FACULTAD DE CIENCIAS

UNIVERSIDAD DE CANTABRIA Facultad de Ciencias

Proyecto Fin de Carrera

SISTEMA PARA GESTIONAR HORARIOS DE DIFERENTES CURSOS EN UN CENTRO DOCENTE

(System to Manage Different Timetables for a Teaching Center)

Para acceder al Título de

INGENIERO EN INFORMÁTICA

Autor: **José Ramón Vejo Gutiérrez**

Director: Domingo Gómez

Junio - 2017

RESUMEN

La gestión de los horarios de un centro docente supone un problema complejo debido a la multitud de recursos a gestionar, la cantidad de restricciones a satisfacer para cada uno de ellos y las diferencias entre la multitud de modelos existentes.

El proyecto abordará el problema conocido como "(High) School Timetabling" con el objetivo de diseñar un sistema que permita gestionar los diferentes horarios de los cursos que se imparten en un centro docente mediante una aplicación.

Esta aplicación permitirá visualizar y realizar modificaciones en los horarios de manera rápida, sencilla y visual.

Palabras clave: Aplicación, gestión de horarios, XHSTT, "(High) School Timetabling"

ABSTRAC

Scheduling of timetables for a teaching center is a complex problem because of the diversity of the resources to manage, the amount of constraints to satisfy to each of it and the differences between models.

In this project we will approach to the problem knows as “(High) School Timetabling” and wewill design a system to manage the timetables of the courses that impart in a teaching center by an application.

This application will allow to make modifications quickly, easily and in a visual way.

Keywords: Application, timetable scheduling, XHSTT, “(High) School Timetabling”

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La finalidad de este primer capítulo es proporcionar al lector una visión general del proyecto. Este capítulo incluye tres apartados. En primer lugar, se comienza exponiendo la motivación detrás del proyecto, así como los objetivos que se pretenden conseguir. Finalmente, en el último apartado se detalla la estructura que se seguirá este documento.

1.1. Motivación y contexto tecnológico

Como parte de la actividad de un centro docente, normalmente, el jefe de estudios tiene que establecer una serie de horarios para los diferentes cursos que se imparten en él.

Establecer estos horarios supone un problema debido a la cantidad de recursos que hay que gestionar (aulas, profesores, asignaturas, horas, etc.) y las diferentes incompatibilidades que se pueden presentar (espacios disponibles, material didáctico necesario, preferencias horarias, etc.).

La realización de este proyecto surge de la necesidad de la facultad de Ciencias de la Universidad de Cantabria de disponer de un software que plantee una solución satisfactoria al problema y que permita realizar cambios a dicha solución de forma rápida. El problema de la gestión de horarios se conoce como "(High) SchoolTimetabling" [1] (horarios de escuela secundaria). Existen otros problemas similares, como “University Course Timetabling".

Para solucionar este problema se hará uso de herramientas de software libre y se expondrán algunas soluciones existentes para problemas similares, adaptándolas a las necesidades del usuario. También, se desarrollará una aplicación que en la que se mostrará la primera solución calculada y permita realizar modificaciones sobre ella. Además mostrará los posibles conflictos que puedan presentarse en dicha solución.

1.1.1. Problema de los horarios de escuela secundaria

Este problema es el más sencillo en lo que a la asignación de horarios se refiere. En general, este tipo de problemas consisten en la existencia de varios grupos de recursos (profesores, asignaturas, aulas y cursos) y unos periodos de tiempos a los que asignar la asignatura que imparte un profesor y el lugar en el que lo hace.

Además, se presentan una serie de restricciones [2] a satisfacer. Estas restricciones pueden ser de dos tipos, "hard" (restricciones que hay que cumplir) y "soft" (restricciones que es aconsejable cumplir.

Las primeras hacen referencia a las condiciones que tiene que cumplir el horario para considerarlo como una solución válida. Este grupo de restricciones indican, por ejemplo, que todas las asignaturas tienen que tener un profesor asociado, un aula y un tiempo, que un aula tiene un uso específico, o tan obvias como que un profesor no puede impartir dos clases a la misma hora.

Las segundas indican lo buena que es la solución y hacen referencia a indicaciones que sería aconsejable cumplir, tales como que un profesor quiera que sus clases sean impartidas de forma continua o en un grupo de horas determinado. Suelen depender de la política del centro docente o de indicaciones personales.

1.2. Objetivos

El principal objetivo de este proyecto consiste en establecer un horario válido para todos los cursos impartidos en un centro docente, satisfaciendo el mayor número de restricciones "soft" posibles y permitiendo realizar modificaciones posteriormente de manera manual, sencilla y de forma gráfica y visual.

Un horario válido es aquel que satisface todas las restricciones "hard" y su solución será de mejor calidad cuantas más restricciones de tipo "soft" cumpla.

El entorno gráfico tiene que permitir consultar los diferentes horarios, realizar modificaciones y almacenar dichos cambios.

1.3. Estructura del documento

El presente documento se encuentra dividido en seis capítulos más, recogiendo toda la información relacionada con el desarrollo del proyecto. La estructura es la siguiente:

- Capítulo 2: Análisis de requisitos. Se recogen y analizan los requisitos funcionales y no funcionales de la aplicación.
- Capítulo 3: Material y métodos. Se presenta la metodología, herramientas y tecnologías que permitieron llevar a cabo este proyecto.
- Capítulo 4: Casos de uso. Se detallan todos los casos de uso junto con los actores que interactúan con el sistema.
- Capítulo 5: XHSTT. Se explica KHE y el formato utilizado de XHSTT y por qué se ha elegido como solución al problema.
- Capítulo 6: Diseño e implementación. Explicación de los procesos de diseño e implementación del sistema.
- Capítulo 7: Evaluación y pruebas. Se muestra el contenido y el resultado de las pruebas realizadas para comprobar la calidad y efectividad de la aplicación.
- Capítulo 8: Conclusiones y trabajos futuros. Se exponen las conclusiones obtenidas y se describen las posibles mejoras en la aplicación.

2. ANÁLISIS DE CONTENIDOS

En este capítulo se presenta la fase de análisis, parte inicial de todo proyecto software. Se muestra una especificación de requisitos detallada, tanto los requisitos funcionales como los no funcionales.

2.1. Requisitos funcionales

En este apartado se estudia el problema y se acuerdan los requisitos que se deben satisfacer. Los requisitos funcionales definen una función del software que el sistema debe cumplir, estableciendo su comportamiento.

A continuación se detallan los requisitos funcionales [3] del sistema:

Identificador	Descripción
RF00	El sistema estará diseñado para ser utilizado por un usuario y su mantenimiento será realizado por un experto conocedor del formato XHSTT.
RF01	El sistema dispondrá de una interfaz gráfica que permita visualizar y realizar los cambios en los horarios.
RF02	El sistema tiene que permitir filtrar los horarios según diversos criterios (cursos, profesores, asignaturas y aulas). Una vez seleccionado un filtro, el resto deberán reducirse, condicionados por el primero.
RF03	El sistema permitirá modificar el día y la hora en la que se imparte una asignatura.
RF04	El sistema indicará los cambios que se han producido en el horario desde que este se cargó.
RF05	El sistema permitirá modificar el aula asignada por defecto a una asignatura.
RF06	El sistema mostrará las restricciones que no se cumplan, indicando su tipo.

Tabla 1. Requisitos funcionales de la aplicación.

2.2. Requisitos no funcionales

Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema software.

Existen diferentes categorías de requisitos no funcionales, los que están relacionados con el sistema a diseñar con los siguientes: usabilidad, mantenibilidad, tecnológicos, accesibilidad y de interfaz.

2.2.1. Usabilidad

La aplicación debe interaccionar con el usuario a través de una interfaz gráfica bien formada. Deberá presentar la información de la manera más clara y simple posible, de forma que el usuario podrá usar el sistema sin complicaciones y sin ningún tipo de formación previa.

2.2.2.Mantenibilidad

El sistema utilizará como formato para manejar los datos un archivo XHSTT en que se incluirán todos los datos para su funcionamiento. Esto permitirá calcular el informe sobre la solución utilizando el evaluador online HSEval [4].

2.2.3.Tecnológicos

La aplicación deberá ser programada en lenguaje Python[5] y Kivy[6], por lo que será necesaria su instalación para ser ejecutada.

Además, los datos deberán ajustarse al formato del fichero XHSTT desarrollado.

2.2.4.Accesibilidad

Debe ser compatible con Windows 8.2 y superiores.

2.2.5.Interfaz

La aplicación presentará diversas pantallas. En la principal se realizarán los filtrados y la carga.

Los horarios se mostrarán en dos pantallas, una para la jornada de mañana y otra para la tarde. En otra pantalla se mostrarán las restricciones que no se cumplan en formato texto.

3. MATERIAL Y MÉTODOS

En este capítulo se expondrán estas herramientas y tecnologías utilizadas en el desarrollo del proyecto, así como la metodología utilizada.

Tanto el lenguaje en el que será programada la aplicación como el empleado para el apartado visual y el de datos vienen impuestos por la petición, por lo que no será necesario realizar una búsqueda de alternativas.

3.1. Herramientas y Tecnologías

El desarrollo del proyecto se ha llevado a cabo utilizando los siguientes lenguajes, tecnologías y entornos de desarrollo.

3.1.1.Python

Python es un lenguaje de programación interpretado multiparadigma y multiplataforma. Su filosofía de diseño hace hincapié en que su sintaxis favorezca la creación de código legible.

Su administración corre por cuenta de Python Software Fundation[7]. Es código abierto.

3.1.2.Kivy

Kivy es un Framework de código abierto para Python para el desarrollo rápido de aplicaciones, que hace uso de interfaces de usuario novedosas.

Es multiplataforma, siendo posible su ejecución en Linux, Windows, Android y iOS.

3.1.3.XHSTT

XHSTTes el formato de archivo utilizado para almacenar los datos y modelar el problema. Con él se generan las soluciones e informes necesarios para que la aplicación funcione. En el capítulo XHSTT se explicará con detalle.

3.1.4.IDLE

IDLE [9]es un entorno de desarrollo integrado para programación en lenguaje Python. Este ha sido el entorno utilizado durante todo el proyecto para la programación de la aplicación.

3.1.5. KHE

KHE [16] es una librería de software de código abierto utilizada para generar soluciones a los problemas modelados en un archivo XHSTT bien formado.

3.1.6.HSEval

Es el evaluador de soluciones para archivos XHSTT que utiliza la aplicación. Este evaluador online calcula un informe sobre la solución a partir del fichero con las restricciones no cumplidas. Además, se pueden generar tablas de horarios para los diferentes recursos. Está basado en KHE.

3.2. Metodología

Una vez definidos los objetivos, se establece el procedimiento para alcanzarlos de manera satisfactoria y obtener un software de calidad.

Existen gran cantidad de metodologías. En este caso, se ha optado por un modelo iterativo e incremental [8].

Según dicho modelo, se suceden iteraciones de varios ciclos de vida en cascada, de manera que se parte de una primera versión a la que se le va añadiendo funcionalidad a la aplicación en cada ciclo y mejorando su calidad.

Esto permite generar software operativo rápidamente, permitiendo detectar errores de manera temprana y facilitando la gestión de riesgos.

