacenados en clases, estructuras o tipos de fuentes de datos diversas, hace falta un modelo que capture la información adecuadamente. Ninguno de los modelos que proporciona Qt sirve para contener los DataFrames que la clase *DataStats* devuelve. Por ello se ha implementado un modelo propio, la clase *TableModel* definida en el fichero *widgets.py*.

Cuando el usuario selecciona un valor dentro de estos desplegables, el método

self.desplegable.activated[int].connect(self.muestraTabla) intercepta la señal y llama al slot que se presenta a continuación:

```
def muestraTabla(self, index):
    tit=self.combo_tit.currentIndex()
    curs=self.combo_curs.currentIndex()
    quad=self.combo_quad.currentIndex()

    cdf = self.get_data_frame(tit, curs, quad)
    self.tablemodel=TableModel(self)
    self.tablemodel.update(cdf)

    self.tableview.setModel(self.tablemodel)
```

El slot consulta el valor que toman las variables actualmente y recurre al método *get_data_frame* para obtener los resultados en formato DataFrame que proporciona *DataStats*. Si han sido consultados previamente, estos resultados se encuentran en la carpeta *data.* Si no los encuentra, entonces llama al método *crear_df_taula* de *DataStats* para que los calcule y almacene. Acto seguido se crea una instancia de la clase propia *TableModel* y se actualiza su contenido. Finalmente, se introducen los datos del modelo en la instancia de *QtGui.QTableView.*

Vista central 6.- *Sèries Global*

En ésta se exponen gráficamente los resultados que se muestran en las tabla de resultados globales, a modo de serie temporal. Se crea con los mismos elementos que la vista *Sèries Fase Inicial*.

Vista central 6.- *Consideracions*

Esta vista es la más sencilla. Simplemente se dispone de un texto mediante un *QtGui.QLabel.* Debajo del mismo se muestra una imagen de la ETSEIB. Para ello, como se ha mencionado anteriormente, hace falta crear otra etiqueta y en su interior agregar un *QtGui.QPixmap*.

7.3.6. widgets.py

Este script contiene de implementación de los widgets o artefactos que haya hecho falta crear a medida para la aplicación. Éstos son *MovieSplashScreen,* que es una clase derivada de *QtGui.QSplashScreen*, y *TableModel*, que es una clase derivada de *QtCore.QAbstractTableModel.*

8. Vías de continuación

Uno de los objetivos específicos de este proyecto es asegurar que éste pueda ser continuado en un futuro. A modo de recomendación, se presenta una lista de posibles mejoras y vías de continuación de la aplicación:

- Ampliar la funcionalidad. Añadiendo nuevos métodos a la clase *DataStats* se puede obtener todo tipo de descripción estadística de los datos. Entre los posibles estudios, sería particularmente recomendable incidir en el referente a la fase selectiva; confiriendo nuevos puntos de vista o aspectos a analizar. Adicionalmente, el análisis global podría profundizarse mediante estudios propiamente estadísticos; como son el análisis de componentes principales, análisis clúster, etc. Paralelamente, aprovechar la información contenida en el fichero de datos personales de los alumnos para realizar estudios adicionales.

- Empaquetar la aplicación. Esto consiste en crear un paquete formado por los programas ejecutables de la aplicación, así como todos los ficheros y bibliotecas de los que depende. El usuario percibe el paquete como la aplicación en sí, cuando en realidad incluye varios ficheros. Este empaquetado permite evitar los problemas de dependencias tanto al instalar la aplicación como al usarla. Además, evita la necesidad de instalar las bibliotecas que requiere para funcionar.

- Implementación para otros sistemas operativos. Modificar el contenido de los ficheros convenientemente para poder ser ejecutados en una computadora con Windows, Linux u otros.

- Aspecto de la interfaz gráfica personalizable. En este proyecto se ha priorizado la simplicidad del código y de la aplicación. Únicamente se ha introducido la funcionalidad que se ha pensado indispensable para realizar las tareas objetivo. No obstante, una mejora podría ser versatilizar el aspecto estético de la GUI mediante la introducción de temas predefinidos que el usuario final fuese capaz de escoger a su gusto.

- Internacionalización. El idioma empleado para la aplicación es el catalán ya que éste predomina en las aulas de la ETSEIB. No obstante, sería interesante incorporar la

opción de selección de lenguaje. Se recomienda comenzar por la traducción al castellano pues es la lengua oficial del país en el que se enmarca la escuela, seguido del inglés ya que es el idioma internacional dominante. Principalmente, este último interesa por los alumnos extranjeros que cursan un programa de movilidad internacional en la escuela.

