# Universidad de las Ciencias Informáticas

**Facultad 5**


# Título: “Aplicación Web para la confección de Horarios Docentes”

**Trabajo de Diploma para optar por el título de**

**Ingeniero en ciencias Informáticas**

**Autor:** Yerandy Reyes Fabregat

**Tutores:** Lic. Pedro Rubén Quintero Rojas

Lic. Oliver Fernández Gil

Ciudad de La Habana, Cuba.

Junio, 2009

## DATOS DE CONTACTO

Lic. Pedro Rubén Quintero Rojas, graduado en la carrera de Licenciatura en Ciencias de la Computación en el año 2004. Actualmente es profesor instructor en la Universidad de las Ciencias Informáticas y la Universidad de la Habana. Correos electrónicos: pedror@uci.cu, pqr@uh.cu

Lic. Oliver Fernández Gil, graduado en la carrera de Licenciatura en Ciencias de la Computación en el año 2005. Actualmente es profesor instructor en la Universidad de las Ciencias Informáticas. Correo electrónico oliver@uci.cu

**AGRADECIMIENTOS**

*Agradezco:*

*A Dios, por ser mi guía en todo momento.*

*A mis padres por el esfuerzo y la dedicación que me han brindado.*

*A Liu por estar a mi lado en las buenas y malas.*

*A mis abuelos, tíos, primos, en especial a mis tíos Rubén y Diana por su apoyo incondicional.*

*A mis tutores por la confianza depositada en mí.*

*A todos mis profesores por todo lo que me han enseñado durante mi carrera.*

*A mis compañeros de aula.*

*A todas las personas que me han ayudado a lograr el sueño de ser ingeniero.*

**DEDICATORIA**

*A toda mi familia...*

**RESUMEN**

El presente trabajo tiene como objetivo digitalizar el proceso de confección del horario docente de la Facultad 5 de la Universidad de las Ciencias Informáticas, permitiendo además su explotación en otros centros de la educación superior cubana a través de una aplicación web.

La herramienta permite confeccionar un horario docente de forma cómoda y sencilla, disminuyendo o evitando los errores humanos. Se ha realizado una investigación de los principales requisitos a tener en cuenta para la creación de un horario docente, en vista a ofrecer al personal involucrado la información que necesite en un determinado momento.

Para el desarrollo de esta herramienta se hizo una investigación de las diferentes tecnologías que servirían de base para su desarrollo y explotación. Se determinó su implementación en lenguaje de programación PHP, haciendo uso de las facilidades de la plataforma Symfony. Como gestor de base datos se usará MySQL, el cual soportará el almacenamiento de toda la información del sistema.

Todo el proceso de desarrollo estará guiado por la Metodología XP, la cual se adapta perfectamente a las características del producto y del equipo de desarrollo.

Este software contará con un sistema para expandir sus funcionalidades, de manera que pueda ser fácilmente adaptado para su utilización en cualquier centro de la educación superior, y permitiendo la mejora constante del sistema sin alterar su núcleo principal.

**PALABRAS CLAVE**

Horario Docente, Metodología XP, Symfony, PHP, MySql

**TABLA DE CONTENIDOS**

## INTRODUCCIÓN

Desde hace relativamente poco tiempo los avances tecnológicos en el campo de la informática han revolucionado el mundo. Hoy en día, el uso de estas tecnologías se ha generalizado en tal medida, que es difícil mantenerse al margen de esta corriente.

La dirección de nuestro país comprendiendo la necesidad de informatizar la sociedad cubana, ha puesto en práctica numerosos programas para aumentar el número de profesionales en la rama de la informática y las ciencias de la computación, además ha realizado importantes esfuerzos económicos encaminados a familiarizar a todos los sectores de nuestro pueblo con el uso de estas nuevas tecnologías.

Las computadoras se han extendido a lo largo y ancho de la isla, por lo que replantearse algunos procesos que cotidianamente se realizan de forma manual para determinar si haciendo uso de las tecnologías informáticas, muchas veces subutilizadas, podría resultar en un aumento de la eficiencia y calidad de los mismos.

Un problema recurrente en los centros de la enseñanza superior cubana es el tema de la confección de los horarios docentes, labor que en la mayoría de estos se realiza de forma manual y por tanto poco eficiente. Los problemas de horarios consisten esencialmente en fijar una secuencia de encuentros entre profesores y estudiantes en un período de tiempo predefinido, satisfaciendo un conjunto de restricciones de varios tipos. Dependiendo del número de recursos involucrados, la resolución de problemas de este tipo de forma manual puede tomar desde algunas horas hasta varios días y no se garantiza que los resultados obtenidos sean los más adecuados para todas las partes involucradas.

Automatizar los procesos de confección de horarios docentes ha sido una tarea a la que se han dedicado numerosas investigaciones por lo que la bibliografía referente al tema es extensa. Algunos autores plantean que estos problemas no pueden ser totalmente automatizados debido a que aunque exista información que permita afirmar que unas soluciones son mejores que otras, esta información no se expresa trivialmente a un sistema automático; además de esto, los espacios de búsqueda son normalmente muy extensos por lo que la intervención humana es fundamental para simplificarlos. No obstante, numerosas soluciones computacionales a este tipo de problemas han surgido en los últimos años. Algunas de estas aplicaciones permiten encontrar soluciones de manera automática brindando la posibilidad al usuario de retocar de forma manual los resultados obtenidos. Por otro lado, las denominadas aplicaciones semiautomáticas, ayudan a la confección de horarios docentes requiriendo

una mayor intervención de los usuarios, aun así simplifican notablemente el tiempo invertido para esta tarea.