En el siguiente diagrama se representan las etapas de las que consta el modelo:

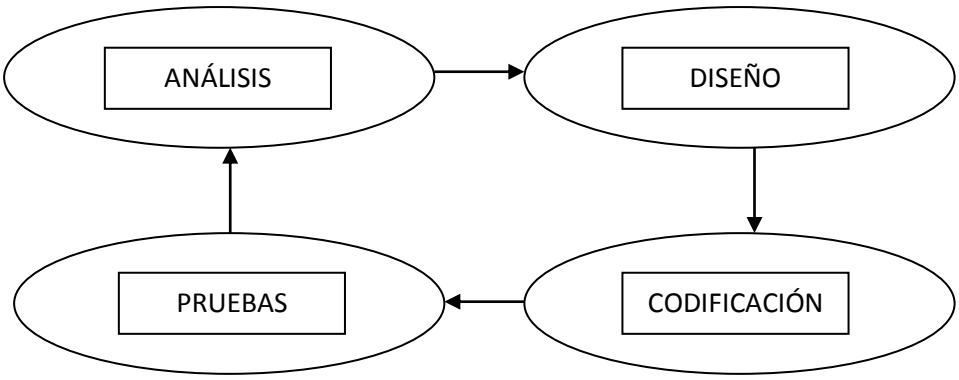

Figura 1. Esquema del ciclo de vida iterativo incremental.

En la primera iteración se persigue conseguir una primera versión funcional de la aplicación. En ella se establecen las pantallas básica que se necesitarán (principal y los horarios de mañana y tarde), así como los botones para navegar entre ellas y una primera aproximación a como se visualizarán los datos (se realiza con datos de prueba). Se implementan los requisitos funcionales [RF00] y [RF01]

En las siguientes se va añadiendo funcionalidad. Cada iteración está centrada en un objetivo concreto:

- Segunda: Permitir el intercambio de asignaturas. [RF03] [RF04]
- Tercera: Añadir el filtrado. [RF02]
- Cuarta: Permitir el cambio del aula a una asignatura. [RF05]
- Quinta: Pantalla con las restricciones incumplidas. [RF06]

Así, se consigue implementar una primera versión funcional a la que se le van añadiendo más características.

4. CASOS DE USO

En este capítulo se identifica a los actores que intervendrán con el sistema tomando como base los requisitos funcionales expuestos en el capítulo dos "ANÁLISIS DE REQUISITOS" y se detallarán los diferentes casos de uso.

4.1. Identificación de los actores

Se identifican los siguientes actores, según el rol que tienen:

- **Usuario**: Acceden a la aplicación y pueden realizar cualquier tipo de acción en ella. Como la aplicación está destinada a gestionar los horarios de los cursos, los usuarios de la misma serán los encargados de realizar los horarios en el centro docente (por norma, el jefe de estudios.

- **Administrador**: Administrador del sistema. Será el encargado de actualizar los datos de la aplicación cuando sea necesario, además de realizar su instalación.

4.2. Diagrama de casos de uso

En un diagrama de casos de uso se indica la relación que mantienen los diferentes actores con los casos de uso del sistema. Esto describe las interacciones entre el usuario y el sistema.

Un caso de uso [9] es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios.

Este es el diagrama de casos de uso establecido para la aplicación desarrollada:

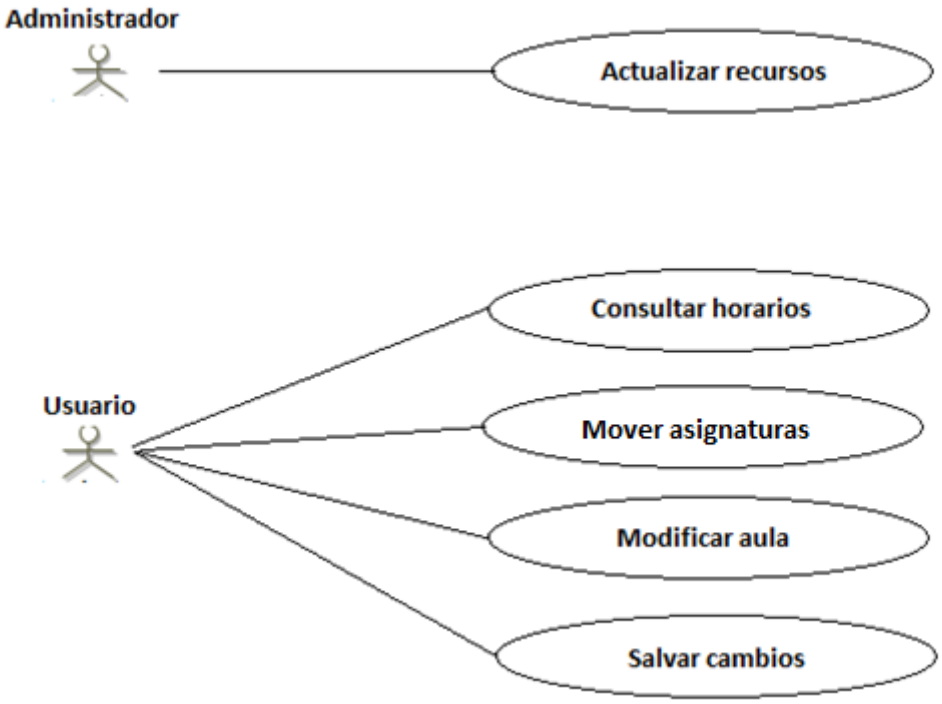

Figura2. Diagrama de casos de uso.

4.2.1.Casos de uso

Este es el detalle de los casos de uso indicados anteriormente:

Nombre	Actualizar recursos
Actor principal	Administrador
Actor secundario	
Descripción	El administrador es el encargado de actualizar el fichero desde el que la aplicación leerá los datos.
Evento de activación	El administrador actualiza cualquier dato del fichero.
Precondición	El fichero tiene que estar bien formado y seguir la estructura XHSTT desarrollada.
Garantías de éxito	Los datos del sistema son actualizados.
Escenario principal	1. El administrador realiza modificaciones en el fichero XHSTT. 2. El administrador inicializa la aplicación. 3. El sistema lee el fichero componiendo el apartado visual de la aplicación. 4. El administrador selecciona los datos a cargar. 5. El sistema lee los datos solicitados del fichero. 6. El administrador comprueba en las diferentes pantallas que los datos han sido actualizados.
Extensiones	2a. El formato del fichero no es correcto 1. Se debe comprobar que el fichero está bien formado antes de actualizarlo. Si no, el sistema no se inicializará.

Tabla 2. Caso de uso Actualizar recursos

Nombre	Cargar datos de horarios
Actor principal	Usuario
Actor secundario	
Descripción	El usuario desea consultar los datos de algún horario en la aplicación.
Evento de activación	El usuario pulsa el botón de carga que mostrará el horario según el filtro seleccionado.
Precondición	Aplicación inicializada correctamente y un filtro seleccionado.
Garantías de éxito	El sistema muestra los datos correspondientes al filtro seleccionado.
Escenario principal	1a. El usuario pulsa el botón de carga. 2a. El sistema lee el fichero insertando en el horario los datos correspondientes al filtro seleccionado. 3a. El usuario consulta en las diferentes pantallas los datos.
Extensiones	1a. El formato del fichero no es correcto, el sistema se cerrará. 3a. El usuario podrá modificar el filtro pulsando el botón de reinicio del mismo, seleccionando otro nuevo y volviendo a cargar.

Tabla 3. Caso de uso Cargar datos de horarios

Nombre	Mover asignaturas
Actor principal	Usuario
Actor secundario	
Descripción	El usuario desea mover una asignatura de día/hora.
Evento de activación	El usuario selecciona las asignaturas a mover.
Precondición	Datos de un horario cargados.
Garantías de éxito	El sistema desmarca las modificaciones realizadas y las guarda en el fichero.
Escenario principal	1a. El usuario selecciona la primera asignatura a mover. 2a. El sistema la marca. 3a. El usuario selecciona la segunda asignatura a mover 4a. El sistema intercambia las asignaturas de lugar y marca la segunda. 5a. El usuario pulsa el botón para guardar las modificaciones. 6a. El sistema guarda las modificaciones en el fichero y desmarca los cambios.
Extensiones	3a. El usuario deja el intercambio a medias intentando asignar un aula. 1. El sistema intercambiará la primera asignatura seleccionada con la siguiente. 2. Tras el intercambio, permitirá la asignación de un aula.

Tabla 4. Caso de uso Mover asignaturas

Nombre	Modificar aula
Actor principal	Usuario
Actor secundario	
Descripción	El usuario desea modificar el aula de algún horario en la aplicación.
Evento de activación	El usuario selecciona un aula y pulsa sobre el día/hora a la que asignarla.
Precondición	Datos de un horario cargados.
Garantías de éxito	El sistema desmarca las modificaciones realizadas y las guarda en el fichero.
Escenario principal	1a. El usuario selecciona el aula deseada en el desplegable. 2a. El usuario selecciona el día/hora al que asignar. 3a. El sistema asigna el aula y marca como modificado el día/hora. 4a. El usuario pulsa el botón para guardar las modificaciones. 5a. El sistema guarda las modificaciones en el fichero y desmarca los cambios.
Extensiones	1a. El usuario puede seleccionar otra aula en cualquier momento. 1. Si el usuario quiere volver a intercambiar asignaturas, selecciona la opción correspondiente. 2a. El usuario se ha equivocado en la asignación. 1. Puede volver a cargar el horario sin guardar. 2. Puede seleccionar el aula anterior y asignarla.

Tabla 5. Caso de uso Modificar aula

Nombre	Consultar incidencias
Actor principal	Usuario
Actor secundario	
Descripción	El usuario desea consultar las incidencias de algún horario en la aplicación.
Evento de activación	El usuario inicia la aplicación
Precondición	Aplicación inicializada correctamente.
Garantías de éxito	El sistema muestra las incidencias encontradas.
Escenario principal	1a. El usuario inicializa la aplicación. 2a. El sistema lee el fichero componiendo el apartado visual de la aplicación. Inicializa el horario vacío. 3a. El usuario selecciona la ventana para visualizar las incidencias. 4a. El sistema muestra en formato texto las incidencias.
Extensiones	1a. El formato del fichero no es correcto, el sistema no se inicializa.

Tabla 6. Caso de uso Consultar incidencias

5. XHSTT

Para resolver el problema de la gestión de horarios y su evaluación comparativa en su modelo más básico, denominado "(High) School Timetabling", un grupo de investigadores de diversas universidades propuso el formato de archivo XHSTT [12] y un evaluador capaz de comprobar la sintaxis de las instancias y evaluar las soluciones.

5.1. Discusión del problema

Debido a que no existía ningún formato estándar para el intercambio de set de datos respecto al problema de la gestión de horarios "(High) School Timetabling", diseñaron el formato XHSTT basado en XML y que sigue una estructura específica.

En la creación de este formato tuvieron en cuenta que este problema no es igual para todas las partes del mundo. Por ejemplo, en el nivel de la solución. En unos lugares se requiere un proceso de planificación a nivel de alumno, en otros a nivel de clase, siendo en el primer caso un nivel más exhaustivo y necesitando una evaluación para cada individuo, volviendo el problema más complejo computacionalmente.