- Cambio a estructura Modelo-Vista-Controlador (a partir de ahora MVC). El MVC separa la lógica de la aplicación de la interfaz de usuario mediante tres componentes: el modelo, la vista y el controlador. Esto tiene múltiples ventajas, entre ellas están el fácil mantenimiento y la versatilidad. Si por ejemplo, una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, con el patrón MVC solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original.

9. Planificación

El objetivo de la planificación del proyecto es obtener una distribución en el tiempo de las actividades que han de realizarse; de modo que permita una utilización de los recursos que minimice el tiempo y, consecuentemente, el coste del proyecto.

En los siguientes apartados se divide el proyecto en las diferentes tareas de las que consta, se definen las dependencias entre ellas, se estima la duración de cada una y finalmente se elabora la planificación global del proyecto.

9.1. Tareas

Se ha dividido el proyecto en una serie de tareas principales. Algunas de ellas se componen por otras tareas secundarias, la consecución de las cuales representa el global de la tarea en cuestión. Las tareas principales son:

A. *Definición del problema:* Se comienza investigando sobre el entorno de trabajo, delimitando los objetivos y abaste del proyecto.
B. *Análisis de antecedentes:* Estudio de mercado y documentación sobre programas ya existentes elaborados en la asignatura de Proyecto I de la ETSEIB y análisis de los mismos.
C. *Entorno de desarrollo:* Definir el OS, el lenguaje de programación y programas para la elaboración de la aplicación.
   C.1. *Documentación:* Documentación sobre los posibles sistemas operativos con los que trabajar, lenguajes de programación que utilizar, entornos de desarrollo y programas para la implementación del código del programa. Aquí se decide el entorno de desarrollo.
   C.2. *Aprendizaje:* Tanto del lenguaje de programación como de programas que se vayan a emplear.
D. *Preparación y procesamiento de datos:*
   D.1. *Documentación:* Módulos y librerías necesarias para la manipulación de los datos.
   D.2. *Aprendizaje:* Tutoriales o manuales para el uso de tales bibliotecas.
   D.3. *Preparación de datos:* Análisis exploratorio y manipulación de los datos.

D.4. Procesamiento de datos: Creación de métodos para la obtención de resultados estadísticos deseados.

E. Diseño y desarrollo de la aplicación:

E.1. Documentación: Módulos y librerías necesarias para la creación de una aplicación de escritorio.

E.2. Aprendizaje: Tutoriales o manuales para el uso de tales bibliotecas.

E.3. Diseño: Diseño gráfico de la GUI y diseño conceptual de funcionamiento.

E.4. Desarrollo: Creación de scripts y código del programa.

F. Optimización: Mejora del funcionamiento, disminución de tiempos de ejecución y corrección de errores. Pese a que esta tarea se realice también durante las subtareas D.4, E.3 y E.4; se considera una etapa final de corrección de errores una vez la aplicación esté prácticamente acabada.

G. Implementación: Empaquetar la aplicación.

H. Redacción de manuales: Redacción del manual de usuario, instrucciones y ayudas.

I. Redacción del informe: Redacción de la memoria donde se explica cómo se ha realizado la aplicación.

9.2. Planificación temporal

Para realizar la planificación temporal del proyecto se ha tenido en cuenta el período de tiempo del que se dispone y las horas que se ha de dedicar al mismo considerando su peso académico.

Por un lado, la duración del proyecto es de 4 meses. Se ha realizado en el cuatrimestre de primavera de 2014-2015, cuyo inicio es el 12 de febrero de 2015. La fecha de depósito del mismo está estipulada entorno el 20 de junio de 2015 y el 1 de julio de 2015. Como es probable que ciertas tareas duren más de lo que se estime en la planificación o surjan imprevistos, una primera medida preventiva será fijar como fecha límite el 20 de junio de 2015. En total se tienen 129 días, considerando que se trabajará en el proyecto festivos inclusive, para completar las tareas expuestas en el apartado anterior.

Por otro lado, el trabajo final de grado corresponde a 12 créditos. Cada crédito equivale a 25 horas de trabajo, lo que implica un total de 300 horas para completar el proyecto.

Si se reparten equitativamente los recursos (300 horas de trabajo personal) en los 129 días de los que se dispone, se deberá trabajar diariamente un total de 2 horas y 20 minutos aproximadamente. Se considera un total de 2 horas y 30 minutos diarios como segunda medida preventiva y para hacer más realista la planificación. Ello implicará que el total de horas que se va a dedicar al proyecto es de 322 horas y 30 minutos. Las duraciones de las actividades se han establecido de modo que el total sume las este número de horas disponibles.