La facultad número cinco de la Universidad de las Ciencias Informáticas cuenta con una matrícula de 1000 estudiantes agrupados en 35 grupos de los distintos años, en ella trabajan más de 150 profesores que con la ayuda de más de 160 alumnos ayudantes imparten las clases pertenecientes a más de 50 asignaturas. La confección del horario docente es realizada manualmente por una persona haciendo uso de una hoja de cálculo de Microsoft Excel, este proceso normalmente tarda semanas desde que se inicia un semestre hasta que los usuarios obtienen una versión oficial de sus horarios, etapa en la que se hacen varias propuestas buscando la máxima conformidad de todas las partes involucradas. Debido al gran volumen de información a manejar, las irregularidades que se puedan presentar y la poca adecuación de la herramienta utilizada, el proceso de confección del horario docente, se torna complicado, tedioso y lento.

Situaciones similares ocurren en la mayoría de los centros de enseñanza de la educación superior de nuestro país, por lo que surge el siguiente **problema científico** ¿Cómo mejorar el proceso de confección de los horarios docentes?

El **objeto de estudio** es el proceso de confección del horario docente en la Universidad de las Ciencias Informáticas. El **campo de acción** es el proceso de confección del horario docente en la facultad cinco de la Universidad de las Ciencias Informáticas.

Para dar solución al problema antes mencionado se propone como **objetivo general**:

Implementar una solución informática usando tecnología web que asista el proceso de confección de horarios docentes en la facultad cinco de la Universidad de las Ciencias Informáticas.

Para lograr este objetivo se trazan las siguientes **tareas investigativas:**

- Selección de la información existente, sobre la creación de horarios docentes, que resulte viable para este trabajo.
- Estudio de sistemas informáticos que realizan esta función en el ámbito internacional y nacional.
- Análisis del proceso manual de creación de un horario docente, para identificar posibles mejoras al proceso.
- Entrevista a los futuros usuarios del sistema para identificar los requisitos del software.
- Estudio de las metodologías de desarrollo de software, tecnologías web y lenguajes de programación con el fin de seleccionar y aplicar los más adecuados.

- Realización del ciclo de desarrollo completo del software.

Para realizar estas tareas se hará uso de los siguientes **métodos científicos:**

**Método Dialéctico-Materialista** como método fundamental de estudio de todas las ciencias.

**Métodos Teóricos:**

- Analítico – Sintético
- Histórico – Lógico

**Métodos Empíricos:**

- Observación
- Entrevista

Como posibles **aportes prácticos** se pueden citar los siguientes**:**

1. Disminución del tiempo de confección de los horarios docentes.
2. Facilidad de búsqueda de información por parte del personal involucrado.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En el presente capítulo se tratarán los principales conceptos relacionados con el tema que se propone. Se aborda la existencia de sistemas similares al que se pretende desarrollar en el ámbito nacional e internacional. Se lleva a cabo un estudio de las metodologías de desarrollo de software y las características de las principales tecnologías y herramientas existentes utilizadas para el desarrollo de este tipo de aplicaciones. Se hará una valoración de las ventajas y desventajas de cada elemento tratado para seleccionar los más adecuadas para la realización de este trabajo.

## 1.2 Planificación docente

La planificación implica un proceso consciente de estudio y selección del mejor curso de acción a seguir, estableciendo prioridades frente a una variedad de alternativas posibles y factibles de acuerdo a los recursos con que se cuenta. La planificación docente no es más que la organización coherente y funcional de un conjunto de actividades destinadas al eficaz aprendizaje de los estudiantes tratando siempre de lograr el óptimo uso de los recursos disponibles, como pueden ser los inmuebles o los recursos humanos como profesores, en un marco de tiempo adecuado.

### 1.2.1 Factores que inciden y participan en la planificación docente

María Onelia Chiang Molina en su libro "Higiene de la actividad docente" aborda el tema de la planificación concibiéndola como la primera función administrativa que sirve de base para muchas otras funciones, la cual determina por anticipado cuáles son los objetivos que deben cumplirse y qué debe hacerse para alcanzarlos; por tanto, es un modelo teórico para actuar en el futuro. La planificación comienza por establecer los objetivos y detallar los planes necesarios para alcanzarlos de la mejor manera posible. Determina a dónde se pretende llegar, qué debe hacerse además del cómo, cuándo y en qué orden deben suceder los acontecimientos.

Se concibe dicha actividad como un proceso complejo, pues se deben tener en cuenta muchos aspectos de singular importancia referentes a la actividad docente, ya que es mediante ésta como se logra la asimilación de conocimientos científicos y la formación de habilidades correspondientes, objetivos y resultados esenciales de la propia actividad para los educandos, la cual es una forma de actividad cognoscitiva dirigida mediante el proceso de enseñanza en la escuela, regida por un conjunto de características que le dan elementos de complejidad, entre los que encontramos:

- Basada en contenidos previamente determinados en el plan de estudio de la carrera y en programas establecidos.

- Puede hacerse por bloques lectivos, ciclos o niveles, en dependencia de la carrera, el curso o el nivel de enseñanza.

Unido a esto debe considerarse la organización de la actividad docente desde el punto de vista higiénico, la cual está muy relacionada con uno de los principios más importantes de la organización científica del trabajo: el principio de la optimización, el cual exige la selección de una variante óptima de la actividad para evitar trastornos a los estudiantes. Esta organización supone:

- La existencia de un balance de carga docente normalizado (equilibrio de clases para un grupo de estudiantes en un período).

- Equilibrio de los tipos de actividades docentes (exámenes, conferencias, clases prácticas, seminarios, laboratorios, etc.), pues está comprobado que muchas actividades evaluativas alteran los biorritmos de aprovechamiento académico, de alimentación, sueño y descanso, llegando a ocasionar alteraciones en el calibre de los vasos sanguíneos y en la respiración.

- Planificación adecuada de medios y materiales de enseñanza, así como de volumen de la información. (Molina, 1995)

### 1.2.2 Planificación docente en la educación superior cubana

Históricamente, en la solución a muchas situaciones y problemas organizativos a los que se ha enfrentado el hombre, ha sido de especial importancia su capacidad para planificar las actividades. El hecho de conducir un proceso se evidencia en cualquier área o sector de la sociedad, desde la producción industrial hasta la educación en todos sus niveles. Las Universidades forman el personal capacitado para asumir tareas específicas de diversa índole, que son animadas por procesos que tienen lugar en su interior, llamados procesos sustantivos, destacándose entre estos el Proceso de Enseñanza Aprendizaje(PEA), que tiene carácter sistémico, organizado y planificado por un personal especializado con la finalidad de formar al profesional que requiere la sociedad.