Carter, M. W. [14] había formulado ya el problema y propuesto una evaluación comparativa que se había convertido en una especie de estándar, pero era limitada y completaba su significado con texto plano. Otros autores extendieron su formulación para adaptarla a los casos reales. Hasta ahora, la formulación más compleja disponible es la realizada por McCollum [15], usada en el segundo campeonato de asignación de horarios (ITC 2007) [16]. De este concurso surgieron dos de las propuestas más investigadas y desarrolladas, PE-CTT [17] y CB-CTT [18]. Sin embargo, aún no existe una formulación general ni el formato de su correspondiente archivo XML.

5.2. Estructura general

La principal finalidad del formato XHSTT es contener sets de datos para el problema de los horarios de escuela secundaria ("High School Timetabling"). Además de almacenar los datos, este tipo de archivo permite contener una solución (o varias) junto a las restricciones que se violan y el coste de la solución.

Para ello, el archivo se divide en dos grandes partes. Por un lado están todos los datos que se quieren almacenar y que modelan el problema, englobados en la etiqueta <instance>. Es aquí donde está la información necesaria para la resolución del problema y las reglas que se aplicarán.

```
<Instance Id="Example">
  <Times>
    <TimeGroups>
      <Day Id="Day1"/> <Name>Monday</Name> </Day>
      ...
      <Day Id="Day5"/> <Name>Friday</Name> </Day>
      <TimeGroup Id="AllTimes"/> <Name>AllTimes</Name> </TimeGroup>
      ...
    </TimeGroups>
    <Time Id="Day1_1">  <Name>Monday 1</Name>
      <TimeGroups>
        <Day Reference="Day1"/>
        <TimeGroup Reference="AllTimes"/>
      </TimeGroups>
    </Time>
    ...
  </Times>
  <Resources>
     ...
  </Resources>
  <Events>
     ...
  </Events>
  <Constraints>
     ...
  </Constraints>
</Instance>
```

Figura 3. Instancia de un problema en formato XHSTT

Por otro, tenemos las soluciones y los informes sobre las restricciones incumplidas, almacenada cada una en la etiqueta <SolutionGroup>.

```
<HighSchoolTimetableArchive>
   <Instances>
      <Instance Id="Instance1">
         ...
      </Instance>
      <Instance Id="Instance2">
         ...
      </Instance>
      ...
   </Instances>
   <SolutionGroups>
      <SolutionGroup>
         <Solution Reference="Instance1">
            ...
         </Solution>
         <Solution Reference="Instance1">
            ...
         </Solution>
         ...
      </SolutionGroup>
   </SolutionGroups>
</HighSchoolTimetableArchive>
```

Figura 4. Instancias y grupo de soluciones

5.3. Estructura de las instancias

Cada instancia ("<instance>") contiene cuatro grupos de ítems diferentes:

- Relacionados con el tiempo
- Relacionados con los recursos
- Relacionados con eventos
- Relacionados con las restricciones

Los tres primeros grupos contienen poca información. La mayor parte de la lógica de negocio reside en el último grupo.

5.3.1.Tiempo

Consta de las entidades: TimeGroups, Weeks, Days y Time.

La etiqueta Time es la unidad básica de tiempo. La asignación de sus propiedades se realiza mediante la vinculación a TimeGroups.

Los ítems denominados por la etiqueta TimeGroups son un conjunto de ítems Time que tienen las mismas propiedades, como por ejemplo, un intervalo de horas concreto. Day y Week son etiquetas un ítem TimeGroup especial. Como todos, pueden ser añadidos como propiedad a la etiqueta Time.

Las etiquetas de tiempo (Time y TimeGroup) se utilizan para modelar todo lo referente a los tiempos en los que tienen que asignarse los recursos. Indican las horas que componen un día o si existen grupos especiales de tiempos, que serán utilizados por las restricciones para calcular la solución.

Por ejemplo, si un profesor quiere que sus clases, habría que crear un grupo de tiempo para indicar si el Time pertenece o no a este, mediante una referencia.

```
<Time Id="Mon4">
    <Name>Mon4</Name>
    <Day Reference="Monday"/>
    <TimeGroups>
        <TimeGroup Reference="BeforeLunch"/>
    </TimeGroups>
</Time>
```

Figura 5. Especificación XHSTT de los tiempos

5.3.2. Recursos

Contiene las entidades: ResourceTypes, ResourceGroups y Resources.

Los ítems recogidos por la etiqueta Resources representan los recursos disponibles. Tan sólo pueden hacer referencia a un ítem ResourceType. Un ítem ResourceTypes indica que todos los recursos que le hagan referencia tienen esa propiedad (profesor, clase, laboratorio, etc).

Los ítem bajo la etiqueta ResourceGroups son un conjunto de recursos con el mismo ResourceTypes.

```
<ResourceTypes>
    <ResourceType Id="Teacher">
        <Name>Teacher</Name>
    </ResourceType>
    <ResourceType Id="Room">
        <Name>Room</Name>
    </ResourceType>
    <ResourceType Id="Class">
        <Name>Class</Name>
    </ResourceType>
</ResourceTypes>
```

Figura 6. Especificación XHSTT de los recursos

5.3.3. Eventos

Contienen tres tipos de entidades: EventGroups, Courses y Events.

Un ítem Event es una reunión de recursos en un determinado y con una duración determinada. Por lo tanto, la resolución del problema de horarios consiste en establecer un tiempo de comienzo a los eventos asignándoles recursos. Estos recursos quedarían ocupados para ese tiempo.

Los ítem EventGroups son un conjunto de ítems Event. Los ítem Courses son un conjunto de ítem EventGroups especial, que sirven para indicar una propiedad de un ítem Event.

Un ítem Event hace referencia, además de a su duración, a una serie de recursos que necesita para ser satisfecho, bien de manera directa o mediante un tipo determinado (rol).

```
Event Id +Color
    Name
    Duration
    +Workload
    +Course
    +Time
    +Resources
    +ResourceGroups
    +EventGroups
```

Figura 7. Especificación XHSTT de un evento

5.3.4. Restricciones

Los ítem de tipo Constraint (Restricciones) pueden ser de dos tipos:

- **Hard Constraints**: Este tipo de restricciones deben ser cumplidas obligatoriamente por la solución para que esta se considere correcta.
- **Soft Constraints**: Este tipo de restricciones no es de obligatorio cumplimiento, pero se considera que la solución es mejor cuantas más restricciones de este tipo cumple.

Existen multitud de restricciones que se pueden implementar:

```
Constraints
    *AssignResourceConstraint
    *AssignTimeConstraint
    *SplitEventsConstraint
    *DistributeSplitEventsConstraint
    *PreferResourcesConstraint
    *PreferTimesConstraint
    *AvoidSplitAssignmentsConstraint
    *SpreadEventsConstraint
    *LinkEventsConstraint
    *OrderEventsConstraint
    *AvoidClashesConstraint
    *AvoidUnavailableTimesConstraint
    *LimitIdleTimesConstraint
    *ClusterBusyTimesConstraint
    *LimitBusyTimesConstraint
    *LimitWorkloadConstraint
```

Figura 8. Constraints de XHSTT

Si un horario cumple con todas las restricciones obligatorias, se dice que el horario es realizable (tiene solución). Este es el objetivo principal en la búsqueda de una solución. Cuando se comparan diferentes soluciones válidas, se determina que la mejor es la que mayor cantidad de restricciones no obligatorias cumple.

Para indicar que una restricción pertenece a uno u otro de los anteriores tipos, se indica mediante la etiqueta <Required>, siendo su valor "true" cuando es obligatoria y "false" en caso contrario.

Además, cada restricción lleva asociada un peso y una función. El peso indica cómo afecta a la solución saltarse la restricción una vez, y la función cómo crece este costo si se repite la violación de la misma.

```
AnyConstraint Id
    Name
    Required
    Weight
    CostFunction
    AppliesTo
    ...
```

Figura 9. Especificación Constraints

El resto de etiquetas definen cómo se aplica la restricción (modelado del problema).

5.4. Estructura de las soluciones

La estructura de la solución es más sencilla. En ella se indica, para cada ítem Event de la estructura <Instance> los recursos necesarios para satisfacerlo, siguiendo la lógica descrita mediante las restricciones establecidas previamente.

La asignación consiste en establecer una referencia temporal y un recurso con rol para el evento. El Evento hace referencia a ambos mediante etiquetas.

```
SolutionGroups
    SolutionGroup Id
        MetaData
        *Solution
```

Figura 10. Especificación de SolutionGroups

5.5. Adaptación de XHSTT

Durante el desarrollo del proyecto se ha generado un fichero con el formato XHSTT que modela el problema.

Primero ha sido necesario recopilar los datos referentes a toda la información que iba a ser necesaria para el modelado del problema en el archivo XHSTT. Esta labor se ha realizado a través de diferentes hojas de cálculo (Excel) proporcionadas por la Facultad de Ciencias con todos los datos necesarios, profesores, asignaturas, aulas y capacidad de las mismas, etc.

Posteriormente, se ha creado el fichero XHSTT de forma manual, introduciendo en él los datos anteriormente recopilados en el lugar adecuado de la primera parte del archivo, en la etiqueta <intances>, y modelando el problema.

5.5.1. Tiempos

En relación a los tiempos, se han creado los grupos de días de la semana que se mostrarán en el horario (de lunes a viernes), los intervalos de horas (cualquiera, tarde, antes o después del descanso) y las unidades de tiempo básicas (una hora del día), cada una de estas haciendo referencia a las correspondientes de las anteriores.

Por ejemplo, la tercera hora del martes hace referencia al día correspondiente (martes) y a los grupos a los que pertenece:

```
▼<Time Id="Martes3">
   <Name>Martes - 3</Name>
   <Day Reference="Martes"/>
  ▼<TimeGroups>
     <TimeGroup Reference="TodasHoras"/>
     <TimeGroup Reference="DespuesDescanso"/>
   </TimeGroups>
 </Time>
```

Figura 11. Ejemplo de ítem Time

5.5.2. Recursos

Para los recursos se han definido sus tipos según los diferentes roles que desempeñarán. Después, se han incluido los diferentes grupos, los correspondientes a los tipos y los nuevos, cada uno modelando uno de los grupos de cada curso .

```
<ResourceGroup Id="gr_Class">
  <Name>Cursos</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
<ResourceGroup Id="Class1_DOBLE_1">
  <Name>Clase 1 (grupo 1) de DOBLE</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
```

Figura 12. Ejemplo de ítem ResourceGroup

Por último, se han añadido cada uno de los recursos en sí, profesores, aulas y cursos.

```
<Resource Id="JOSE_MIGUEL_PRELLEZO_GUTIERREZ">
  <Name>JOSE MIGUEL PRELLEZO GUTIERREZ</Name>
  <ResourceType Reference="Teacher"/>
  <ResourceGroups>
    <ResourceGroup Reference="gr_Teacher"/>
  </ResourceGroups>
</Resource>
<Resource Id="AULA_1">
  <Name>AULA 1</Name>
  <ResourceType Reference="Room"/>
  <ResourceGroups>
    <ResourceGroup Reference="gr_Room"/>
    <ResourceGroup Reference="gr_Aula_Grande"/>
  </ResourceGroups>
</Resource>
```

Figura 13. Recurso profesor y aula

Todos incluyen en la información su nombre y las referencias a su tipo y posibles grupos.