Además, se ha tenido en cuenta que hay ciertas tareas que no se pueden llevar a cabo si no se han completado unas previas. Tales dependencias entre tareas y las duraciones de las mismas se exponen en la tabla 9.1.

Dependencia y duración de tareas			
ID	**Tarea**	**Dependencia**	**Duración (horas)**
A	Definición del problema	-	5
B	Análisis de antecedentes	A	5
C	Entorno de desarrollo	A	40
C.1	Documentación	A	5
C.2	Aprendizaje	C.1	35
D	Preparación y procesamiento de datos	B,C	92,5
D.1	Documentación	B,C	2,5
D.2	Aprendizaje	D.1	20
D.3	Preparación de datos	D.2	20
D.4	Procesamiento de datos	D.3	50

E	Diseño y desarrollo de la aplicación	D	120
E.1	Documentación	D	5
E.2	Aprendizaje	E.1	20
E.3	Diseño	E.2	5
E.4	Desarrollo	E.3	90
F	Optimización	E	10
G	Implementación	F	15
H	Redacción de manuales	E	5
I	Redacción del informe	H	30

Tabla 9.1. Duraciones y dependencias entre tareas.

Nótese que las dependencias se han señalado de modo que si una tarea depende de múltiples anteriores, no hace falta indicarlas todas si la anterior ya depende del resto. Indicarlas todas sería redundante. Adicionalmente, se ha indicado la duración de las tareas principales como la suma de las duraciones de las tareas secundarias para aquellas que las poseen. Esto se puede hacer gracias a la naturaleza lineal de las dependencias entre tareas.

Para obtener la planificación temporal se han considerado los aspectos que se exponen a continuación:

- Se deben respetar las duraciones y dependencias entre tareas expuestas en la tabla 9.1.
- Evitar, a ser posible, que las tareas se solapen; incluso en los casos en que no exista dependencia entre ellas. La razón es que trabajar en aspectos diferentes al mismo tiempo puede resultar contraproducente

La planificación temporal de proyecto resultante se ha representado mediante un diagrama de Gantt y puede verse en la siguiente página. La fecha de finalización del proyecto definitiva, sin imprevistos o demoras, es el 20 de junio de 2015.