El PEA en su organización como sistema, requiere de una distribución adecuada de las diferentes asignaturas: por semestres, semanas, días y horas lectivas, siempre teniendo en cuenta las exigencias propias de las disciplinas que se imparten a los estudiantes y su importancia para la especialidad, así como las características de cada grupo de educandos, el tipo de enseñanza, la forma de organización

de sus actividades docentes y las actividades extra docentes que se realizan, ya que estos factores, y otros, afectan directamente la manera en que han de ser distribuidos los contenidos a lo largo de la formación del profesional; elementos claves para lograr una planificación balanceada de acuerdo con los principios y normas de la higiene de la actividad docente. En el caso particular de la Enseñanza Superior en Cuba, teniendo en cuenta los cambios que han venido ocurriendo en ella a raíz de la Universalización, que conlleva al incremento del número de estudiantes y a la demanda de docentes, unido a la heterogénea organización de las actividades docentes, que incluye distintas modalidades tales como presenciales, semi-presenciales, a distancia, etcétera; conduce a un proceso de planificación complejo, en extremo dinámico y altamente propenso a irregularidades y descoordinaciones que tienden a afectar la calidad del Proceso Docente Educativo. (Molina, 1995).

Esta situación es aún más crítica en la Universidad de las Ciencias Informáticas, la cual es una universidad atípica, de corte productivo, donde el sistema de planificación docente se ve seriamente afectado.

### 1.3 Sistemas automatizados existentes vinculados al campo de acción

En la actualidad existen múltiples soluciones informáticas para la confección de horarios docentes.

La empresa española Peñalara mantiene el software **Generador de Horarios para Centros de Enseñanza**, actualmente en la versión 2009. Este software provee mecanismos de intercambio de datos con las principales aplicaciones de gestión académica, posee un potente motor de generación de horarios que obtiene resultados optimizados a costa de un consumo de tiempo elevado durante el proceso de generación de los horarios. Como aspectos negativos podemos mencionar que se distribuye bajo licencia propietaria y en su versión gratuita sólo permite modelar centros académicos con menos de 10 profesores, sólo corre sobre plataforma Windows y es poco flexible para modelar organizaciones con una estructura diferente a la preconcebida por sus autores. (Peñalara, 2009)

**KRONOWIN** es un generador de horarios escolares para Windows producido también en España que se encuentra en su quinta versión estable. Su mayor desventaja es el alto precio de la licencia para su uso y que solo se ejecuta sobre plataforma Windows.

Uno de los generadores de horarios docentes más populares internacionalmente es **Untis**, de la compañía suiza Untis Software, presenta versión "multiusuario" permitiendo que varios usuarios trabajen simultáneamente con el programa, al igual que los anteriores sólo corre bajo plataforma Windows. Este software proporciona varias soluciones parciales y específicas para centros

educacionales típicos de la región de procedencia y basados en algunos estándares relacionados con el sistema educacional. (Untis, 2009)

**Timetab** es un producto guatemalteco producido por la empresa de igual nombre que permite generar horarios de manera automática rápidamente, su versión ligera y gratuita es aplicable sólo centros pequeños con 5 grupos docentes y corre sólo sobre plataforma Windows. (Timetab, 2007)

**EAH (Elaborador Automático de Horarios)** es un *sistema experto* que reproduce el conocimiento adquirido tras años de elaborar los horarios de la Escuela Superior de Informática de la Universidad Europea de Madrid (UEM). El conjunto de restricciones que impone la filosofía de la Universidad introduce complejas limitaciones y conflictos entre asignaturas. Mejora los mecanismos genéricos de elaboración de horarios por cuanto que permite la introducción de restricciones concretas que determinan la forma del horario resultante. Los resultados obtenidos cuentan con la aprobación de un experto y se espera obtener en el futuro cercano una versión revisada para ser utilizada en la práctica.

Todos los sistemas existentes en el ámbito internacional por regla general no se adaptan a las particularidades del sistema educacional de nuestro país y en particular de nuestra universidad, además de poseer altos costos para la obtención de sus licencias.

En nuestra universidad se han realizado varias investigaciones en este campo. En los años 2006 y 2007 se desarrollaron sendos trabajos de diploma titulados **"Sistema para la gestión de horarios docentes"** y **"Asistente de ayuda para la confección de horarios docentes"** respectivamente. Estos trabajos no concluyeron con la realización de un software funcional limitándose al análisis y el diseño.

En el año 2008 en la Universidad de las Ciencias Informáticas se elaboraron otras investigaciones en este campo en las facultades 3 y 4, que tampoco concluyeron con la realización de un software. Los títulos de estos trabajos **son "Análisis y Diseño del Sistema de Ayuda para la planificación docente en la facultad 3"** y **"Análisis y Diseño de un sistema para la planificación automatizada del Horario Docente de la facultad 4".**

En este propio año en la facultad 7 se creó un sistema funcional que permite la obtención de horarios docentes de manera automática aunque concebido para el ambiente específico de esta facultad.

Los sistemas informáticos previamente desarrollados en nuestra universidad han sido concebidos para escenarios poco flexibles lo cual dificulta su uso en ambientes fuera de esta universidad donde la lógica de los procesos puede diferir notablemente.

El sistema que se propone permitirá gestionar toda la información referente a los horarios docentes de la facultad 5 de la Universidad de las Ciencias Informáticas y contará con la flexibilidad adecuada para ser aplicable a las demás facultades que conforman esta alta casa de estudios así como de otros centros educacionales del país. El fin principal del presente sistema difiere un tanto de los existentes al no proponerse la meta de sustituir al hombre sino de ayudarlo en los procesos de creación y gestión de horarios docentes, evitando inconsistencias y errores que a menudo pasan desapercibidos.