5.5.3. Eventos

Los ítem Event más básicos son las partes más pequeñas de cada asignatura, por ejemplo, la clase de teoría de una asignatura.

```
<Event Id="PROGRAMACION_t">
  <Name>PROGRAMACION: Teoria</Name>
  <Duration>3</Duration>
  <Resources>
    <Resource Reference="Class1_MATEMATICAS_1"/>
    <Resource Reference="Class1_DOBLE_1"/>
    <Resource Reference="Class1_FISICA_1"/>
    <Resource Reference="JOSE_JAVIER_GUTIERREZ_GARCIA"/>
    <Resource Reference="MICHAEL_GONZALEZ_HARBOUR"/>
    <Resource Reference="JOSE_CARLOS_PALENCIA_GUTIERREZ"/>
    <Resource Reference="ADOLFO_GARANDAL_MARTIN"/>
    <Resource Reference="JOSE_IGNACIO_ESPESO_MARTINEZ"/>
    <Resource>
      <Role>Room</Role>
      <ResourceType Reference="Room"/>
    </Resource>
  </Resources>
  <EventGroups>
    <EventGroup Reference="gr_AllEvents"/>
    <EventGroup Reference="PROGRAMACION"/>
  </EventGroups>
</Event>
```

Figura 14. Ejemplo de Ítem Event

En estos ítems se indica el nombre, su duración, los recursos que los satisfacen y los GroupEvent a los que pertenecen. Estos se añaden también, creando uno por cada asignatura.

5.5.4. Restricciones

Con las restricciones se modela el problema indicando como se tiene que comportar KHE a la hora de crear la solución.

Para ello, hemos creado las siguientes Constraints:

AssignResourceConstraint: Restricción de obligatorio cumplimiento para asignar las aulas. Tiene un coste de una unidad y una función lineal.
AssignTimeConstraint: Restricción de obligatorio cumplimiento para asignar las asignaturas. Tiene un coste de una unidad y una función lineal.
AvoidClashesConstraint: Restricción de obligatorio cumplimiento para evitar las colisiones. XHSTT no evita esto por defecto, por eso esta restricción es necesaria y con ella se impide que los profesores y las aulas estén asignados al mismo tiempo en dos asignaturas. Tiene un coste de una unidad y una función lineal.
PreferTimesConstraint: Restricción de no obligatorio cumplimiento (preferencias) para indicar que los ítem Event referenciados sean asignados en un determinado horario siempre que sea posible. Tiene un coste de una unidad y una función lineal.

```
<AssignResourceConstraint Id="Aulas">
  <Name>Asignar aulas</Name>
  <Required>true</Required>
  <Weight>1</Weight>
  <CostFunction>Linear</CostFunction>
  <AppliesTo>
    <EventGroups>
      <EventGroup Reference="gr_AllEvents"/>
    </EventGroups>
  </AppliesTo>
  <Role>Room</Role>
</AssignResourceConstraint>
```

Figura 15. Constraint básica de asignación de aula

5.6. KHE

Para este proyecto se ha adaptado KHE para obtener una primera solución y comprobar a su vez que las restricciones diseñadas y anteriormente descritas son correctas.

KHE es una librería de software en estándar ANSI C de código abierto creada por Jeff Kingston, cuyo principal propósito es proporcionar una solución rápida y robusta a problemas de asignación de horarios modelados en formato XHSTT. Se distribuye bajo licencia GNU.

KHE ha sido utilizado en competiciones internacionales de resolución de horarios, combinados con otros métodos de cálculo.

5.7. Otros problemas de TimeTabling

Dada la versatilidad de XHSTT para modelar este tipo de problemas, el formato suele ser aplicado a otros similares []. Cómo se ha señalado con anterioridad, existen multitud de planteamientos al problema de asignación de horarios sin existir aún un estándar.

A continuación se reseña el problema horarios de clases universitarias [] ("University Course Timetabling").

5.7.1. Horarios de clases universitarias

El problema horarios de clases universitarias, al igual que “(High) School Timetabling”, consiste en la planificación de un conjunto de clases para un número de aulas y tiempos, con diferentes restricciones que cumplir y sin un formato estándar.

Para modelar este problema se propusieron los modelos Curriculum-Based Course Timetabling (CB-CTT) y Post-Enrolment Course Timetabling (PE-CTT) en la Competición Internacional de Planificadores (ITC2007).

CB-CTT consiste en la planificación semanal de clases en diferentes cursos universitarios dados unas aulas y tiempos concretos.

PE-CTT, por su parte, asume que el alumno ya se ha apuntado a las clases que quiere seguir y la planificación debe resolverse de tal forma que no genere ningún conflicto a ningún estudiante.

En general, con XHSTT se pueden modelar muchos de estos casos, con lo que el problema aquí resuelto es sólo uno de ellos y en otra ocasión se podría ampliar o integrar otras soluciones.

6. DISEÑO E IMPLEMENTACIÓN

En este capítulo se explica el proceso que se ha seguido para el diseño e implementación del sistema. Se especifica la arquitectura de la aplicación, el formato del archivo XHSTT que se utiliza para almacenar los datos y la interfaz gráfica con la que el usuario interactuará con el sistema.

6.1. Arquitectura de la aplicación

En el diseño de la aplicación se ha intentado seguir una arquitectura de tres capas[10]. Este diseño consta de tres niveles: Capa de presentación, capa de negocio y capa de datos. Cada capa cumple una funcionalidad propia y se apoya en la anterior para cumplir su objetivo. Esto permite realizar modificaciones en la aplicación de manera sencilla, sin afectar a las funcionalidades ya existentes, otorgándole robustez y flexibilidad a la aplicación.

Debido a las restricciones impuestas en los requisitos y vistas en el apartado de ANÁLISIS, la aplicación no cumple al 100% con esta filosofía (no cuenta con una base de datos, en su lugar utiliza un archivo XHSTT).

Se ha decidido implementar esta arquitectura, a pesar de no contar con una base de datos, porque facilitaría el mantenimiento de los datos y el cálculo de posibles soluciones al problema.

- **Capa de presentación**: Incluye la interfaz que presenta de manera visual la información e interactúa directamente con el usuario. Esta interfaz debe ser entendible e intuitiva para el usuario. Sólo se comunica con la capa de negocio.

- **Capa de negocio**: Hace de intermediaria entre la capa de presentación y la capa de datos. Implementa las funcionalidades el usuario solicitará a través de la capa de presentación. Recupera y modifica la información de la capa de datos (fichero XHSTT).

- **Capa de datos**: Es donde residen los datos. Normalmente, está formada por uno o más gestores de bases de datos, pero debido a la necesidad de tener que almacenar los datos en un fichero XHSTT para calcular soluciones, este fichero cumple con la función. La capa de negocio realiza la lectura del fichero y las modificaciones pertinente en él.

6.1.1.Capa de *datos*

Como se ha indicado anteriormente, la capa de datos está formada por uno o varios gestores de base de datos. Sin embargo, para esta aplicación ha sido necesario sustituirlo por un fichero con formato XHSTT. En este fichero es donde se almacenarán todos los datos que mostrará la aplicación.

```
<ResourceGroup Id="Class2_MATEMATICAS_2">
  <Name>Clase 2 (grupo 2) de MATEMATICAS</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
<ResourceGroup Id="Class2_MATEMATICAS_3">
  <Name>Clase 2 (grupo 3) de MATEMATICAS</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
<ResourceGroup Id="Class3_DOBLE_1">
  <Name>Clase 3 (grupo 1) de DOBLE</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
<ResourceGroup Id="Class3_DOBLE_2">
  <Name>Clase 3 (grupo 2) de DOBLE</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
<ResourceGroup Id="Class3_DOBLE_3">
  <Name>Clase 3 (grupo 3) de DOBLE</Name>
  <ResourceType Reference="Class"/>
</ResourceGroup>
```

Figura 16. Ejemplo de datos almacenados en código XHSTT

Esta decisión de diseño ha sido tomada como consecuencia a la restricción impuesta de utilizar este formato como almacenamiento de datos. Esto es así porque la herramienta externa utilizada para calcular una solución hace uso de este tipo de archivo.

En este archivo se sigue el formato explicado en el apartado anterior, XHSTT, de este mismo documento. Almacena los datos utilizados para componer los filtros, las restricciones que tiene que cumplir la solución y la solución propuesta por la herramienta externa HVAL.

Es fundamental que el archivo esté bien formado, ya que de otra manera la aplicación no funcionará correctamente.

La capa de negocio es la encargada de trabajar con él, recorriéndolo para leer los datos que contiene y modificándolos en caso de que sea necesario.

6.1.2.Capa de negocio

La capa de negocio es la encargada de implementar las diferentes funcionalidades de la aplicación. En este caso, también se encarga de acceder y gestionar los datos del fichero XHSTT, al no poder contar con un gestor de base de datos.

En ella se implementan diversas funcionalidades, siendo las más destacadas las siguientes:

```
#Formato para los desplegables
profesbutton = Button(text = 'Profesores', size_hint = (None, None), width = 330)
profesbutton.bind(on_release=profes.open)
aulasbutton= Button(text = 'Aulas', size_hint = (None, None), width = 200)
aulasbutton.bind(on_release=aulas.open)
asignsbutton= Button(text = 'Asignaturas', size_hint = (None, None), width = 400)
asignsbutton.bind(on_release=asigns.open)
cursosbutton= Button(text = 'Cursos', size_hint = (None, None), width = 250)
cursosbutton.bind(on_release=curs.open)

aulas.bind(on_select=lambda instance, x: setattr(aulasbutton, 'text', x))
profes.bind(on_select=lambda instance, x: setattr(profesbutton, 'text', x))
asigns.bind(on_select=lambda instance, x: setattr(asignsbutton, 'text', x))
curs.bind(on_select=lambda instance, x: setattr(cursosbutton, 'text', x))
```

Figura 17. Ejemplo de código Python (desplegables)

6.1.2.1. Composición de los filtros

La capa de negocio accede al archivo XHSTT para leer sus datos y componer los filtros que permitirán realizar la carga que el usuario desee. Hay cuatro tipos de filtro diferentes a disposición del usuario: Profesor, Aula, Asignatura y Curso.

Estos filtros se construyen de forma dinámica. De esta manera, si los datos del archivo XHSTT cambian, la composición de los filtros es transparente para el usuario y simplemente verá las nuevas opciones en ellos sin tener que realizar ninguna acción.

Se pueden seleccionar hasta dos filtros simultáneamente para aplicar en la carga.

6.1.2.2. Lectura en el fichero XHSTT

La capa de negocio accede al archivo XHSTT para leer sus datos aplicando los filtros que el usuario haya seleccionado previamente.

Para ello, la aplicación recorre la solución seleccionada de las múltiples posibles, el archivo XHSTT puede almacenar multitud de soluciones, y rellena el horario en función de la selección de los filtros. Las horas que quedan libres se indican con textos estándar para facilitar posteriores modificaciones.

6.1.2.3. Escritura en el fichero XHSTT

La capa de negocio accede al archivo XHSTT para escribir en él los datos modificados en un horario.