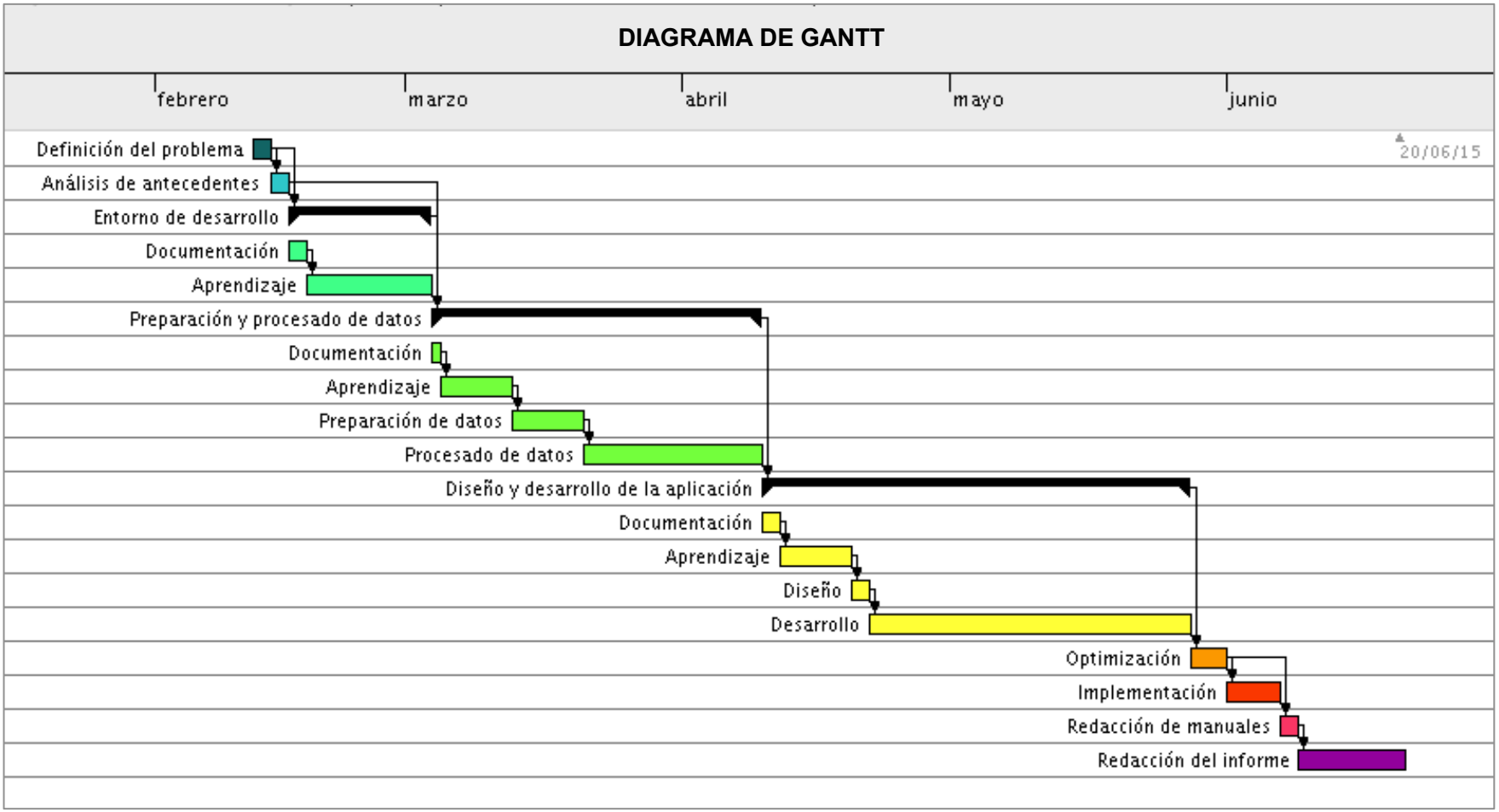


Fig. 9.1. Planificación temporal del proyecto – Diagrama de Gantt.

10. Costes

En esta sección se detallan los costes necesarios para la realización del proyecto. Se distinguen tres tipos de costes, los relacionados con las horas invertidas por cada trabajador o costes de recursos humanos, los concernientes a los recursos materiales o costes directos y los que se derivan del uso de los recursos materiales o costes indirectos.

10.1. Costes de recursos humanos

El proyecto ha estado desarrollado en su totalidad por una sola persona. No obstante, las tareas que se han realizado son de naturaleza diversa y corresponden a actividades cuyos roles pueden considerarse de diferente reconocimiento económico. Es por ello que se distinguen dos salarios: el que concierne a las actividades llevadas a cabo por un graduado en ingeniería en tecnologías industriales y las realizadas por un programador.

Para determinar dichos salarios de tienen en cuenta las tablas salariales del BOE [14] y las horas anuales trabajadas de promedio [15], al que se aplica un aumento del 70% para corresponder al salario mínimo y evitar así tener en cuenta pagas extra salariales, honorarios, etc. De tal forma, el salario de un graduado en ingeniería en tecnologías industriales queda de 22,79€/h y para un programador a 16,29€/h.

Costes de recursos humanos				
Tarea	**Ejecutor**	**Duración [h]**	**Salario [€/h]**	**Coste [€]**
Definición del problema	Ingeniero	5	22,79	113,95
Análisis de antecedentes	Ingeniero	5	22,79	113,95
Entorno de desarrollo	-	-	-	-
Documentación	Programador	5	16,29	81,45
Aprendizaje	Programador	35	16,29	570,15
Preparación y procesamiento de datos	-	-	-	-

Documentación	Programador	2,5	16,29	40,73
Aprendizaje	Programador	17,5	16,29	285,08
Preparación de datos	Programador	20	16,29	325,80
Procesamiento de datos	Programador	40	16,29	651,60
Diseño y desarrollo de la aplicación	-	-	-	-
Documentación	Programador	5	16,29	81,45
Aprendizaje	Programador	20	16,29	325,80
Diseño	Ingeniero	5	22,79	113,95
Desarrollo	Programador	80	16,29	1.303,20
Optimización	Programador	10	16,29	162,90
Implementación	Programador	15	16,29	244,35
Redacción de manuales	Programador	5	16,29	81,45
Redacción del informe	Ingeniero	30	22,79	683,70
			Precio Base	5.179,50
COSTE TOTAL			IVA (21%)	1.087,70
			TOTAL	**6.267,20**

Tabla 10.1. Cálculo desglosado de los costes de recursos humanos.

10.2. Costes directos

En los costes directos se tiene en cuenta los costes del material que se requiere para completar el proyecto.