### 1.4 Tendencias y tecnologías actuales

El entorno donde se utilizará la presente propuesta de solución es un factor importante a tener en cuenta a la hora de seleccionar las mejores opciones para su desarrollo y puesta en funcionamiento. A continuación mostramos el estudio realizado para seleccionar las tecnologías y herramientas que serán utilizadas.

#### 1.4.1 Metodologías de desarrollo de software

Muchas veces el proceso de desarrollo de software resulta riesgoso y se convierte en una tarea difícil hallar el modo de controlar su curso de principio a fin. El problema principal radica en cómo coordinar todas las actividades que comprende el desarrollo de un proyecto de software, sobre todo si se trata de un proyecto de gran envergadura. De modo que se torna imprescindible contar con una forma organizada y adecuadamente estructurada para trabajar. Se necesita un proceso que integre las múltiples fases del desarrollo, un método común, un proceso que:

Proporcione una guía para ordenar las actividades de un equipo.

Dirija las tareas de cada desarrollador por separado y del equipo como un todo.

Especifique los artefactos que deben desarrollarse.

Ofrezca criterios para el control y la medición de los productos y actividades del proyecto.

##### 1.4.1.1 Metodologías tradicionales

Las metodologías tradicionales son aquellas que aplican procesos estrictos en el desarrollo de software. Son adecuadas para empresas grandes y productos complejos. Son procesos con un alto grado de burocracia que tienden a tener definidos todos los aspectos del desarrollo. La metodología

tradicional de más aceptación se conoce como Proceso Unificado de Desarrollo o Proceso Unificado de Rational. (Sánchez, 2004)

#### 1.4.1.1.1 Proceso Unificado de Rational

El Proceso Unificado de Rational (Rational Unified Process, por sus siglas *RUP*) es un proceso de desarrollo de software, cuyos modelos y artefactos se expresan en el Lenguaje Unificado de Modelado. *RUP* es una metodología robusta que puede ser adaptada a proyectos de mayor o menor complejidad, aplicable a diferentes esferas y ajustable a las necesidades de cada organización. Se trata de un proceso iterativo e incremental debido a que se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes. Se caracteriza por ser centrado en la arquitectura porque establece refinamientos sucesivos de una arquitectura ejecutable, construida como un modelo evolutivo de manera que no se afecte de forma significativa ante posibles modificaciones, para lograr finalmente una arquitectura comprensible, adaptable y robusta. Por último, *RUP* está dirigido por los casos de uso, pues guía el desarrollo del proyecto manteniendo como un aspecto de vital importancia la satisfacción del usuario y no sólo teniendo en cuenta las funcionalidades del sistema sino que permite controlar el proceso de desarrollo del proyecto al mismo tiempo que es elaborado, quedando conformada, a su vez, una guía para posteriores mejoras del producto.

En *RUP* se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

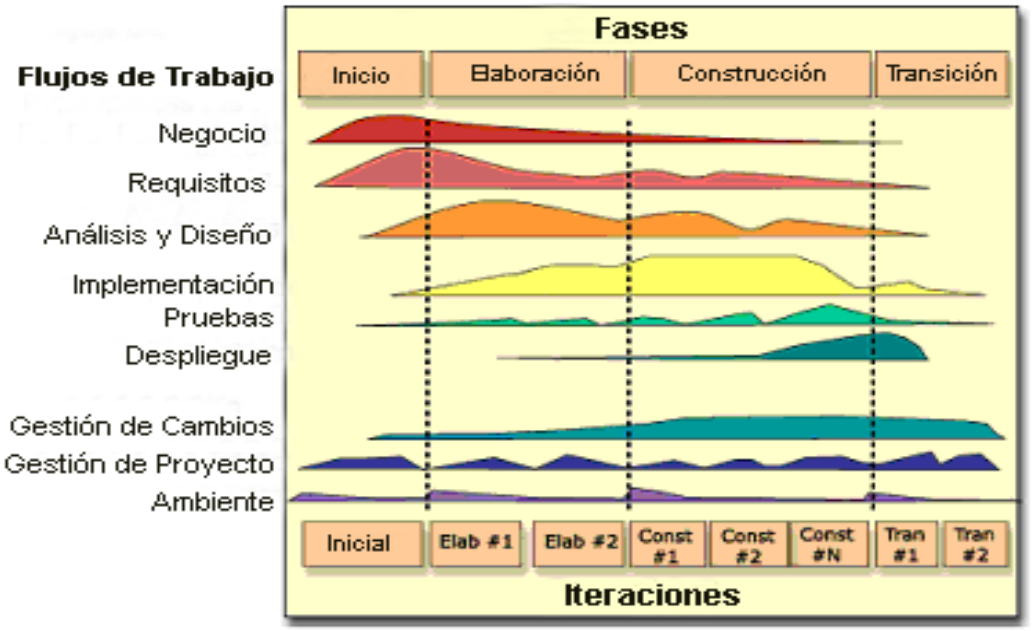


Figura 1.1 Fases y flujos de trabajo de RUP

**Flujos de trabajo que propone RUP:**

- **Modelación del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requisitos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán, la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Pruebas:** Busca los defectos a lo largo del ciclo de vida.
- **Instalación:** Produce un entregable del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

**Fases de RUP:**

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo con el alcance definido.

- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

- **Transición:** El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores. (Jacobson, 2000)

#### 1.4.1.2 Metodologías ágiles

Las metodologías ágiles, al contrario de las tradicionales, proponen una guía para acelerar el tiempo de desarrollo de los proyectos sin afectar la calidad de los mismos. Las Metodologías Ágiles o "ligeras" constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración. (Hernán, 2004)

Actualmente este tipo de metodología está ganando mucho auge en todo el mundo y se consideró prudente hacer un estudio de ellas, sus características, ventajas y desventajas.