Para ello, la aplicación recorre la solución seleccionada aplicando los filtros anteriormente establecidos y elimina los nodos del archivo. Posteriormente, inserta en la solución los datos modificados.

6.1.2.4. Intercambio de horas

La funcionalidad de intercambio de horas permite realizar modificaciones en el horario cargado.

Una vez seleccionada esta funcionalidad (viene seleccionada por defecto), el usuario tiene que escoger un dos asignaturas del horario mostrado para que intercambien sus posiciones. Al seleccionar la primera, esta se marca como seleccionada. Al hacer lo mismo con la segunda, estas intercambian sus posiciones en el horario de manera inmediata.

El intercambio funciona tanto dentro de una pantalla (mañana/tarde) como entre ellas, permitiendo pasar una asignatura de un periodo de tiempo al otro.

6.1.2.5. Asignación de un aula

Esta funcionalidad permite asignar un aula concreta a una asignatura mostrada en el horario.

Para ello, hay que seleccionar el aula deseada en el desplegable de acciones y posteriormente la asignatura a la cual se desea asignar.

6.1.2.6. Visualización de las incidencias

Esta funcionalidad muestra en formato texto las restricciones que no se han cumplido al generar la solución para los diferentes horarios.

Estas incidencias se cargan inmediatamente en la pantalla correspondiente al iniciar la aplicación, permitiendo al usuario consultarlas antes de decidirse por el horario a cargar.

Al igual que las soluciones a los horarios, están ligadas a la solución del archivo XHSTT indicada para ser mostrada.

SI se quieren actualizar las incidencias una vez realizado cualquier cambio en el horario, es necesario utilizar la herramienta externa HSEval para generar el informe correspondiente sobre la solución creada tras las modificaciones.

6.1.3.Capa de presentación

Esta capa está compuesta por la interfaz gráfica con la que el usuario interaccionará para realizar cualquier acción.

Para ello, la interfaz está formada por un conjunto de pantallas, todas accesibles en cualquier momento, pero que mostrarán una u otra información según las acciones realizadas por el usuario.

Para la realización de dichas acciones, se dispone de una serie de botones, tanto en la parte superior de todas las pantallas, como en la inferior. Cada uno de estos botones aporta una funcionalidad a la aplicación.

6.1.3.1. Botones de acciones

La aplicación consta de dos barras de botones, una en la parte superior y otra en la inferior. Estos botones permiten al usuario realizar acciones sobre los horarios y navegar entre las diferentes pantallas.

Figura 18. Botones superiores de la aplicación

- Botones superiores:
  - Botón de acción: Permite realizar cambios en los horarios. Es un desplegable con las diferentes opciones de modificación. Se pueden intercambiar asignaturas o asignar un aula concreta.
  - Botón de reinicio del filtro: Este botón permite vaciar los filtros seleccionados y comenzar de nuevo con otra carga diferente.
  - Botón de filtrado: Su única función es mostrar en una etiqueta el filtro que ha seleccionado el usuario al cargar para que al navegar sepa en todo momento a qué corresponden los datos visualizados en la aplicación.

Figura 19. Botones inferiores

- Botones inferiores:
  - Principal: Visualiza la pantalla principal, con los filtros y el botón de carga.
  - Mañana: Visualiza la pantalla con el horario de mañana de la selección.
  - Tarde: Visualiza la pantalla con el horario de tarde de la selección.
  - Incidencias: Visualiza la pantalla con las restricciones que no se cumplen en formato texto.
  - Guardar: Salva los datos modificados del horario cargado en el fichero XHSTT.
  - Salir: Cierra la aplicación.

6.1.3.2. Pantalla principal

Es la pantalla que se presenta nada más inicializar la aplicación. En ella se realizan todas las operaciones involucradas en la carga de los diferentes horarios. Está compuesta por una serie de botones, asociado cada uno a una funcionalidad diferente.

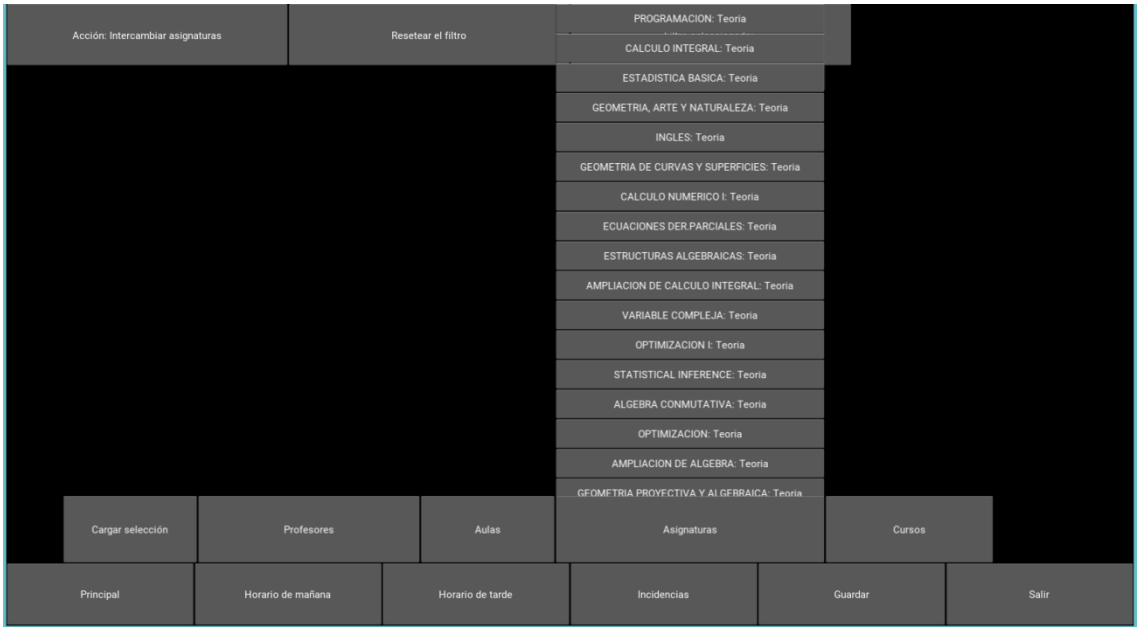

Figura 20. Pantalla principal

Está compuesta por cuatro desplegables que muestran las opciones disponibles para filtrar a la hora de cargar un horario. Estos desplegables se cargan directamente al iniciar la aplicación desde el archivo XHSTT. Esto permite que si los datos referentes a los recursos de este archivo son modificados (por ejemplo, se eliminan o añaden profesores), aparezcan reflejados en los desplegables de manera inmediata.

También incluye el botón asociado a la carga de los horarios. Se permite introducir dos filtros para tener en cuenta a la hora de cargar.

6.1.3.3. Pantalla de mañana

En esta pantalla se muestra el horario de mañana cargado según los filtros aplicados.

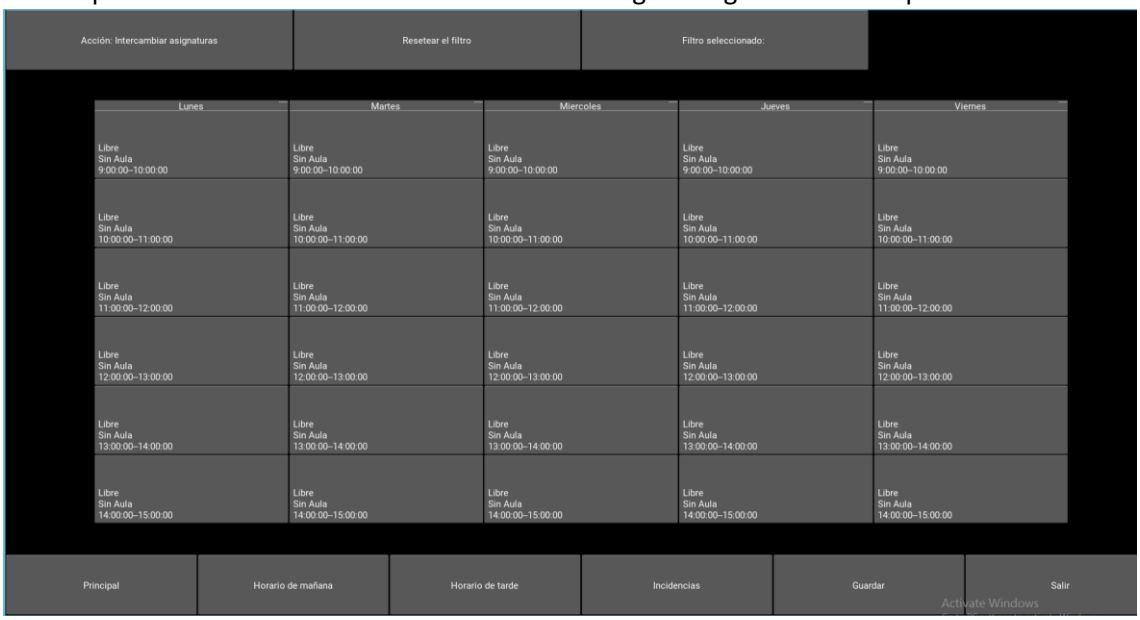

Figura 21. Horario de mañana

Además, permite realizar modificaciones en el horario según la opción seleccionada en el desplegable de acciones.

6.1.3.4. Pantalla de tarde

En esta pantalla se muestra el horario de tarde cargado según los filtros aplicados.

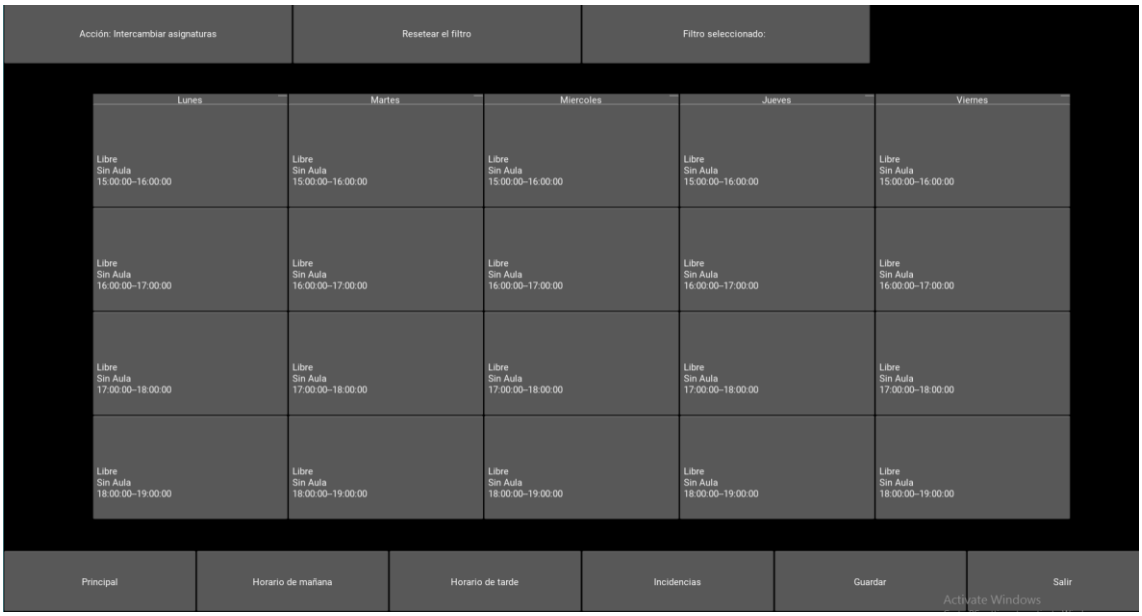

Figura 22. Horario de tarde

Además, permite realizar modificaciones en el horario según la opción seleccionada en el desplegable de acciones.