Entre ellos destaca en primer lugar la adquisición de un equipo informático, ya que sin este no se podría haber realizado el proyecto. En concreto, se considera el coste de un ordenador portátil Macintosh v10.0 o superior. Esto es debido a que es a partir de esta versión cuando integran el sistema operativo OS X, basado en Unix, que es el que se ha utilizado. También se incluye la amortización del mismo y la suscripción mensual a Office para poder emplear sus paquetes. El IDE Spyder y el resto de programas inutilizados han sido gratuitos, así que no implican coste adicional.

El cálculo de la amortización se ha realizado mediante el método de amortización según porcentaje constante o estimación directa. Para ello se ha consultado el porcentaje de pérdida de valor que establece la Agencia Tributaria para equipos informáticos, es decir, el porcentaje de amortización vigente máximo aplicable a un ordenador. Éste es del 25% anual; y como el proyecto tiene una duración de 4 meses, el factor de amortización será 0.083.

Nótese que este gasto deducible en concepto de amortización se realiza respecto de la base imponible o precio antes de IVA; es por ello que se ha buscado los precios de los recursos materiales sin IVA, incluyendo el ordenador Mac, y se aplicará posteriormente el IVA correspondiente sobre los mismos. No se aplicará, no obstante, sobre los costes de amortización.

Por otro lado, también se incluye el coste de un manual de programación que se ha adquirido para el aprendizaje de la librería PyQt4 y creación de la interfaz de usuario. Por último, se imputan también los costes del material que será necesario para la impresión y presentación del informe final.

Costes directos			
Concepto	**Coste unitario [€/mes]**	**Uso [meses]**	**Coste [€]**
Ordenador Mac OS X 10.9	-	-	1.121,51
Paquetes Office para Mac	10,00	4	40,00
Tutorial avanzado PyQt4	-	-	15,00
Impresión, encuadernación y CD's	-	-	50,00
COSTE SUBTOTAL		Precio Base	1.226,51
		IVA (21%)	257,57
		TOTAL	1.484,08
	Coste [€]	**Factor**	
Amortización del ordenador	1.121,51	0,083	93,46
COSTE TOTAL			**1.577,54**

Tabla 10.2. Cálculo desglosado de los costes directos.

10.3. Costes indirectos

En los costes indirectos se tiene en cuenta los costes que se derivan del uso de los recursos materiales, en concreto los que derivan del uso de la computadora, como son los costes del consumo eléctrico y la conexión a internet. No se va a considerar en los costes el alquiler de un local, teléfono, suministro de agua o gas ni material de oficina complementario por la ambigüedad del uso diario de estos recursos en las horas asignadas a la elaboración del proyecto y la posibilidad de emplear los espacios de la universidad para la realización del proyecto.

Para el cálculo del coste en consumo eléctrico se han considerado únicamente las horas que implican el uso del mismo dentro de la planificación, que son todas asi que se

asignan un total de 322,5 horas.

Costes indirectos				
Concepto	**Coste unitario [€/kWh]**	**Uso [horas]**	**Consumo [W]**	**Coste [€]**
Consumo eléctrico	0,15	322,5	250	12,09
	Coste unitario [€/mes]	**Uso [meses]**		
Conexión a internet	38,82	4	-	155,28
COSTE TOTAL			Precio Base	155,28
			IVA (21%)	32,61
			TOTAL	**187,89**

Tabla 10.3. Cálculo desglosado de los costes indirectos.

Si se suman los resultados obtenidos en las secciones anteriores se obtiene el coste final del proyecto después de impuestos que se puede ver en la tabla 10.4.

Costes totales del proyecto	
Concepto	**Coste [€]**
Costes de recursos humanos	6.267,20
Costes directos	1.577,54
Costes indirectos	187,89
COSTE TOTAL	**8.032,63**

Tabla 10.4. Cálculo del coste total del proyecto.

11. Impacto Ambiental

En esta sección se van a identificar y evaluar los efectos que se derivan del proyecto sobre el medio físico y social, tanto durante la realización del mismo como los que pueden aparecer en el uso de la aplicación objeto del proyecto.

11.1. Referente a elaboración del proyecto

Por lo que al medio físico respecta, al ser un proyecto de tipo software no genera contaminantes de manera directa. Aun así se van a analizar los recursos consumidos durante la realización del proyecto y los recursos que puede consumir el usuario. No se consideran los residuos generados por el uso de consumibles como papel ya que el uso de éste ha sido casi inexistente. La única fase donde se ha utilizado es la fase de diseño de la GUI, para la elaboración de croquis y esquemas preliminares. Además, se empleó para ello papel reciclado con lo que el impacto ambiental que se deriva es negligible. Tampoco se consideran los efectos ambientales de la impresión de la memoria del proyecto [16].

