ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA
Universidad de Córdoba

Implementación de una Tarea de Valoración para Diferentes Dominios de Conocimiento en Python

Asignatura de Ingeniería Sistemas Software Basados en Conocimiento

Curso 2019/20

Andrés Salinas Lima

Índice General

Índice Detallado

Todos los archivos de este proyecto (incluido el vídeo-defensa) están disponibles en:

http://www.uco.es/~i52salia/issbc/#Trabajo

INTRODUCCIÓN

Para poner en práctica los conocimientos adquiridos en la parte teórica de este trabajo implementaremos nuestra propia Tarea de Valoración. En concreto, hemos desarrollado completamente desde cero un programa valorador de requisitos con PyQt4 siguiendo la arquitectura Modelo-Vista-Controlador. Permite llevar a cabo la Tarea de Valoración de CommonKADS sobre diferentes dominios de conocimiento.

El programa cuenta una interfaz gráfica para que el usuario pueda introducir el valor de cada requisito y comprobar el resultado de la valoración de forma cómoda.

Hemos seguido una aproximación diferente a la de años anteriores al crear un valorador de requisitos en vez de un valorador de criterios. Cada caso (dominio de conocimiento) cuenta con una serie de requisitos que serán evaluados de forma independiente: cada requisito es evaluado como "Aprobado" o "Rechazado" independientemente del resto y si uno solo de ellos es rechazado todo el caso será rechazado. Esto permite usar la aplicación en diferentes dominios de conocimiento que representan situaciones que se dan en la vida real, como veremos en los ejemplos incluidos.

Los casos y todos sus requisitos se cargan a través de un fichero JSON cuyo formato analizaremos más adelante. Esto permite que cualquier usuario pueda crear nuevos casos sin necesidad de que tenga conocimientos sobre Python. De hecho, el formato es tan amigable que incluso un usuario sin conocimientos de programación y que nunca haya visto un fichero de este tipo podría entender su contenido.

El programa trabaja con tres tipos de requisitos:

- Requisito Booleano: Su valor es *True* o *False*. El requisito será valorado como "Aprobado" si el valor introducido coincide con el *valor_deseado* indicado en el JSON.

- Requisito Porcentaje: Su valor es un porcentaje (0% - 100%) expresado con un número decimal (0.0 - 1.0). El requisito será valorado como "Aprobado" si el valor introducido se encuentra entre los valores *valor_minimo* y *valor_maximo* especificados en el JSON.
- Requisito Numero: Su valor es un número (entero o decimal). El requisito será valorado como "Aprobado" si el valor introducido se encuentra entre los valores *valor_minimo* y *valor_maximo* especificados en el JSON.

GUÍA DE USUARIO

PRIMEROS PASOS

Para iniciar el programa simplemente navegamos hasta la carpeta *src* y ejecutamos:

```
$ python valorador.py
```

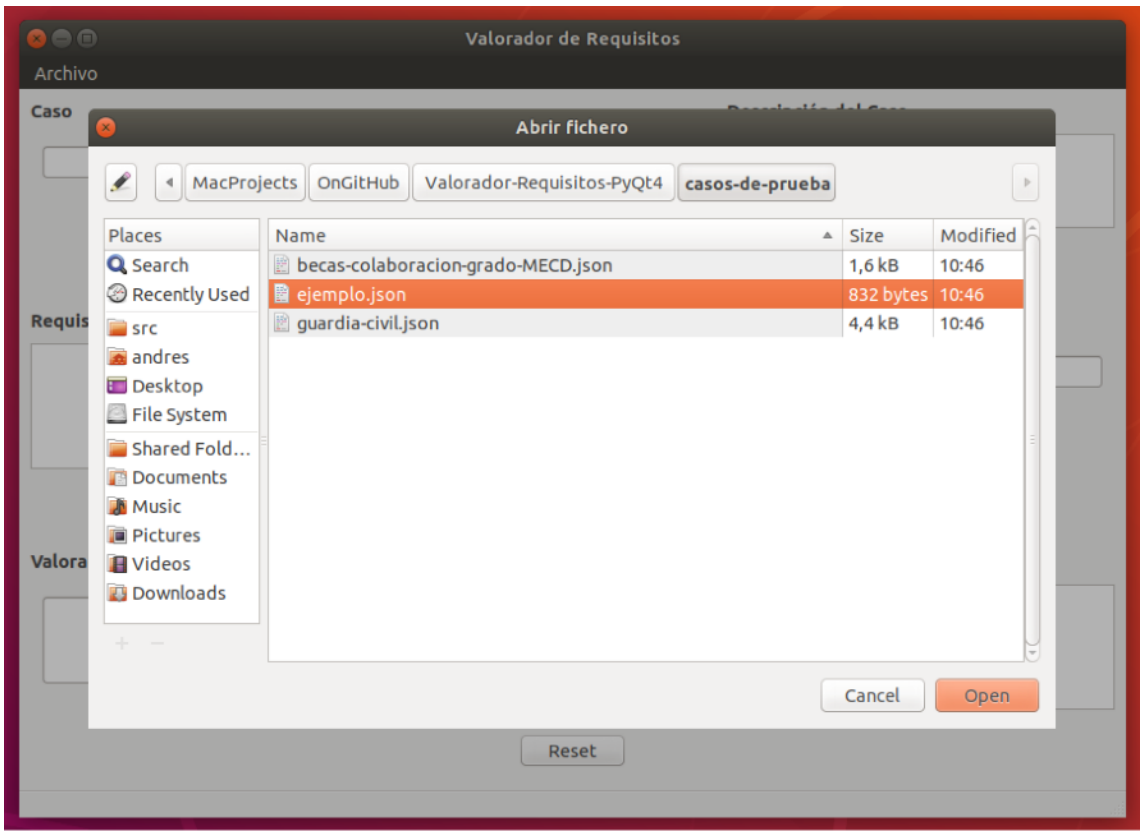
Tenga en cuenta que deberá tener instalada la librería PyQt4 para Python2 para que el programa funcione. Este es el único requisito.