##### 1.4.1.2.1 Programación Extrema

La programación extrema (*E**X**treme **P**rogramming*, por sus siglas *XP*) es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y equipo de desarrollo pequeño. Esta es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define especialmente para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

**Principalmente esta metodología se caracteriza por:**

1. Pruebas Unitarias**:** Se basa en las pruebas realizadas a los principales procesos, de manera que se prevén las fallas antes de que ocurran.

2. Refabricación**:** Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

3. Programación en pares**:** Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen de un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa. (Sánchez, 2004)

**Lo fundamental en este tipo de metodología es:**

1. La comunicación, entre los usuarios y los desarrolladores.
2. La simplicidad, al desarrollar y codificar los módulos del sistema.

La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales para así impedir la aparición de riesgos críticos que perjudiquen la realización con éxito del proyecto y simultáneamente, ayuda en gran medida a disminuir el tiempo de desarrollo y sus costos.

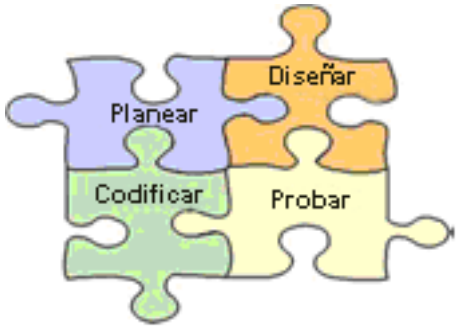



Figura 1.2. Fases de XP

**1.4.1.2.2 SCRUM**

Dentro de las metodologías ágiles se encuentra SCRUM. Esta metodología se enfoca fundamentalmente hacia la planeación iterativa y el seguimiento del proceso. El nombre “scrum” proviene de un término deportivo con el que sus creadores encontraron analogías y significa

cooperación para cumplir una meta en forma de equipo. Sus características son muy cercanas a las de XP, sin embargo presenta procesos de iteración con un período de 30 días, denominadas carreras cortas.

La metodología SCRUM se divide de forma tal que cuando se vaya a hacer una primera carrera se debe definir la funcionalidad para la misma y luego se deja al equipo de desarrolladores para que haga entrega de la primera iteración. La actividad principal en esta primera corrida es la de estabilizar los requisitos. Para ello la gerencia se da a la tarea de reunir al equipo todos los días durante 15 minutos para discutir las tareas del próximo día, haciendo que no se pierda el objetivo perseguido durante la carrera: la comunicación entre todos. Durante esta reunión se muestran todos los inconvenientes que impidan el progreso de la carrera y la gerencia debe resolver los mismos. Al igual la gerencia debe tener toda la información diaria que se genera como una forma de mantener la actualización diaria del proyecto. (Fowler, 2003)

#### 1.4.1.2.3 ¿Por qué se utilizará XP?

XP es una de las variantes de las metodologías ágiles con más aceptación en la comunidad de desarrollo, se utilizará para guiar todo el proceso de desarrollo del presente trabajo fundamentalmente por adaptarse perfectamente al tamaño del equipo de desarrollo. Otro factor de peso que se tuvo en cuenta a la hora de seleccionar esta metodología es su gran agilidad con lo que podremos obtener resultados tangibles y de utilidad para el usuario final en un menor plazo de tiempo.

### 1.4.2 Aplicaciones Web

Con la introducción de Internet y de la web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso a la información desde casi cualquier sitio. Esto representa un desafío a los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar la web.

Una aplicación web es una aplicación informática que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet. Las aplicaciones web son populares debido a la comodidad del uso de un navegador web como cliente ligero. La habilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. Aplicaciones como los Webmails, wikis, Weblogs, MMORPGs y tiendas en línea son ejemplos bien conocidos de aplicaciones web. (Rodríguez, 2003)

### 1.4.3 Plataformas de desarrollo y lenguajes de programación del lado del servidor

Cualquiera que sea la aplicación, una plataforma de desarrollo es un conjunto de clases construidas para un desarrollo rápido y sencillo ofreciendo herramientas para tareas recurrentes y aburridas.

A continuación se exponen algunas de las plataformas más populares para el desarrollo de aplicaciones Web, así como el lenguaje o los lenguajes que soporta.

#### 1.4.3.1 ASP.NET

ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos y empresariales, aplicaciones Web y servicios XML. Forma parte de la plataforma .NET y es la tecnología sucesora de Active Server Pages (ASP). ASP.NET es una nueva estructura de programación y fue diseñado con el objetivo de que dichas aplicaciones respondan rápidamente a las solicitudes de los usuarios, sin importar la cantidad de datos que se estén procesando en el servidor debido a que en una página ASP podemos incluir casi todo: HTML plano, código interpretado por los navegadores Web y texto, no hay una distinción formal entre el contenido de una página y su comportamiento. ASP.NET impone un cierto orden sobre el modelo de programación estándar ASP, proporciona diversas mejoras en las cuales se destacan:

**Rendimiento:** se compila desde el código nativo, lo que permite mucho mejor rendimiento y un almacenamiento de la caché en el servidor.

**Rapidez en programación:** mediante diversos controles, podemos con unas pocas líneas y en cuestión de minutos mostrar toda una base de datos y hacer rutinas complejas.

**Servicios Web:** trae herramientas para compartir datos e información entre distintos sitios.

**Seguridad:** tiene diversas herramientas que garantizan la seguridad de nuestras aplicaciones. (Programación en Castellano, 2005)

Esta plataforma permite usar dos lenguajes de programación diferentes para la lógica del servidor, estos son Visual C#.Net y Visual Basic.Net, ambos con igual potencia y posibilidades, aunque no permite usar ambos a la vez.

La principal desventaja de esta plataforma es que su desarrollo no está sustentado por la comunidad sino por un productor comercial (Microsoft Corporation) y esto significa una dependencia tecnológica importante.

#### 1.4.3.2 Java

Java es toda una tecnología orientada al desarrollo de software con el cual podemos realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. Pero Java no se queda ahí, ya que en la industria para dispositivos móviles también hay una gran acogida para este lenguaje.

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java (Java Virtual Machine). Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que el mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje orientado a objetos.