Consecuentemente, el impacto ambiental provocado sobre el medio físico en la realización del presente proyecto se reduce a las emisiones de CO_2 derivadas del consumo eléctrico; tanto para iluminar el lugar de trabajo, como para hacer funcionar el ordenador portátil. Para el cómputo de dichas emisiones, se ha considerado que 1kWh produce 0,65kg de CO_2 como establece la Comisión Europea [17][18]. En la tabla 11.1 se muestra un desglose de las emisiones de CO_2 derivadas del consumo eléctrico.

Emisión de CO_2 durante la elaboración del proyecto				
Elemento de consumo	**Tiempo de consumo [h]**	**Consumo de potencia [W]**	**Consumo de energía [kWh]**	**Emisión de CO_2 [kg]**
Ordenador	322,5	250	80,63	52,41
Iluminación	100,0	90	9,00	5,85
EMISIONES TOTALES DE CO_2				**58,26**

Tabla 11.1. Emisiones de CO_2 derivadas del consumo eléctrico por desarrollo del proyecto.

Se obtiene un total de 58,26 kg de CO_2 emitidos por el consumo de electricidad a lo largo de toda la ejecución del proyecto.

Por lo que respecta al medio social, no se consideran efectos sobre el entorno sociocultural o económico. Éstos aparecerán una vez el software esté disponible y abierto al público.

11.2. Referente al uso de la aplicación

El usuario necesita un ordenador para poder utilizar la aplicación y para que estos dispositivos funcionen necesitan electricidad. Consecuentemente y como en el caso anterior, el impacto ambiental provocado sobre el medio físico por el uso del software objeto del proyecto se reduce a las emisiones de CO_2 derivadas del consumo eléctrico. A diferencia de la sección anterior, sólo se considera el consumo eléctrico derivado del uso de una computadora; ya que la iluminación que el usuario decida emplear durante el uso de la aplicación es decisión del mismo y no concierne al producto desarrollado en este proyecto.

A continuación se van a describir las consideraciones empleadas para el cómputo del consumo eléctrico anual por el uso de la aplicación una vez esté disponible para el usuario.

En primer lugar, se distinguen tres tipos de usuarios en la universidad: alumnado, Personal de Docencia e Investigación (a partir de ahora PDI) y Personal de Administración y Servicios (a partir de ahora PAS). Según el tipo de usuario se ha aproximado el tiempo anual de consulta y uso de la aplicación en relación con el interés que éste pueda desarrollar por los contenidos de la misma. Gracias a la información que se dispone en la página web de la universidad se conoce el número de individuos que forman parte de los grupos de usuarios recién mencionados.

En segundo lugar, se aplica un factor corrector al volumen total de individuos que forman cada grupo que refleje la popularidad y conocimiento del software entre éstos. Cabe mencionar que se establece un índice de conocimiento a la alza y constante. Esto se ha hecho así para ser más restrictivos en cuanto al análisis del impacto ambiental. En la realidad, el porcentaje de personas que conozcan la aplicación iría aumentando progresivamente y serían, de partida, mucho inferiores que los que se exponen en la tabla 11.2.

Por último, se recuerda que el consumo de potencia de un ordenador es de 250W y que el cálculo de las emisiones de CO_2 se realiza considerando que 1kWh produce 0,65kg del mismo.

Emisión de CO_2 durante el uso de la aplicación					
Usuario consumidor	**Unidades [personas]**	**Popularidad [%]**	**Tasa de uso [h/año]**	**Consumo de energía [kWh/año]**	**Emisión de CO_2 [kg/año]**
Alumnado	3170	50	4	1.585,0	1.030,25
PDI	410	70	10	717,5	466,38
PAS	240	5	1	3,0	1,95
EMISIONES TOTALES DE CO_2					**1.498,58**

Tabla 11.2. Emisiones de CO_2 derivadas del consumo eléctrico por el uso de la aplicación.

Se obtiene un total de 1.498,58 kg de CO_2 emitidos anualmente. Cabe destacar que este resultado pueden variar notoriamente en función del conocimiento y popularidad que adquiera el software en la escuela.