Una vez ejecutado ese comando nos encontraremos con la ventana principal de la aplicación:

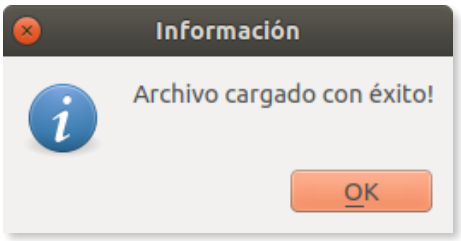
PRINCIPALES FUNCIONES

CARGAR CASO

El primer paso para empezar a trabajar con la aplicación será abrir un caso. Para ello haremos click en el botón "*Abrir Caso*" o usaremos el atajo de teclado "*Ctrl + O*". Esto nos abrirá una ventana que nos permitirá seleccionar el fichero a que deseamos abrir.



Una vez seleccionado el fichero a abrir, y si todo va bien, se nos mostrará un mensaje confirmándonos que el fichero se ha abierto correctamente:

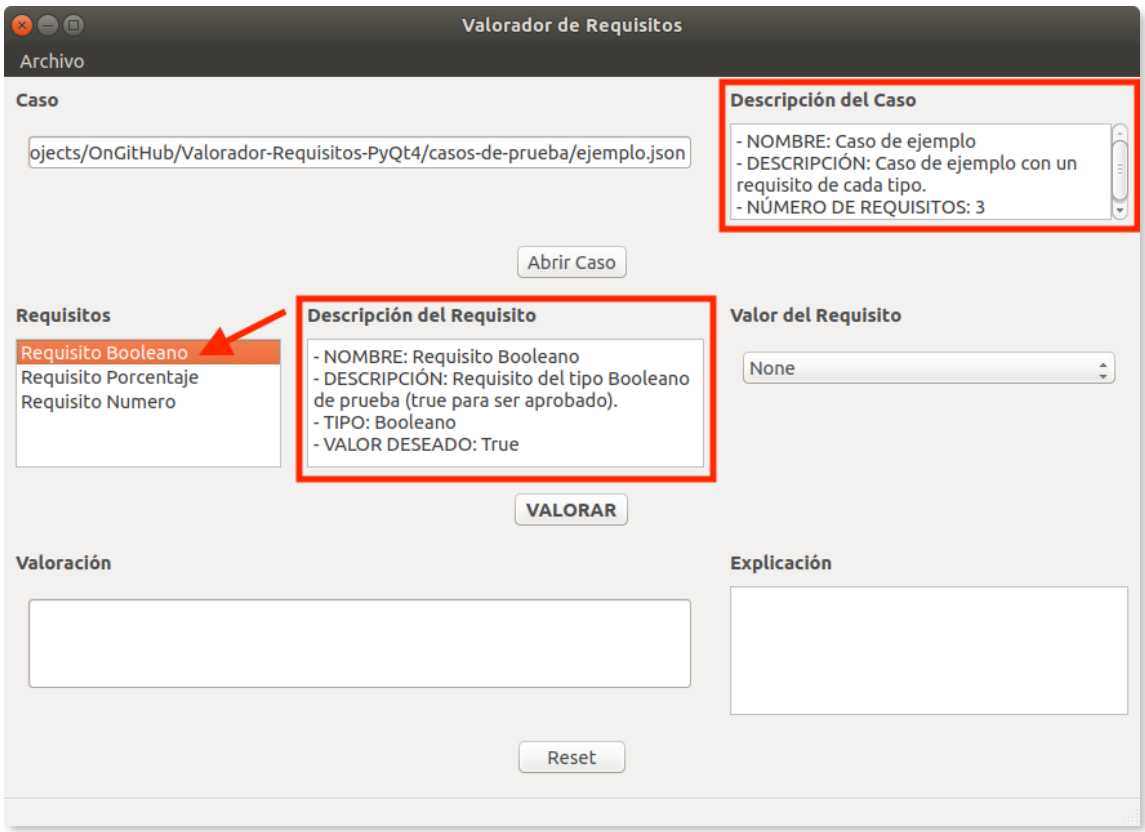


En caso de abrir un fichero con el formato incorrecto se nos mostrará un mensaje de error:

Inspeccionar Descripción del Caso y de sus Requisitos

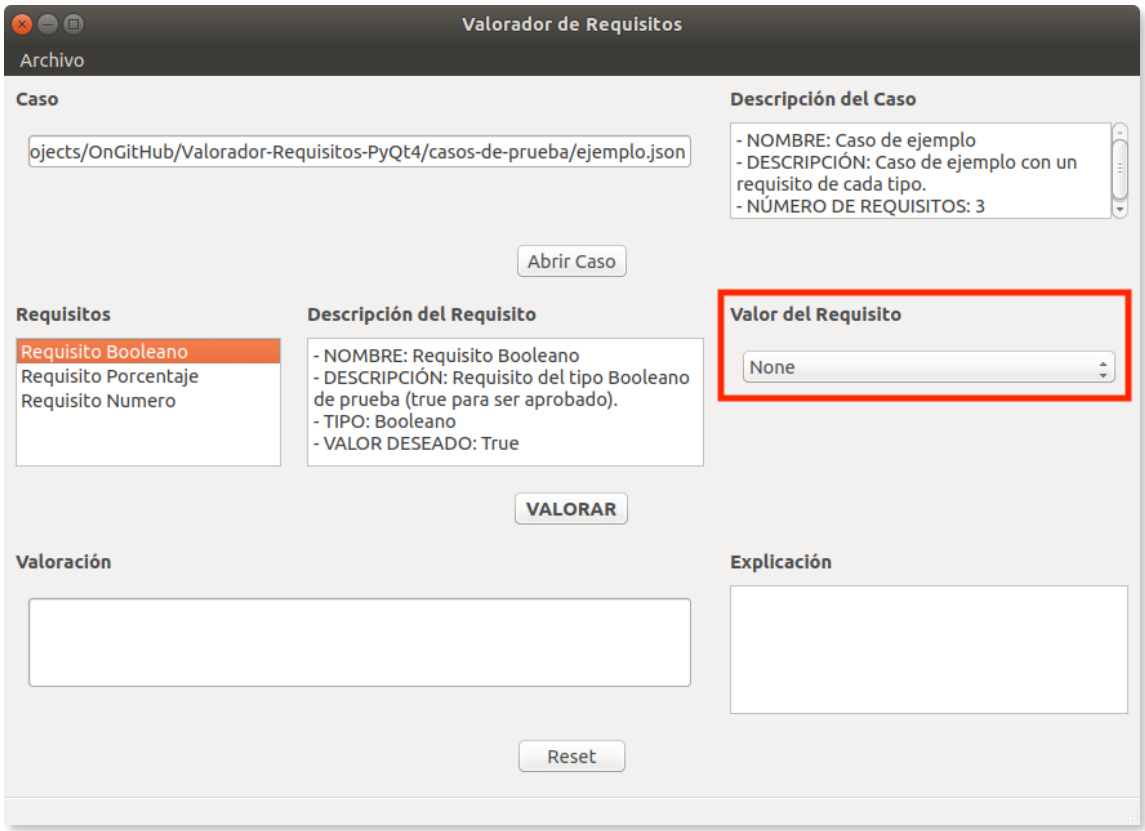
Una vez abierto el caso podremos ver la descripción de este y de todos sus requisitos. La descripción del caso se muestra automáticamente al abrirlo, a la derecha del campo que nos indica la ruta del fichero abierto.

Para ver la descripción de cada uno de los requisitos simplemente lo seleccionaremos en la lista de requisitos y automáticamente se cargarán todos sus datos.



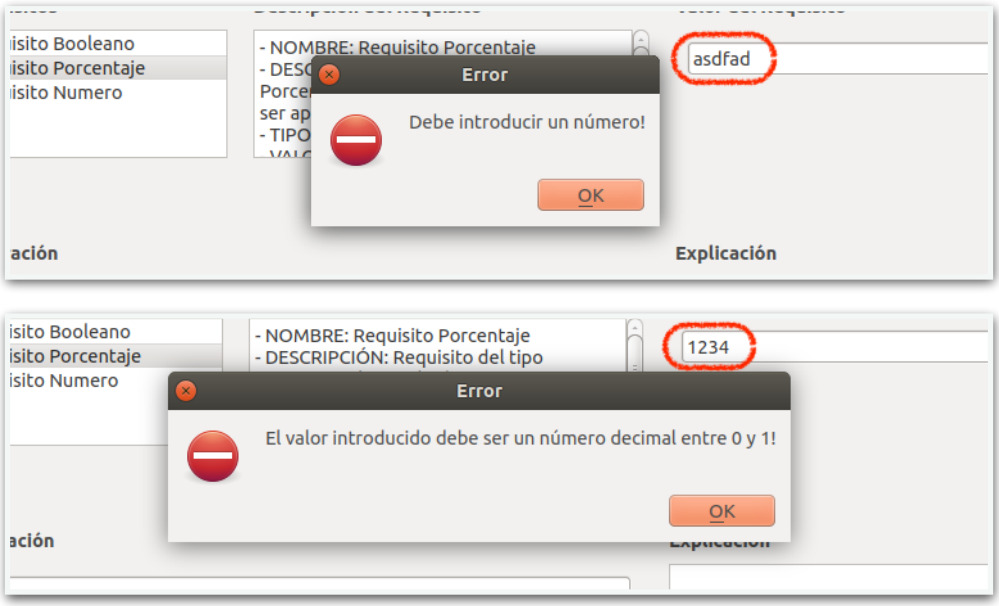
Introducir Valores para los Requisitos

Una vez seleccionado un requisito de la lista de requisitos podrá introducir su valor en el panel dedicado a ello:



Nótese que en caso de seleccionar un requisito del tipo "Booleano" en ese panel se mostrará un panel desplegable que le permitirá seleccionar su valor. En caso de tratarse de un requisito del tipo "Porcentaje" o "Numero" se mostrará un campo de texto en el que deberá introducir el valor que desee.