6.1.3.5. Pantalla de incidencias

Muestra las restricciones que no se han cumplido al generar los horarios de la solución en formato texto. Para ello, se basa en un informe generado por la herramienta externa HEVAL.

Si se realizan modificaciones en el horario, se debe volver a calcular este informe para que las incidencias mostradas correspondan con los datos de la solución.

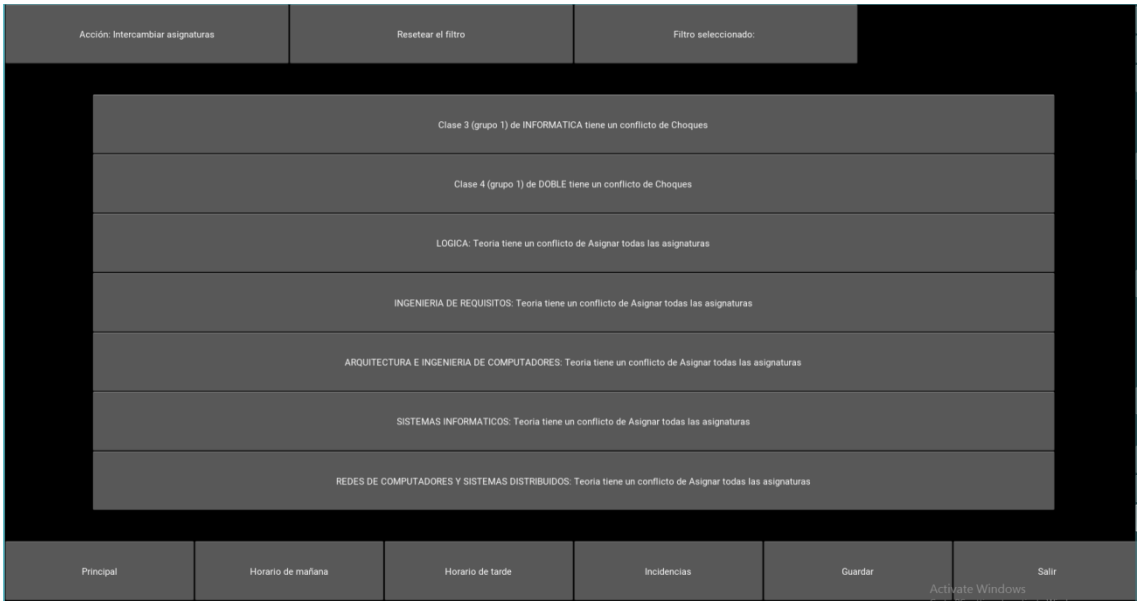

Figura 23. Pantalla de incidencias

7. EVALUACIÓN Y PRUEBAS

Se describen las pruebas que se han realizado durante el desarrollo del software para comprobar que el sistema funciona de manera correcta. Además sirve para verificar que se cumplen los requisitos recogidos en la fase de análisis.

Durante el desarrollo del software han surgido dos grandes grupos de pruebas: Las pruebas funcionales y las pruebas no funcionales [11].

En el primer grupo se engloban las siguientes: Pruebas unitarias, pruebas de regresión y pruebas de integración.

El segundo está formado por la pruebas de usabilidad.

7.1. Pruebas funcionales

Este tipo de pruebas tienen la finalidad de comprobar que el software cumple correctamente con los requisitos establecidos durante la fase de análisis.

7.1.1. Pruebas unitarias

Estas pruebas sirven para verificar el correcto funcionamiento de cada trozo del software. La realización de estas pruebas ha ido a la par que se realizaba la implementación de cada funcionalidad de la aplicación.

El método que se ha utilizado es el de Debug por la línea de comandos. Se ha ido imprimiendo por pantalla las entradas introducidas y evaluando la solución aportada por el sistema en función de la solución esperada.

Como el desarrollo no se ha realizado en un entorno de programación (ha sido completamente realizado con el editor de ILDE para Python), no se podido introducir puntos de ruptura, por lo que se ha ido mostrando por pantalla los datos necesarios para la depuración.

7.1.2. Pruebas de regresión

Este tipo de pruebas tienen la finalidad de encontrar errores o faltas de funcionalidad que se hayan producido como consecuencia de la modificación del código ya existente.

Al optar por un modelo iterativo incremental, este tipo de pruebas es indispensable para garantizar que las nuevas modificaciones en la aplicación y sus funcionalidades no impactan sobre otras existentes anteriormente.

El método utilizado ha sido repetir las pruebas unitarias en las iteraciones en las que la funcionalidad antigua se ha visto impactada por la introducción de otra funcionalidad nueva, garantizando que no se han provocado errores en el trabajo anterior.

7.1.3.Pruebas de integración

Estas pruebas se realizan una vez que todas las pruebas unitarias de una funcionalidad han sido pasadas con éxito y esta nueva funcionalidad se añade a la aplicación.

Tienen la finalidad de comprobar que todas las funcionalidades pueden coexistir sin provocar errores en el resto, probando las funcionalidades en grupo.

Se centrar sobre todo en la comunicación entre los diferentes componentes del sistema.

A lo largo del desarrollo se han utilizado para comprobar que la capa de negocio recogía y almacenaba los datos de forma correcta en el fichero XHSTT o para comprobar que la capa gráfica se comunicaba correctamente con la de negocio.

También se han mantenido reuniones con la jefa de estudios (usuario final) Beatriz Porras, en las que se han sugerido añadir funcionalidades no contempladas en la primera toma de requisitos y que gracias al modelo incremental empleado en el desarrollo se han podido integrar con facilidad, como la posibilidad de asignar directamente un aula.

7.2. Pruebas no funcionales

Este tipo de pruebas verifican que se cumple con un requisito no funcional. Se han utilizado para comprobar que la interfaz era agradable e intuitiva para el usuario final.

7.2.1.Pruebas de usabilidad

Estas pruebas sirven para evaluar el software junto al usuario final. Como se ha mencionado con anterioridad, además de la primera reunión de toma de requisitos, se han mantenido otro par más para ir mostrando el estado y apariencia de la aplicación al usuario final a lo largo del desarrollo.

Han servido para cerciorarse de que el usuario final estaba satisfecho con la funcionalidad. Tras estas pruebas, se realizaron modificaciones en la interfaz, por ejemplo, la inclusión de un nuevo botón en la barra superior para saber en todo momento que filtro había utilizado el usuario en la carga del horario, evitando que este tuviera que volver a la pantalla principal a consultarlo.

8. CONCLUSIONES Y TRABAJOS FUTUROS

Tras concluir con el desarrollo del proyecto toca evaluar los logros alcanzados comparándolos con los objetivos marcados al comienzo del mismo.

En este capítulo se presentan las conclusiones alcanzadas una vez finalizado el proyecto y se proponen algunas nuevas funcionalidades a añadir al sistema.

8.1. Conclusiones

A continuación se expondrán las conclusiones obtenidas tras la finalización del trabajo. Se mostrarán desde un punto de vista técnico como personal, indicando las dificultades surgidas durante el desarrollo.

Desde la jefatura de estudios de la Facultad de Ciencias de la Universidad de Cantabria surge la necesidad de disponer de una aplicación para gestionar los horarios de los diferentes cursos impartidos en el centro. Este objetivo principal ha sido satisfecho con éxito.

Existías ciertas restricciones a la hora de realizar el proyecto. El director del mismo es el que se encargará de su mantenimiento, por lo que puso como requisito que la aplicación se desarrollara en lenguaje Python y con el Framework Kivy. Además, los datos serían almacenados en un archivo XHSTT.

Por mi parte, nunca había trabajado con ninguno de estos elementos, por lo que el aprendizaje de todos ellos ha sido el mayor problema al que me he tenido que enfrentar en el desarrollo de la aplicación.

Si bien el proyecto ha ido avanzando más rápido a medida que pasaba el tiempo, al comienzo supuso un reto bastante grande.

Junto a esto, el otro gran problema durante el desarrollo ha sido la elaboración del fichero XHSTT debido a su estructura compleja y estricta. El director también colaboro a su elaboración, así como otro estudiante.

El problema que generaba este archivo era de gran envergadura, ya que modificarlo suponía algunas veces revisar todo el código de comunicación con él.

Desde un punto de vista personal, la experiencia obtenida durante el desarrollo del proyecto ha sido muy satisfactoria, aunque alguna vez me haya supuesto un gran trabajo y me haya costado seguir el ritmo y los plazos marcados. Pero durante el desarrollo he adquirido gran cantidad de conocimientos sobre tecnologías que desconocía hasta el momento.

8.2. Trabajos futuros

El trabajo se ha concluido con éxito, alcanzando los objetivos marcados en su comienzo. Sin embargo, hay puntos en los que esta aplicación puede mejorarse, añadiendo funcionalidad a la misma e intentando facilitar su utilización aún más.

La mejora más necesaria sería la integración, tanto del cálculo de los horarios como de los informes con las incidencias, en la propia aplicación. Para ello habría que integrar tanto una versión adaptada para nuestra solución del cálculo de soluciones de KHE como evaluador de soluciones HSEval.

La siguiente mejora que propondrías sería la creación de un documento de salida (por ejemplo, en formato Excel) para poder imprimir los horarios creados.

En cuanto a lo relativo al modelado del problema, se podrían realizar añadidos, cómo establecer que las horas de laboratorio deban ser en grupos de dos horas.

Por último, se podría realizar una mejora en la interfaz gráfica, para que esta tuviera un acabado más amigable y profesional.

ANEXO I. MANUAL DE USO

Esta guía pretende servir como guía para la utilización correcta de la aplicación a cualquier usuario que tenga que trabajar con ella.

Está dividida en dos partes. En la primera, se indican los pasos a seguir que tiene que realizar el administrador de la aplicación.

Administrador

El administrador de la aplicación tiene asignadas dos tareas. Es el encargado de actualizar los datos del fichero XHSTT que se utiliza en la aplicación y también llevará a cabo la instalación de todo el software necesario para su ejecución.

Este fichero es de suma importancia y de su correcta formación depende el correcto funcionamiento de la aplicación.

Actualización de los datos del fichero

Junto al código de este proyecto se incluye una primera versión de este documento, con su correspondiente solución e informe.

Este fichero servirá de base para futuras actualizaciones. El trabajo del administrador consistirá principalmente en actualizar la información de dicho documento. En principio, esta información deberá ser actualizada cada año, estableciendo los cambios necesarios, sobre todo en lo referente a los profesores.

Actualización de la solución

Para la generación de la primera (u otra más) solución es necesario pasar el archivo XHSTT por el software KHE.