Por lo que respecta al impacto social, hay que considerar que la aplicación es de carácter informativo. Por un lado, permite al profesorado de la escuela evaluar el rendimiento académico de sus alumnos y observar aquellas asignaturas donde sus alumnos encuentran más dificultades. El conocimiento de dichas flaquezas puede emplearse para tomar medidas al respecto; ya sea proporcionando más herramientas para facilitar el aprendizaje, aligerando la carga lectiva o realizando una evaluación de los estudiantes menos severa. Por otro lado, los alumnos también pueden tomar provecho de este software; por ejemplo, un alumno de nuevo acceso puede informarse de aquellas asignaturas que tengan un índice de aprobados más bajo y tomar medidas preventivas para superarlas con éxito. Con todo, el impacto social que se deriva del proyecto en estos términos es positivo.

Conclusiones

El resultado de este proyecto es PyPRISMA, una aplicación de escritorio que realiza un análisis descriptivo en diferentes niveles de los resultados académicos de los estudiantes de GETI, GEM y GEQ de la ETSEIB. La funcionalidad que ofrece es innovadora y se diferencia de la que proporcionan otros programas implementados en la asignatura de Proyecto I.

El software creado mantiene su funcionalidad al introducir nuevos datos y es robusto frente a modificaciones del plan de estudios o errores en los ficheros de datos de entrada. Además, la optimización de los módulos del programa ha permitido minimizar el tiempo de ejecución y conseguir un funcionamiento fluido de la aplicación.

PyPRISMA se ha diseñado pensando en las expectativas del usuario razón de ser: personal docente, alumnado y terceros. Posee una interfaz de usuario homogénea e intuitiva y los gráficos que se obtienen del análisis son sencillos y de fácil interpretación. La claridad prima en lo que es la UI y se dispone de atajos de teclado para facilitar la navegación.

Además, el código se presenta de forma estructurada y contiene múltiples explicaciones para facilitar la comprensión del contenido al usuario operador. Todo esto simplifica la tarea de reprender el proyecto en un futuro, que ha sido uno de los objetivos del trabajo.

El resultado de este proyecto está enfocado y programado para un sistema operativo concreto. No obstante, al tratarse de un software libre y gracias a las herramientas existentes, puede implementarse fácilmente para otros sistemas operativos.

En resumen, se han cumplido en buena medida los objetivos planteados. Sin embargo, se considera que existen aún aspectos potenciales de mejora. Como se ha mencionado en el apartado de vías de continuación, entre ellos destacan los referentes a la profundidad del análisis descriptivo.

Para concluir este proyecto, se desea mencionar que para desarrollar algunas partes del mismo ha hecho falta una investigación intensa. Ésta ha requerido tiempo y ello ha reducido la dedicación al procesamiento de los datos y, consecuentemente, ha limitado la

funcionalidad creada. Sin embargo, esta investigación ha aportado nuevos y valiosos conocimientos a su ejecutor. Sin duda, ha sido una gran oportunidad para introducirse en el mundo de la programación y aprender sobre el desarrollo de aplicaciones.

Agradecimientos

En primer lugar, quisiera agradecer a mi tutor Lluís Solano por guiarme, aconsejarme y resolver mis dudas a lo largo del proyecto. Su disposición ha sido incondicional, incluso durante períodos de vacaciones.

En segundo lugar, dar las gracias a mi madre y hermanas; cuyo permanente apoyo y consejo han sido un elemento clave para hacer realidad este proyecto.

En tercer lugar, debo gran parte de este proyecto a aquella gente que, de forma altruista, dedica su tiempo a proporcionar ayuda para la programación de software. En concreto a Jan Bodnar, cuyos tutoriales y ejemplos han sido indispensables. Junto a éste, agradecer a la web stackoverflow {http://stackoverflow.com/} por su certera ayuda con esas inevitables dudas que surgen al desarrollar código desde cero.

Por último, debo y dedico este humilde trabajo a mi padre José y mi abuelo Isidoro. Su predisposición al trabajo y fuerza de voluntad han sido y serán siempre una gran fuente de energía e inspiración para mí.

Bibliografía

Referencias bibliográficas

[1] WARGAS, W. *Ventajas y desventajas de Windows y Mac OS X.* Abril de 2009. [https://wiltonvargas.readyhosting.com/content.asp?contentid=1778, 14 de julio de 2015]

[2] ESPINOSA, D. *Windows vs. Mac en el entorno empresarial.* Octubre de 2014. [http://www.sidertia.com/Home/Community/Blog/2014/10/06/Windows-vs-Mac-en-el-entorno-empresarial, 12 de febrero de 2015]

[3] RASO, A. *¿Qué lenguaje de programación elegir?* Febrero de 2015. [http://hipertextual.com/2015/02/elegir-lenguaje-de-programacion, 12 de febrero de 2015]

[4] SIMOES, C. *¿Conoces el Framework que utilizas?* Abril de 2014. [http://inusual.com/articulos/conoces-el-framework-que-utilizas/, 14 de julio de 2015]