Java es un lenguaje de programación con el que se pude realizar múltiples tipos de programas. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como teléfonos móviles, agendas personales y automóviles.

Una de las desventajas de Java es la falta de estandarización. Existen versiones diferentes de su máquina virtual, siendo las más populares las desarrollas por Microsoft Corporation y Sun Microsystems. (Ciberaula, 2006)

#### 1.4.3.3 Ruby on Rails

Ruby on Rails, también conocido simplemente como Rails es una plataforma de desarrollo de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, y sigue el paradigma del patrón Modelo-Vista-Controlador *(MVC)*. Combina la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo poco código y con un mínimo de configuración. Rails se distribuye a través de RubyGems, conocidas como Gemas de Ruby que es el formato oficial de paquete y canal de distribución de librerías y aplicaciones Ruby.

Entre las desventajas del uso de esta plataforma podemos citar la falta de familiarización del equipo de desarrollo con el lenguaje Ruby y en sentido general con la plataforma Rails, sin embargo se considera una opción muy ventajosa pues una vez dominadas sus técnicas de programación permite un rápido desarrollo de aplicaciones web complejas. (Fowler, 2006)

#### 1.4.3.4 Plataforma Symfony

Symfony es una plataforma de desarrollo Web escrita en PHP 5 y para usar PHP 5, su concepción proviene de la empresa francesa Sensio y es adecuada para sitios Web profesionales, con necesidades complejas y donde se requiera mejorar los procesos y las prácticas del diseño y el desarrollo.

Está basada en librerías de código abierto como Mojavi, Propel y PRADO i18N.

Hereda conceptos de otras plataformas como el enrutamiento, la realización de pruebas funcionales y unitarias de manera automática, el uso YML y los ayudantes de Rails.

Además adiciona nuevos conceptos como la barra de depuración Web, el generador de los módulos de administración y la configuración en cascada. (Potencier, 2008)

#### 1.4.3.5 ¿Por qué se utilizará Symfony?

**Soporte:** Sigue la política de *LTS* (*Long Term Support*). Las versiones estables se mantienen durante 3 años pero con una continua corrección de los errores conocidos.

**Licencia:** Utiliza una licencia MIT con la que se pueden desarrollar aplicaciones web comerciales, gratuitas y/o de software libre.

**Compromiso:** La empresa que ha creado Symfony no vive de la plataforma, sino de las aplicaciones que hace con ella. Esto significa que a ellos les interesa tanto como a nosotros aspectos relacionados con el rendimiento, la buena documentación y el soporte a largo plazo.

**Código:** Desde su primera versión ha sido creado para PHP 5, desechando la versión 4 que ha sido declarada obsoleta.

**Seguridad:** Se puede controlar hasta el último acceso a la información e incluye por defecto protección contra los ataques informáticos más conocidos.

**Documentación:** Está muy bien documentado, existen miles de páginas en la Wiki oficial, tutoriales de hasta 250 páginas y un libro gratuito de casi 500 páginas el cual está traducido al español en su totalidad.

**Calidad:** Su código fuente incluye más de 8.000 pruebas unitarias y funcionales.

**Internacionalización:** Se pueden crear aplicaciones en varios idiomas. La internacionalización está integrada en la plataforma, funciona bien, sigue los estándares, es muy completa y está probada en aplicaciones reales.

**Conocimiento previo:** La experiencia del autor del trabajo y sus tutores en el uso de esta plataforma y en sentido general con el lenguaje PHP es relativamente alta comparada con otras tecnologías similares lo cual hace que el desarrollo sea mucho más rápido evitando la perdida de tiempo valioso en el aprendizaje y la familiarización con otras plataformas.

**Actualidad:** Symfony utiliza los conceptos más avanzados hasta la fecha para el desarrollo web, aúna las mejores prácticas de todo el mundo para agilizar el proceso de creación de sitios web complejos. Entre estos conceptos podemos citar al uso de mapeo objeto-relacional y del patrón *Modelo-Vista-Controlador.* (Potencier, 2008)

#### 1.4.3.6 ¿Qué es el patrón Modelo-Vista-Controlador?

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

**Modelo**: Ésta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

**Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

**Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

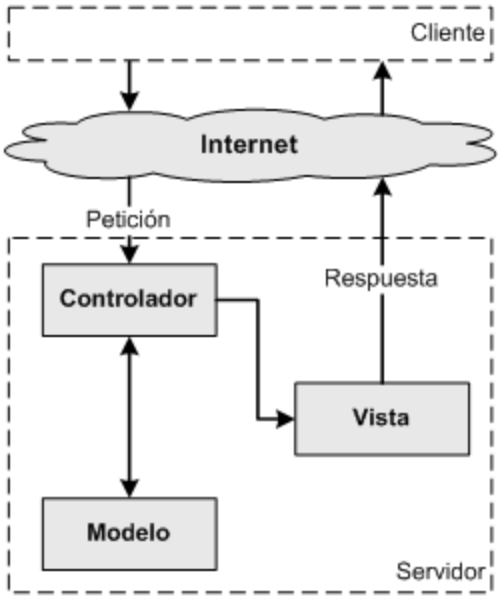



Figura 1.3 Patrón Modelo-Vista-Controlador

### 1.4.4 Servidores Web

Un servidor Web no es más que un programa que se ejecuta continuamente en una computadora (También llamada por lo general servidor) que interpreta las peticiones *HTTP* que recibe por parte de un cliente (un navegador Web) y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor, devolviendo algún tipo de resultado *HTML* al cliente o navegador que realizó la petición. (Rodríguez, 2003)

#### 1.4.4.1 Internet Information Server

Es la solución de Microsoft a las necesidades de las empresas y usuarios de enviar y recibir la información no sólo de sus clientes si no también en el entorno de la empresa. Este servidor web engloba una serie de herramientas administrativas que le permitirán controlar sitios web, *FTP*, *SMTP* (correo saliente) y servicio de noticias. Dispone también del soporte necesario para crear páginas dinámicas (*ASP* y *ASP.NET*), lenguaje de aplicaciones para Internet bastante extendido permitiendo la conexión y acceso a bases de datos, consiguiendo realizar aplicaciones web dinámicas y escalables.