Se generará la solución sobre el archivo XHSTT proporcionado, añadiéndola al final si ya existe otra. El software, junto a su manual de instrucciones e instalación está disponible en el sitio: http://sydney.edu.au/engineering/it/~jeff/khe/

Instalación de la aplicación

La otra tarea del administrador será instalar el software. A continuación se describe la instalación del software requeridopara su ejecución en Windows 8.

- Instalación de Python

En este momento, las últimas versiones de Python (3.6 y 3.5) no son compatibles con Kivy, por lo que se instalará una anterior. Utilizaremos la versión 3.4.3. Se puede descargar desde la página oficial de Python, desde el enlace:
https://www.python.org/downloads/release/python-343/

- Instalación de Kivy

Esta instalación se realizará desde la línea de comandos, instalando tanto Kivy como sus dependencias. Los comandos a introducir son los siguientes:

Instalación de los últimos Pip y Wheel de Python:
py -m pip install --upgrade pip wheel setuptools

Dependencias con gstreamer
py -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
py -m pip install kivy.deps.gstreamer

Instalación de Kivy
py -m pip install kivy

Actualización del fichero

Si se desea, se puede cambiar el fichero utilizado, pero siempre con mucho cuidado. El fichero tiene que tener en cuenta que los tiempos establecidos para los horarios son de seis horas por las mañanas y de cuatro por las tardes.

Para cambiar de fichero, sólo es necesario introducir el nombre del nuevo en el trozo de código siguiente de la clase main.py:

```
#variables globales
dias=['Lunes','Martes','Miercoles','Jueves','Viernes']
horarioPrincipal = horario(dias,timedelta(hours=9))
filterTotal = set()
documento = 'datos/outfile_nuevo_solucion.xml'

class TestApp(App):
    def build(self):
```

Figura 24. Cambio de documento

Usuario (Jefe de estudios)

Inicialización de la aplicación

Para ejecutar la aplicación tan sólo es necesario hacer doble click sobre el archivo main.py. Esto inicializará la aplicación, mostrando la pantalla principal.

Filtrado

Una vez en la pantalla principal, el usuario debe seleccionar un filtro para posteriormente realizar la carga.

El filtrado se realiza mediante la selección del dato deseado en los desplegables de la pantalla principal.

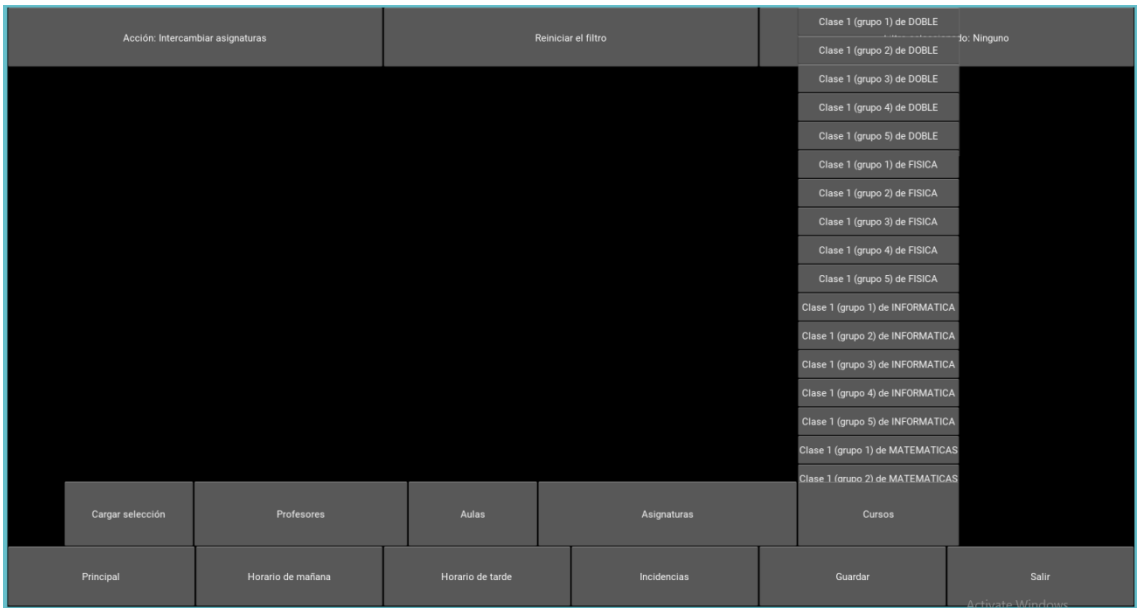

Figura 25. Selección del filtro(curso)

El filtrado se puede realizar por cuatro campos diferentes: Profesores, Aulas, Asignaturas y Cursos.

Carga de datos

La carga de datos se realiza tras la selección de un filtro. Para ello hay disponible un botón dedicado en la pantalla principal. Tras pulsarlo, se cargará el horario según el filtro.

Para visualizar los datos cargados, hay que desplazarse a las ventanas de "Horario de mañana" y "Horario de tarde".

Lunes	Martes	Miercoles	Jueves	Viernes
ESTADISTICA BASICA: Teoria AULA 6 9:00:00–10:00:00	FISICA BASICA EXPERIMENTAL III: LA MATERIA Y SUS PROPIEDADES: Teoria AULA 7 9:00:00–10:00:00	PROGRAMACION: Teoria AULA 1 9:00:00–10:00:00	LABORATORIO MULTIDISCIPLINAR: Laboratorio LABORATORIO 3 9:00:00–10:00:00	PROGRAMACION: Laboratorio LABORATORIO 1 9:00:00–10:00:00
ESTADISTICA BASICA: Teoria AULA 6 10:00:00–11:00:00	PROGRAMACION: Laboratorio LABORATORIO 1 10:00:00–11:00:00	F.E.IV PRACTICAS EWB: Laboratorio LABORATORIO 4 10:00:00–11:00:00	LABORATORIO MULTIDISCIPLINAR: Teoria AULA 1 10:00:00–11:00:00	FISICA BASICA: Teoria AULA 8 10:00:00–11:00:00
FISICA EXPERIMENTAL: Teoria AULA 6 11:00:00–12:00:00	ESTADISTICA BASICA: Teoria AULA 6 11:00:00–12:00:00	LABORATORIO DE FISICA BASICA III: Laboratorio LABORATORIO 4 11:00:00–12:00:00	PROGRAMACION: Teoria AULA 7 11:00:00–12:00:00	ESTADISTICA BASICA: Teoria AULA 6 11:00:00–12:00:00
PROGRAMACION: Teoria AULA 7 12:00:00–13:00:00	LABORATORIO MULTIDISCIPLINAR: Laboratorio LABORATORIO 1 12:00:00–13:00:00	LABORATORIO DE FISICA BASICA IV: Laboratorio LABORATORIO 4 12:00:00–13:00:00	LABORATORIO DE FISICA BASICA IV: Laboratorio LABORATORIO 1 12:00:00–13:00:00	LABORATORIO DE FISICA BASICA III: Laboratorio LABORATORIO 3 12:00:00–13:00:00
F.E.IV PRACTICAS EWB: Laboratorio LABORATORIO 4 13:00:00–14:00:00	LABORATORIO MULTIDISCIPLINAR: Teoria AULA 7 13:00:00–14:00:00	FISICA BASICA: Teoria AULA 8 13:00:00–14:00:00	FISICA BASICA EXPERIMENTAL III: LA MATERIA Y SUS PROPIEDADES: Teoria SEMINARIO INFORMÁTICA 13:00:00–14:00:00	FISICA EXPERIMENTAL: Teoria AULA 3 13:00:00–14:00:00
Libre Sin Aula 14:00:00–15:00:00	Libre Sin Aula 14:00:00–15:00:00	Libre Sin Aula 14:00:00–15:00:00	Libre Sin Aula 14:00:00–15:00:00	Libre Sin Aula 14:00:00–15:00:00

Figura 26. Ejemplo de horario cargado(Mañana)

Modificaciones: Intercambio de horas

Una vez cargado el horario se pueden realizar modificaciones en él.

El intercambio de horas funciona de manera sencilla. La primera vez que haces click sobre un elemento del mismo, se marca para ser intercambiado por otro:

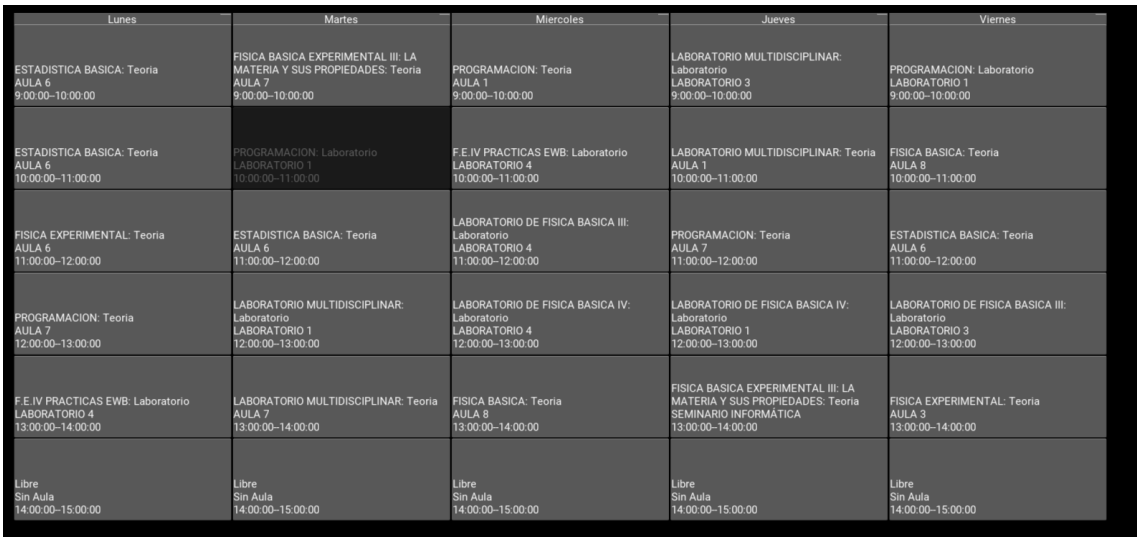

figura 27. Primer elemento marcado

Tras seleccionar el segundo elemento, estos se intercambian. Este intercambio puede ser realizado incluso entre las dos partes del horario, mañana y tarde.