[5] BLANCO, C. *Entornos de Desarrollo Integrado (IDE's) – Introducción.* Abril de 2012. [http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/, 13 de febrero de 2015]

[6] RAYBAUT, P. *Spyder Oficial Website - Spyder Documentation.* Agosto de 2012. [https://pythonhosted.org/spyder/index.html,13 de febrero de 2015]

[7] UNIVERSITAT POLITÈCNICA DE CATALUNYA. *Servei PRISMA.* [En línea]. [https://www.upc.edu/prisma, 15 de julio de 2015]

[8] ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA INDUSTRIAL DE BARCELONA. *Memòria 2012-2013*. [http://www.etseib.upc.edu/ca/lescola/2507-memories, 18 de marzo de 2015]

[9] MCKINNEY, W. *Python Data Analysis Library*. [En línea]. [http://pandas.pydata.org/, 15 de julio de 2015]

[10] SUMMERFIELD, M. *Rapid GUI Programming with Python and Qt.* Michigan: Pearson Education, Inc. 2008, p. 13-20

[11] HUNTER, J. , DALE, D. , FIRING, E. , DROETTBOOM, M. *Matplotlib – License.* [En línea] [http://matplotlib.org/users/license.html, 21 de julio de 2015]

[12] BLÁZQUEZ, E. *Gráficos 2D y 3D en Python. Matplotlib.* Mayo de 2014. [http://root404.com/blog/informatica/graficos-2d-y-3d-en-python-matplotlib/, 21 de julio de 2015]

[13] WASKOM, M. *An Introduction to Seaborn: Statistical Data Visualization.* [En línea] [http://stanford.edu/~mwaskom/software/seaborn/introduction.html, 21 de julio de 2015]

[14] BOLETÍN OFICIAL DEL ESTADO. *Resolución de 3 de marzo de 2014, de la Dirección General de Empleo, por la que se registran y publican las tablas salariales del Convenio colectivo nacional del ciclo de comercio del papel y artes gráficas*. Núm.87, 2014.

[15] BOLETÍN OFICIAL DEL ESTADO. *Orden JUS/2538/2013, de 27 de diciembre, por la que se modifica la Orden JUS/615/2012, de 1 de marzo, en cuanto al cómputo anual de trabajo efectivo y se aprueba el calendario laboral para el personal de la Administración de Justicia*. Núm.14, 2014.

[16] GONZÁLEZ, M. *Consideraciones ambientales para los PFC.* Dpto. Proyectos de Ingeniería – UPC. Junio de 2011.

[http://www.etseib.upc.edu/docs/-_Informaci_Acadmica/-_Portal_de_docencia/Projectes_i_Treballs/Guies/Consideracions_ambientals.pdf, 16 de julio de 2015]

[17] GENERALITAT DE CATALUNYA - OFICINA CATALANA DEL CAMBI CLIMÀTIC. *Guía práctica para el cálculo de emisiones de Gases de Efecto Invernadero (GEI).* Marzo de 2012.

[http://canviclimatic.gencat.cat/web/.content/home/politiques/politiques_catalanes/la_mitigacio_del_canvi_climatic/guia_de_calcul_demissions_de_co2/120301_guia_practica_calcul_emissions_rev_es.pdf, 16 de julio de 2015]

[18] MORIANA, J. *Desarrollo de software estadístico en Python para el análisis de material arqueológico de Oriente Próximo y Medio.* Barcelona: Escola Tècnica Superior d'Enginyeria Industrial de Barcelona. Enero de 2015, p. 76.

Bibliografía complementaria

En este apartado se muestra la bibliografía y páginas web utilizadas para la desarrollo del código de la aplicación, pero que no se citan en el proyecto ya que se ha utilizado para obtener conocimientos de programación. También se incluye la página web

donde se han obtenido los iconos para la interfaz gráfica de usuario.

[1] BODNAR, J. *PyQt4 Tutorial*. Enero de 2012. [http://zetcode.com/gui/pyqt4/]

[2] BODNAR, J. *Advanced PyQt4 Tutorial.* Diciembre de 2013. [http://zetcode.com/ebooks/advancedpyqt4/]

[3] http://stackoverflow.com/

[4] http://pandas.pydata.org/pandas-docs/stable/index.html

[5] http://matplotlib.org/index.html

[6] http://stanford.edu/~mwaskom/software/seaborn/

[7] http://pyqt.sourceforge.net/Docs/PyQt4/classes.html

[8] http://www.flaticon.com/