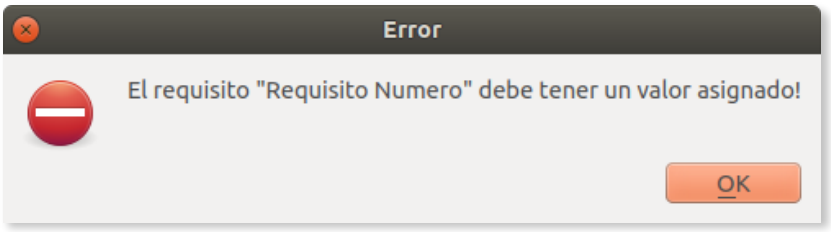
En caso de no introducir un número, o introducir un número que no esté entre 0 y 1 cuando se trate de un requisito del tipo "Porcentaje", se mostrará un mensaje de error que le obligará a introducir un valor correcto:



Ejecutar Valoración y Ver el Resultado

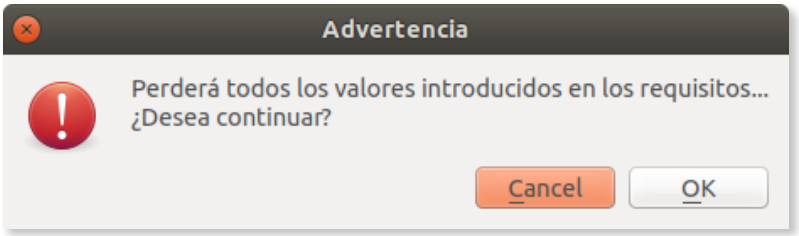
Una vez introducido el valor de todos los requisitos podremos proceder a valorar el caso. Para ello simplemente presione el botón "VALORAR" y el resultado de la valoración será visible inmediatamente. También podrá ver un informe del resultado en el que se mostrarán todos los requisitos con su descripción, el valor introducido y su valoración.

En caso de intentar ejecutar la valoración sin haber introducir el valor de alguno de los requisitos se mostrará un mensaje de error que le indicará el requisito cuyo valor debe introducir:



Reinicializar Valoración y Valores de los Requisitos

Si desea reinicializar el valor de todos los requisitos (y por consiguiente el resultado de la valoración) simplemente presione el botón "Reset". En ese momento se mostrará un ventana para que confirme la operación:



Formato del Fichero JSON

Los casos se almacenan y son cargados a partir de un fichero JSON (extensión .json). Esto permite que cualquier usuario pueda crear nuevos casos sin necesidad de que tenga conocimientos sobre Python. De hecho, el formato es tan amigable que incluso un usuario sin conocimientos de programación y que nunca haya visto un fichero de este tipo podría entender su contenido. Es por esto que el uso de este formato está muy extendido y se usa en aplicaciones de todo tipo. Además, prácticamente todos los lenguajes de programación modernos son compatibles con este formato de forma nativa e incluyen librerías que facilitan su uso.

A continuación se muestro el caso de ejemplo que hemos incluido con los archivos del proyecto. Cuenta con un requisito de cada tipo en cuya descripción se indica que valor deberá ser introducido para que sea valorado como "Aprobado".

Puede seguir este ejemplo para crear sus propios casos fácilmente.

.../casos-de-prueba/ejemplo.json

```
{
  "caso": {
    "nombre": "Caso de ejemplo",
    "descripcion": "Caso de ejemplo con un requisito de cada tipo.",
    "requisitos": [
      {
        "nombre": "Requisito Booleano",
        "descripcion": "Requisito del tipo Booleano de prueba (true para ser
  aprobado).",
        "tipo": "Booleano",
        "valor_deseado": true
      },
      {
        "nombre": "Requisito Porcentaje",
        "descripcion": "Requisito del tipo Porcentaje de prueba (entre 0.5 y 1 para ser
  aprobado).",
        "tipo": "Porcentaje",
        "valor_minimo": 0.5,
        "valor_maximo": 1
      },
      {
        "nombre": "Requisito Numero",
        "descripcion": "Requisito del tipo Numero de prueba (entre 0 y 1500 para ser
  aprobado).",
        "tipo": "Numero",
        "valor_minimo": 0,
        "valor_maximo": 1500
      }
    ]
  }
}
```

EJEMPLOS INCLUIDOS

Dentro de la carpeta "*casos-de-prueba*" del proyecto hemos incluido dos ejemplos de casos de la vida real demostrando que nuestra aplicación puede trabajar con diferentes dominios de conocimiento.