#### 1.4.4.2 Apache

Apache es un servidor web que corre sobre los sistemas operativos más utilizados (*Unix, Linux, Windows*), lo que lo hace prácticamente universal. Es gratuito y su código es abierto, esto lo hace transparente de manera que puede ser estudiado, sin ningún secreto o puerta trasera que nos sorprenda. Es un servidor altamente configurable de diseño modular. Actualmente existe gran cantidad de módulos para Apache que son adaptables a este, y pueden ser instalados cuando los necesitemos. Con una cierta experiencia en programación en *Perl* o *C* se pueden construir módulos personalizados para realizar funciones específicas. Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Es altamente configurable en la creación y gestión de *logs*. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (Kabir, 1999)

En nuestros días, a pesar de que existen cierto número de servidores Web, el mercado de este tipo de software está prácticamente dominado por Apache. Se estima según estudios realizados en la red mundial que alcanzó su máxima cota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios Web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Netcraft, 2009)

#### 1.4.4.3 ¿Por qué se utilizará Apache?

Apache es sin lugar a dudas el servidor web más popular del mundo, su popularidad es el efecto de su alta calidad en todos los aspectos. Apache 2, su más reciente versión estable, posee todas las características deseables para este y para la mayoría de los sistemas basados en tecnología web, como son seguridad, estabilidad, facilidad de configuración, alto rendimiento y escalabilidad. Además de todo esto es multiplataforma, de código abierto y altamente integrado con *PHP*.

### 1.4.5 Sistemas de Gestión de Bases de Datos

Los Sistemas de Gestión de Bases de Datos (*SGBD*) son un tipo de software que sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan y se compone de los siguientes elementos:

Un lenguaje de definición de datos.

Un lenguaje de manipulación de datos.

Un lenguaje de consulta.

Un Sistema Gestor de Base de Datos (*SGBD*) es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Por tanto debe permitir:

1. Definir una Base de Datos: especificar tipos, estructuras y restricciones de datos.
2. Construir la Base de Datos: guardar los datos en algún medio controlado por el mismo *SGBD*.
3. Manipular la Base de Datos: realizar consultas, actualizarla, y generar informes.
4. Redundancia mínima. Un diseño correcto de una Base de Datos debe evitar la duplicación o redundancia de la información.
5. Independencia. Debe ser posible modificar la Base de Datos sin necesidad de realizar cambios en las aplicaciones que la accedan.

#### 1.4.5.1 Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de de datos relacionales que utiliza el lenguaje *Transact-SQL* y es capaz de poner a disposición de muchos usuarios un gran volumen de información. Entre sus ventajas se pueden mencionar su seguridad, su estabilidad y su escalabilidad. Soporta transacciones, procedimientos almacenados y disparadores (triggers). Posee un potente entorno gráfico para su administración y se integra muy bien con Microsoft Visual Studio. En sus últimas versiones se puede encontrar una característica que lo hace aun más potente y es su capacidad para soportar la programación en lenguajes de la plataforma .Net como son *C#* y *Visual Basic*.

Como desventajas cabe destacar que Microsoft SQL Server, al contrario de su más cercana competencia, no es multiplataforma, ya que sólo está disponible en sistemas operativos Windows.

(Microsoft Corporation, 2008)

#### 1.4.5.2 Oracle

Oracle es un sistema de base de datos relacional extremadamente potente y flexible. Esta potencia y flexibilidad, sin embargo, implican también una cierta complejidad. Para poder diseñar aplicaciones útiles basadas en Oracle es necesario entender como manipula los datos almacenados en el sistema. *PL/SQL* es una herramienta de gran importancia diseñada para la manipulación de datos, tanto internamente dentro de Oracle como externamente, en las propias aplicaciones. *PL/SQL* está

disponible en diversos entornos, cada uno de los cuales tiene diferentes ventajas. Es una aplicación propietaria y sus precios son muy altos en el mercado. (Colectivo de autores, 2008)

#### 1.4.5.3 PostgreSql

PostgreSQL es un sistema gestor de bases de datos relacional basada en objetos de código abierto. Esta base de datos comienza su andadura en 1986, como un sucesor de Ingres y en los últimos años se ha ido desarrollando gracias a un equipo de voluntarios a través de Internet. Actualmente es totalmente compatible con el lenguaje estándar de consultas (*ANSI SQL 92*), incluyendo selección anidada (subselects) y llaves foráneas (foreign keys).

En la versión 7.0 de esta base datos existe una limitación en el tamaño de las filas de una tabla que puede variar entre los 8 y los 32KB (esta última opción penaliza el rendimiento), siendo ésta una limitación importante dependiendo del tipo de datos que se deseen almacenar en la base de datos. Esta restricción ha sido eliminada en la versión 7.1, permitiendo un tamaño de filas prácticamente ilimitado.

Hay que hacer notar que el consumo de recursos por parte de PostgreSQL es muy elevado y sobrecarga el sistema, si se compara con un sistema mucho más sencillo como es MySQL, pero en cambio y debido a su arquitectura de diseño, escala muy bien al aumentar el número de *CPUs* y la memoria *RAM* disponible.