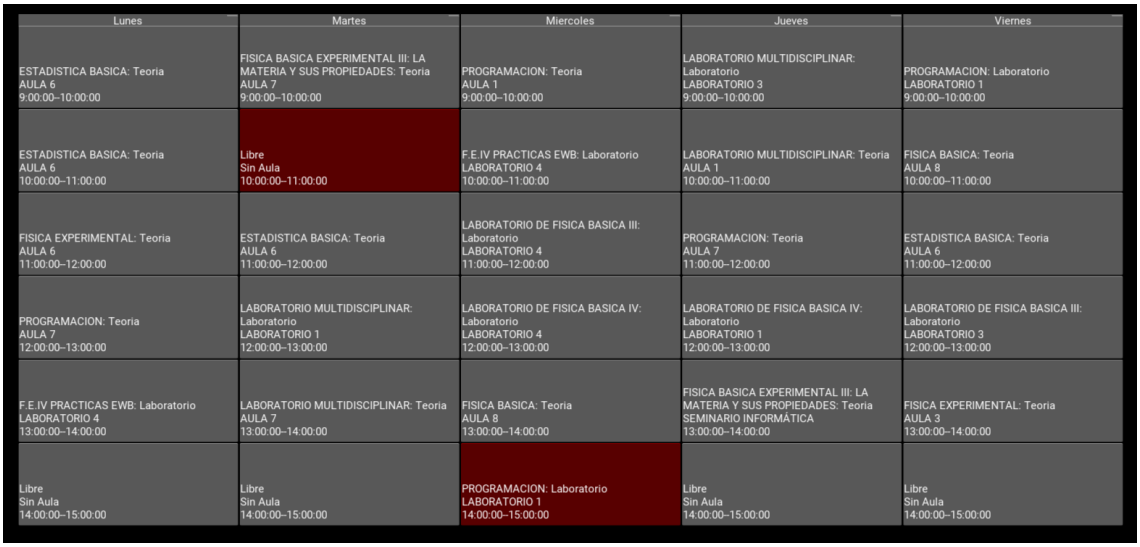

Figura 28. Elementos intercambiados

Modificaciones: Asignación de aula

La otra modificación permitida a un horario es la asignación directa de una aula. Para ello, es necesario seleccionar el aula que se quiere asignar en el desplegable superior y marcar el elemento al que se quiere asignar:

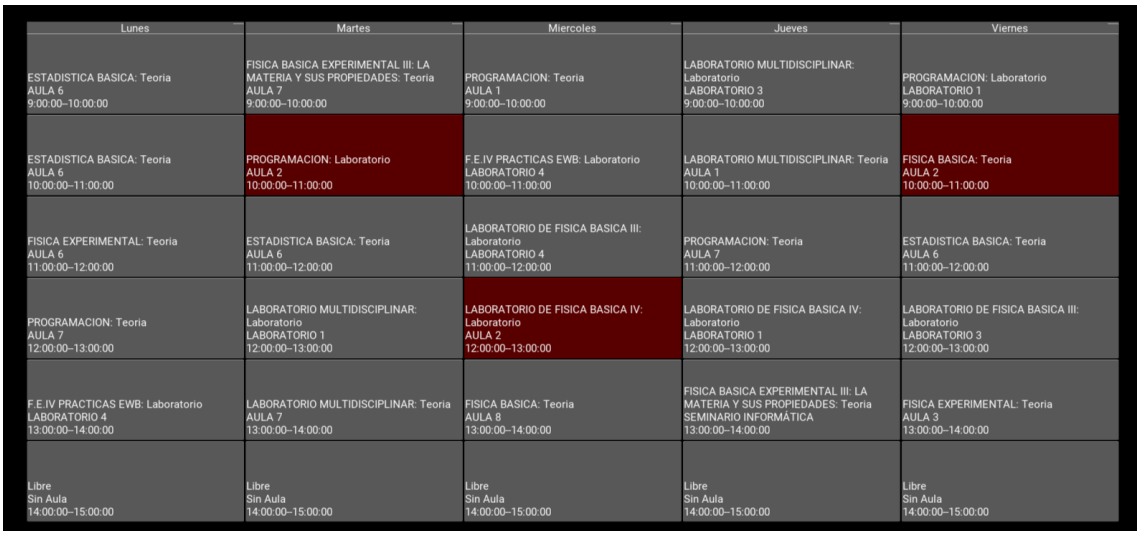

Figura 29. Ejemplo de asignación de aula (AULA 2)

Si se intenta asignar un aula en medio de un intercambio, se intercambiarán los elementos seleccionados y, posteriormente estará disponible la asignación de aula.

Guardado de los datos

Para salvar los posibles cambios realizados en los horarios, hay dedicado un botón en la parte inferior de la aplicación.Simplemente con pulsarlo los datos se guardarán en el fichero XHSTT en el formato correcto. Los elementos del horario recuperarán su color inicial.

Visualización de incidencias

La visualización de las incidencias del horario se realiza desde su propia pantalla. Estas se cargan de manera automática nada más inicializar la aplicación.

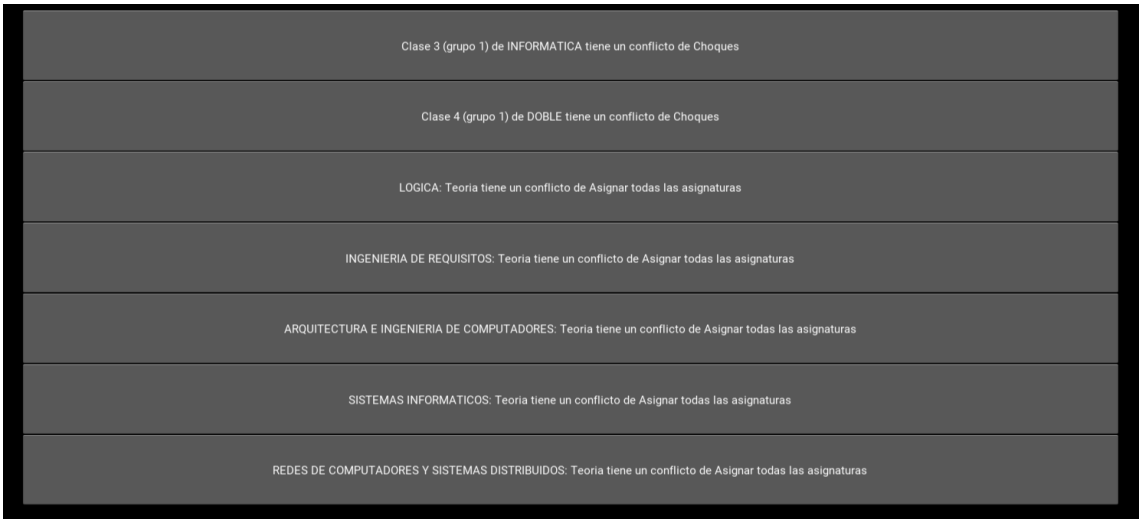

Figura 30. Incidencias

Para actualizarlas, es necesario actualizar el informe de la solución.

Reinicio del filtro

Para esta acción hay dedicado un botón en la barra superior. Simplemente con pulsarlo, el filtro se limpiará y se podrá cargar otro horario según el filtro deseado.

Actualización del informe

Para actualizar el informe de una solución, hay que dirigirse al sitio web:
http://www.it.usyd.edu.au/~jeff/cgi-bin/hseval.cgi

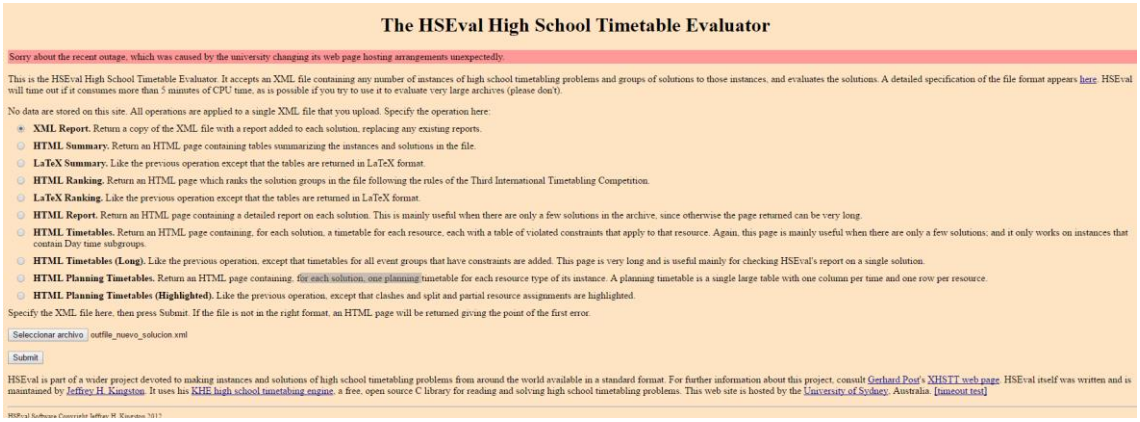

Figura 31. Web del evaluador

Una vez en él, el usuario deberá seleccionar la primera opción 'XML Report', seleccionar desde los archivos locales el documento del que se quiere actualizar el informe y enviarlo.

Devolverá una copia del archivo XHSTT con el nuevo informe añadido a las soluciones.

REFERENCIAS

[1] University of Twente. Centre for Telematics and Information Technology CTIT [sitio web]. [Consulta: 29 Octubre 2016]. Disponible en: https://www.utwente.nl/ctit/hstt/

[2] Jeffrey H. Kingston (2012*) High School Timetable File Format Specification: Constraints.*[Sitio Web]. [Consulta 5 Noviembre]. Disponible en: http://www.it.usyd.edu.au/~jeff/cgi-bin/hseval.cgi?op=spec&part=constraints

[3] Pytel, P., Uhalde, C., Ramón, H. D., Castello, H., Tomasello, M., Pollo Cattaneo, M. F. (2011). *Ingeniería de requisitos basada en técnicas de ingeniería del conocimiento*.http://sedici.unlp.edu.ar/bitstream/handle/10915/20070/Documento_completo.pdf?sequence=1

[4] Jeffrey H. Kingston (2012)TheHSEval High SchoolTimetableEvaluator. [Sitio Web]. [Consulta 5 Noviembre]. Disponible en:http://www.it.usyd.edu.au/~jeff/cgi-bin/hseval.cgi

[5] Pilgrim, M.(2009)*Dive Into Python 3* Disponible en: http://www.diveintopython3.net/index.html

[6] Kivy[Sitio Web].https://kivy.org/#home

[7] Python Software Foundation [Sitio Web].Disponible en:https://www.python.org/psf-landing/

[8] Instituto Nacional de Tecnologías de la Comunicación. 2009. Ingeniería del software: metodologías y ciclos de vida. Laboratorio Nacional de Calidad del Software.

[9] Ceria, S. 2002. Casos de uso. Un Método Práctico para Explorar Requerimientos. Ingeniería de Software I.

[10] Eastern Software Systems Pvt. Ltd. 2006. Arquitectura de Tres Capas. [Consulta: 10 junio 2016]. Disponible en: http://www.managinf.com/arquitectura.pdf

[11] Sommerville, I. 2011. Sofware Engineering. 9ª Edición. Addison-Wesley.

[12] Fonseca, G., Gambini Santos, H., Carrano, E., Stidsen, T. (2016)Modelling and Solving University Course Timetabling Problems Through XHSTT. Disponible en: http://www.patatconference.org/patat2016/files/proceedings/paper_12.pdf

[13] Jeffrey H. Kingston (2012)High School Timetable Data Format Specification Disponible en: http://www.it.usyd.edu.au/~jeff/cgi-bin/hseval.cgi?op=spec

[14] Carter, M. W., Laporte, G., & Lee, S. Y. (1996). Examination timetabling: algorithmic strategies and applications. The Journal of the Operational Research Society, 74, 373–383.

[15] McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., & Qu, R. (2007). The second international timetabling competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/-Exam/v4. 0/17, Queen's University, Belfast.

[16] Kingston, J. (2014). KHE14: An algorithm for high school timetabling. In Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (pp. 498-501).