BECAS DE COLABORACIÓN DEL MECD PARA ESTUDIANTES DE GRADO - CONVOCATORIA 2019-2020

En el fichero "*.../casos-de-prueba/becas-colaboracion-grado-MECD.json*" nos encontramos con un caso que nos permite valorar si nuestra solicitud para una beca de colaboración podría ser aceptada. Se trata de becas del Ministerio de Educación, Cultura y Deporte destinadas a estudiantes universitarios para realizar tareas de investigación en departamentos universitarios.

Cuenta con tan solo 4 requisitos, pero nos permite probar los 3 tipos de requisitos con los que nuestra aplicación trabajo:

```
*** Requisito 1/4 ***
- NOMBRE: Posesión de un título de grado
- DESCRIPCIÓN: No estar en posesión o en disposición legal de obtener un título
  académico de Licenciado, Ingeniero, Arquitecto, Graduado o Máster oficial.
- TIPO: Booleano
- VALOR DESEADO: False


*** Requisito 2/4 ***
- NOMBRE: Matriculado de la totalidad de créditos pendientes
- DESCRIPCIÓN: Estar matriculado en el curso 2019-2020 en enseñanza oficial de la
  totalidad de las asignaturas o créditos que le resten para finalizar sus estudios.
- TIPO: Booleano
- VALOR DESEADO: True


*** Requisito 3/4 ***
- NOMBRE: Porcentaje de carga lectiva superada
- DESCRIPCIÓN: Los estudiantes de grado deberán encontrarse cursando los últimos
  créditos para completar los requisitos para la obtención del título, y haber
  superado el 75% de la carga lectiva.
- TIPO: Porcentaje
- VALOR MÍNIMO: 0.75
- VALOR MÁXIMO: 1.0


*** Requisito 4/4 ***
- NOMBRE: Nota media de expediente
- DESCRIPCIÓN: Tener como nota media de los créditos superados un mínimo de 7,25 puntos
  para la rama de Ingeniería y Arquitectura o Enseñanzas Técnicas.
- TIPO: Numero
- VALOR MÍNIMO: 7.25
- VALOR MÁXIMO: 10.0
```

Información obtenida de: http://www.educacionyfp.gob.es/gl/servicios-al-ciudadano/catalogo/estudiantes/becas-ayudas/para-estudiar/universidad/grado/becas-colaboracion.html

REQUISITOS PARA OPOSITAR A GUARDIA CIVIL

En el fichero "*.../casos-de-prueba/guardia-civil.json*" nos encontramos con un caso con los requisitos que los aspirantes a Guardia Civil deberán reunir, en la fecha que finalice el plazo de admisión de instancias y mantener durante el proceso selectivo y los periodos de formación.

Cuenta con nada menos que 12 requisitos:

```
*** Requisito 1/12 ***
- NOMBRE: Nacionalidad Española
- DESCRIPCIÓN: Poseer la nacionalidad española.
- TIPO: Booleano
- VALOR DESEADO: True
```

```
*** Requisito 2/12 ***
- NOMBRE: Derechos civiles
- DESCRIPCIÓN: No estar privado de los derechos civiles.
- TIPO: Booleano
- VALOR DESEADO: True

*** Requisito 3/12 ***
- NOMBRE: Antecedentes penales
- DESCRIPCIÓN: Carecer de antecedentes penales.
- TIPO: Booleano
- VALOR DESEADO: False

*** Requisito 4/12 ***
- NOMBRE: Incurso en algún procedimiento judicial
- DESCRIPCIÓN: No hallarse incurso en algún procedimiento judicial por delito doloso
  como procesado, investigado judicialmente o acusado con declaración de apertura de
  juicio oral correspondiente.
- TIPO: Booleano
- VALOR DESEADO: False

*** Requisito 5/12 ***
- NOMBRE: Habilitado para el servicio de la administración pública
- DESCRIPCIÓN: No haber sido separado mediante expediente disciplinario del servicio de
  cualquiera de las Administraciones Públicas ni hallarse inhabilitado con carácter
  firme para el ejercicio de funciones públicas.
- TIPO: Booleano
- VALOR DESEADO: True

*** Requisito 6/12 ***
- NOMBRE: Edad
- DESCRIPCIÓN: Tener cumplidos 18 años de edad en el año de la convocatoria y no tener
  cumplida ni cumplir durante el año de la convocatoria la edad de 41 años.
- TIPO: Numero
- VALOR MÍNIMO: 18.0
- VALOR MÁXIMO: 41.0

*** Requisito 7/12 ***
- NOMBRE: Altura (en metros)
- DESCRIPCIÓN: Se requiere una altura mínima de 1,65 metros en el caso de los hombres y
  de 1,60 en el caso de las mujeres. Del mismo modo tu altura no puede exceder de los
  2,03 metros.
- TIPO: Numero
- VALOR MÍNIMO: 1.6
- VALOR MÁXIMO: 2.03

*** Requisito 8/12 ***
- NOMBRE: Aptitud psicofísica
```

- DESCRIPCIÓN: Poseer la aptitud psicofísica necesaria para cursar los respectivos planes de estudios, que será acreditada mediante la superación del reconocimiento y las pruebas que se determinen en cada convocatoria.
- TIPO: Booleano
- VALOR DESEADO: True

*** Requisito 9/12 ***
- NOMBRE: Cumplir alguno de los requisitos de acceso a ciclos formativos de Formación Profesional de Grado Medio
- DESCRIPCIÓN: Cumplir alguno de los requisitos de acceso requeridos en el Sistema Educativo Español para acceder a las enseñanzas conducentes a ciclos formativos de Formación Profesional de Grado Medio.
- TIPO: Booleano
- VALOR DESEADO: True

*** Requisito 10/12 ***
- NOMBRE: Compromiso de portar armas y de su uso.
- DESCRIPCIÓN: Adquirir el compromiso de portar armas y, en su caso, llegar a utilizarlas, conforme a los principios básicos de actuación de congruencia, oportunidad y proporcionalidad en la utilización de los medios a su alcance, que se prestará a través de declaración del solicitante.
- TIPO: Booleano
- VALOR DESEADO: True

*** Requisito 11/12 ***
- NOMBRE: Poseer permiso de conducir del tipo B
- DESCRIPCIÓN: Estar en posesión del permiso de conducción de la clase B.
- TIPO: Booleano
- VALOR DESEADO: True

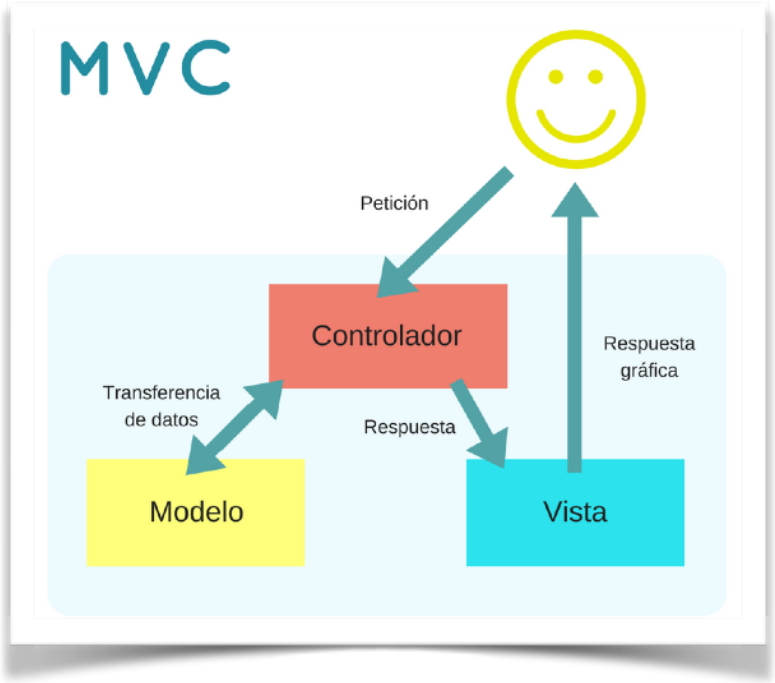
*** Requisito 12/12 ***
- NOMBRE: Poseer tatuajes conflictivos
- DESCRIPCIÓN: Carecer de tatuajes que contengan expresiones o imágenes contrarias a los valores constitucionales, Autoridades o virtudes militares, que supongan desdoro para el uniforme, que puedan atentar contra la disciplina o la imagen de la Guardia Civil en cualquiera de sus formas, que reflejen motivos obscenos o inciten a discriminaciones de tipo sexual, racial, étnico o religioso. Asimismo, tampoco se permiten los tatuajes, argollas, espigas e inserciones, automutilaciones o similares que puedan ser visibles vistiendo las diferentes modalidades de los uniformes de uso general del Cuerpo de la Guardia Civil.
- TIPO: Booleano
- VALOR DESEADO: False

Información obtenida de: http://www.interior.gob.es/web/servicios-al-ciudadano/oposiciones/cuerpo-de-la-guardia-civil/escala-de-cabos-y-guardias/requisitos

ANÁLISIS DEL CÓDIGO

ARCHITECTURA

La aplicación ha sido desarrollada siguiendo rigurosamente la arquitectura Modelo-Vista-Controlador.



Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

MODELO

El modelo se encarga de mantener el estado actual del programa.

En nuestro caso esto se consigue principalmente gracias a la clase *Caso* que contiene los datos del caso abierto y todos sus requisitos. Para gestionar los diferentes tipos de requisitos se usan las clases *RequisitoBooleano*, *RequisitoPorcentaje* y *RequisitoNumero* cada una de las cuales heredan de la clase base *Requisito.*

VISTA

Se trata de la interfaz gráfica de nuestro programa.

Está formado por una ventana principal (clase *ValoradorMainWindow*) que contiene la barra de menús, la barra de herramientas, la barra de estado y, por supuesto, el widget con el valorador

(clase *ValoradorWidget*). Este widget contiene los diferentes paneles, campos y botones de la aplicación.

También se incluye la clase auxiliar *ValoradorMessageBoxes* que contiene los métodos para mostrar mensajes emergentes y ventanas de diálogo para abrir ficheros.

CONTROLADOR

Se podría decir que la clase ValoradorController, el controlador, es la más importante de nuestro programa al ser la clase que controla el funcionamiento del programa y consigue que el resto de clases trabajen en sintonía. Se encarga de modificar el modelo a partir del input del usuario y, cuando este se ha modificado, actualizar la vista para reflejar estos cambios.

ESTRUCTURA DEL CÓDIGO

MÓDULO VALORADOR.PY

Módulo principal.

- Clase **Valorador**: Monta todas las piezas del MVC e inicia el programa.

MÓDULO VALORADOR_MODEL.PY

Módulo con el modelo del valorador.

- Clase ***`ValoradorModel`***: Clase con el modelo del valorador.
  - Atributo *`caso`*: El caso a valorar (objeto de la clase *`Caso`*).
  - Atributo *`opened_file_path`*: String con la ruta del fichero de caso abierto.
- Clase ***`Caso`***: Representa un caso (contiene los requisitos a valorar).
  - Propiedad *`nombre`*: String con el nombre del caso.
  - Propiedad *`descripcion`*: String con la descripción del caso.
  - Propiedad *`explicacion`*: String con la explicación del resultado de la valoración.
  - Propiedad *`requisitos`*: Los requisitos a evaluar (array con objetos de la clase *`Requisito`*).
  - Método *`__str__(self)`*: Devuelve la representación en string del objeto (para usar con *`print`*).
  - Método *`load_from_JSON_file(self, file_path)`*: Carga el caso y todos sus requisitos a partir de un fichero JSON con el formato adecuado.
  - Método *`_parse_JSON_file(self, file_path)`*: Lee y parsea un fichero JSON.
  - Método *`_full_reset(self)`*: Reinicializa el caso completamente (elimina todo, incluido los requisitos).
  - Método *`reset(self)`*: Reinicializa el caso (elimina la explicación y el valor de los requisitos).
  - Método *`valorar(self)`*: Evalúa todos los requisitos, devuelve el resultado de la valoración y actualiza el atributo explicacion con la explicación del resultado.

- Clase ***Requisito***: Clase base para representar los requisitos. Esta clase no debe ser instanciada; es solo una interfaz (clase base abstracta).
  - Propiedad *nombre*: String con el nombre del requisito.
  - Propiedad *descripcion*: String con la descripción del requisito.
  - Propiedad *tipo*: String con el tipo del requisito ("Booleano", "Porcentaje" o "Numero").
  - Propiedad *valor*: Valor actualmente asignado al requisito (su tipo dependerá del tipo de requisito). El requisito será evaluado en base a este valor.
  - Método *__str__(self)*: Devuelve la representación en string del objeto (para usar con *print*).
  - Método *valorar(self)*: Evalúa el requisito y devuelve True o False según corresponda.
  - Método *reset(self)*: Reinicializa el valor del requisito.
- Clase ***RequisitoBooleano***: Representa un requisito del tipo Booleano.
  - Propiedad *nombre*: String con el nombre del requisito.
  - Propiedad *descripcion*: String con la descripción del requisito.
  - Propiedad *tipo*: String con el tipo del requisito ("Booleano").
  - Propiedad *valor*: Valor actualmente asignado al requisito.
  - Propiedad *valor_deseado*: Valor que el requisito debe tener para ser evaluado como True.
  - Método *__str__(self)*: Devuelve la representación en string del objeto (para usar con *print*).
  - Método *valorar(self)*: Evalúa el requisito y devuelve True o False según corresponda.
- Clase ***RequisitoPorcentaje***: Representa un requisito del tipo Booleano.
  - Propiedad *nombre*: String con el nombre del requisito.
  - Propiedad *descripcion*: String con la descripción del requisito.
  - Propiedad *tipo*: String con el tipo del requisito ("Porcentaje").
  - Propiedad *valor*: Valor actualmente asignado al requisito.
  - Propiedad *valor_minimo*: Porcentaje mínimo necesario para evaluar el requisito como True.
  - Propiedad *valor_maximo*: Porcentaje máximo necesario para evaluar el requisito como True.
  - Método *__str__(self)*: Devuelve la representación en string del objeto (para usar con *print*).
  - Método *valorar(self)*: Evalúa el requisito y devuelve True o False según corresponda.
- Clase ***RequisitoNumero***: Representa un requisito del tipo Booleano.
  - Propiedad *nombre*: String con el nombre del requisito.
  - Propiedad *descripcion*: String con la descripción del requisito.
  - Propiedad *tipo*: String con el tipo del requisito ("Numero").
  - Propiedad *valor*: Valor actualmente asignado al requisito.
  - Propiedad *valor_minimo*: Valor mínimo necesario para evaluar el requisito como True.
  - Propiedad *valor_maximo*: Valor máximo necesario para evaluar el requisito como True.
  - Método *__str__(self)*: Devuelve la representación en string del objeto (para usar con *print*).

- Método *valorar(self)*: Evalúa el requisito y devuelve True o False según corresponda.

MÓDULO VALORADOR_VIEW.PY

Módulo con la vista del valorador.

- Clase ***ValoradorView***: Clase con la vista del valorador.
  - Atributo *main_widget*: *QWidget* con la interfaz del valorador (objeto de la clase ValoradorWidget).
  - Atributo *main_window*: Ventana principal (objeto de la clase ValoradorMainWindow).
  - Método *show(self)*: Hace visible la ventana principal.
- Clase ***ValoradorWidget***: *QWidget* con la interfaz del valorador.
  - Atributo *abrir_Button*: *QPushButton* para abrir fichero del caso.
  - Atributo *valorar_Button*: *QPushButton* para valorar el caso.
  - Atributo *reset_Button*: *QPushButton* para reinicializar el caso.
  - Atributo *ruta_caso_LineEdit*: *QLineEdit* de solo lectura que muestra la ruta del fichero del caso.
  - Atributo *valor_LineEdit*: *QLineEdit* para introducir el valor de un requisito del tipo Porcentaje o Numero.
  - Atributo *valor_ComboBox*: *QComboBox* para introducir el valor de un requisito del tipo Booleano.
  - Atributo *desc_caso_TextEdit*: *QTextEdit* de solo lectura que muestra la descripción del caso.
  - Atributo *desc_requisito_TextEdit*: *QTextEdit* de solo lectura que muestra la descricpión del requisito seleccionado.
  - Atributo *valoracion_LineEdit*: *QLineEdit* de solo lectura que muestra el resultado de la valoración.
  - Atributo *explicacion_TextEdit*: *QTextEdit* de solo lectura que muestra la explicación de la valoración.
  - Atributo *requisitos_List*: *QListWidget* con la lista de requisitos del caso.
  - Método *_init_UI(self)*: Inicialización de la interfaz.
- Clase ***ValoradorMainWindow***: *QMainWindow* con la ventana principal del programa.
  - Atributo *valorador_Widget*: Widget con el valorador (objeto de la clase *valoradorWidget*).
  - Atributo *exit_Action*: *QAction* para salir del programa.
  - Atributo *open_file_Action*: *QAction* para abrir fichero de caso.
  - Método *_init_UI(self)*: Inicialización de la interfaz.
- Clase ***ValoradorMessageBoxes***: Contiene métodos para mostrar mensajes emergentes y ventanas de diálogo para abrir ficheros.
  - Método *show_error_message(error_text)*: Muestra una ventana emergente de error con el mensaje indicado en el argumento *error_text*.
  - Método *show_info_message(info_text)*: Muestra una ventana emergente de información con el mensaje indicado en el argumento *info_text*.

- Método *confirm_operation_message(info_text)*: Muestra una ventana emergente de advertencia para confirmar que el usuario desea continuar con la operación.
- Método *open_file_dialog(parent, selectedFilter="")*: Muestra una ventana de diálogo para seleccionar el fichero a abrir.

MÓDULO VALORADOR_CONTROLLER.PY

Módulo con el controlador del valorador.

- Clase ***ValoradorController***: Clase con el controlador del valorador.
  - Método *_init_model(self)*: Inicializa el modelo.
  - Método *_init_view(self)*: Inicializa la vista.
  - Método *_init_controller(self)*: Inicializa el controlador. Conecta los botones y acciones de la vista con métodos del controlador.
  - Método *_load_caso(self)*: Muestra la ventana de diálogo para seleccionar el archivo a abrir y lo abre.
  - Método *_valorar_caso(self)*: Ejecuta la valoración de los requisitos del caso.
  - Método *_reset_caso(self)*: Reinicializa los valores de los requisitos y el resultado de la valoración.
  - Método *_set_valor_requisito(self)*: Actualiza el valor del requisito seleccionado con el valor introducido.
  - Método *_update_entire_UI(self)*: Actualiza todos los campos de la interfaz.
  - Método *_update_caso_fields(self)*: Actualiza los campos de la interfaz con la ruta y la descripión del caso.
  - Método *_update_requisitos_list(self)*: Carga los requisitos del caso en la lista de requisitos de la interfaz.
  - Método *_update_requisito_fields(self)*: Actualiza la vista con la descripción y el valor del requisito seleccionado.
  - Método *_update_valoracion_fields(self, valoracion_result)*: Actualiza los campos de la interfaz con el resultado y la explicación de la valoración del caso.
  - Método *_clean_valoracion_fields(self)*: Limpia los campos valoración y explicación de la vista.

Referencias

- **http://www.uco.es/~i52salia/issbc/**

Python2 y PyQt4

- https://www.tutorialspoint.com/python/python_quick_guide.htm
- https://google.github.io/styleguide/pyguide.html
- https://www.python.org/dev/peps/pep-0257/
- https://www.programiz.com/python-programming/exceptions
- http://srinikom.github.io/pyside-docs/index.html
- http://zetcode.com/gui/pyqt4/

Casos de Ejemplo

- http://www.interior.gob.es/web/servicios-al-ciudadano/oposiciones/cuerpo-de-la-guardia-civil/escala-de-cabos-y-guardias/requisitos
- http://www.educacionyfp.gob.es/gl/servicios-al-ciudadano/catalogo/estudiantes/becas-ayudas/para-estudiar/universidad/grado/becas-colaboracion.html