Quizá el principal problema que se le achacaba a PostgreSQL es que tradicionalmente ha sido lento, pero esto ha cambiado con la aparición de la versión 7.0 y, sobre todo, de la versión, 7.1, las cuales demuestran en los *benchmarks* realizados que actualmente PostgreSQL es, al menos, tan rápido como MySQL o InterBase, e incluso se puede comparar con las bases de datos de código propietario. (Espinoza, 2005)

#### 1.4.5.4 MySql

MySQL es un sistema gestor de bases de datos relacionales multihilo y multiusuario, cuyo principal objetivo de diseño fue la velocidad (ofrece alto rendimiento debido a su rapidez de respuesta). Sus creadores sacrificaron algunas características, como verificación de integridad y uso de disparadores, a cambio de ganar en velocidad. Provee múltiples motores de almacenamiento. Posibilita conexiones entre diferentes computadoras con distintos sistemas operativos y una integración perfecta con PHP. Este gestor de bases de datos es muy usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Su gran aceptación es debida, en parte, a que existe un gran número de librerías y

otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. Entre las características disponibles en las últimas versiones se pueden destacar las que se muestran a continuación:

1. Amplio subconjunto del lenguaje SQL.
2. Disponibilidad en gran cantidad de plataformas y sistemas (Windows NT, 2000, 2003 y XP, Mac OS X, Linux, etc.).
3. Varias opciones de almacenamiento, teniendo en cuenta si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
4. Transacciones y claves foráneas.
5. Conectividad segura.
6. Replicación.
7. Búsqueda e indexación de campos de texto.

**1.4.5.5 ¿Por qué se utilizará MySQL?**

Para seleccionar el Sistema Gestor de Bases de Datos más indicado para usar en nuestra propuesta de solución se tuvo en cuenta que perteneciera al movimiento de software libre, lo cual limita el marco de elección entre MySQL y PostreSql, los más populares y mundialmente probados en este campo.

Por las características específicas del sistema a implementar se determinó que resultaría óptimo el uso de MySQL para la gestión de datos del proyecto, debido a la poca complejidad de los procesos a realizar en la base de datos y la necesidad de rapidez al ejecutar las operaciones sobre los datos. Otro aspecto importante que se tuvo en cuenta es la facilidad de uso e instalación donde MySQL es líder indiscutible al venir incorporado en los paquetes más usados para *hosting* de sitios web profesionales, entiéndase WAMP, XAMPP y AppServ, lo que lo hace ideal para usuarios con poca experiencia en la instalación y configuración de Sistemas Gestores de Bases de Datos.

El uso de PostgreSQL no se descarta por lo que el producto final tendrá incorporado una utilidad de instalación que permita la selección por parte del usuario entre estos dos *SGBD,* para optar por el que le resulte más conveniente.

### 1.4.6 Lenguajes de programación y tecnologías del lado del cliente

En la arquitectura cliente-servidor, y específicamente en el argot de la Web, se denomina cliente al proceso o programa que hace pedidos *HTTP* al servidor Web a través de una red y muestra los resultados obtenidos de manera entendible y agradable al usuario final. En la actualidad existen variadas opciones en el mercado de aplicaciones que se conocen como navegadores web y cuya función es precisamente comunicarse con los servidores web y mostrar a los usuarios el contenido de las páginas web. Las tecnologías que hacen posible este proceso han sufrido un largo periodo de evolución y su desarrollo avanza a ritmos acelerados.

A continuación se exponen las principales tecnologías y lenguajes que son usados hoy día para conformar las páginas web que se visualizan en los navegadores.

#### 1.4.6.1 El lenguaje de marcas de hipertexto y el lenguaje extensible de marcas de hipertexto

El lenguaje de marcas de hipertexto (Hypertext Markup Language, HTML) es un lenguaje utilizado normalmente en la World Wide Web. Es muy sencillo y permite describir textos de forma estructurada y agradable, con enlaces, los cuales permiten conectar dos elementos entre si o fuentes de información relacionadas. HTML es un sistema de etiquetas el cual está basado en especificar la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.) así como los diferentes efectos que se quieren dar (cursiva, negrita o un gráfico determinado), los mismos indican como debe verse la información.

El lenguaje extensible de marcación de hipertexto (XHMTL) ofrece la norma clásica para crear páginas Web, el lenguaje de marcación de Hipertexto (HTML), y la nueva norma para datos descriptivos, el Lenguaje de marcación extensible (XML, Extensible Markup Language). La versión XHTML 1.0, no es muy diferente de HTML 4.01, y no es difícil aprenderla. Sin embargo, XHTML sigue el conjunto de reglas impuestas por XML; por lo tanto, desde el punto de vista sintáctico, se necesita ser más cuidadoso y más correcto cuando se construyan páginas web con XHTML que cuando se hagan con HTML. (Colectivo de autores, 2007)

#### 1.4.6.2 Hojas de estilo en cascada

Las hojas de estilo en cascada (Cascading Style Sheets, *CSS*) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en *HTML* o *XML*. *CSS* se utiliza para dar estilo a documentos *HTML* y *XML*, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos *HTML* y *XML*. CSS permite a los desarrolladores web controlar el

estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la *CSS* afectará a todas las páginas vinculadas a esa *CSS* en las que aparezca ese elemento. Su uso permite además que un mismo contenido se muestre de manera totalmente diferente, lo cual es muy útil si pensamos que la web moderna está cada vez más llegando a dispositivos móviles que no pueden mostrar las páginas como un navegador tradicional. (Colectivo de autores, 2008)

#### 1.4.6.3 Javascript

Javascript es un lenguaje de programación interpretado fundamentalmente por los navegadores web en el contexto de una página web. Es guiado por eventos y dispone de mecanismos de herencia aunque no es totalmente orientado a objetos. Posee una alta compatibilidad con todos los navegadores web modernos a diferencia de otros lenguajes como VBScript que sólo se interpreta bien en Internet Explorer (navegador web de código propietario). El código Javascript permite definir interactividad y dinamismo a las páginas Web. El único recurso con que cuenta este lenguaje es el propio navegador. Es un lenguaje de programación bastante sencillo, potente y pensado para hacer las cosas con rapidez. (Colectivo de autores, 2008)

#### 1.4.6.4 Javascript asíncrono y XML (AJAX)

Ajax, acrónimo de Asynchronous Javascript And XML, es una técnica de desarrollo web. Ajax no es una tecnología en sí sino que engloba un grupo de éstas que trabajan conjuntamente. Mediante el uso de esta técnica se logra mantener la comunicación entre el navegador web y el servidor de manera asíncrona, es decir sin actualizar la página web. Ajax utiliza un objeto *XMLHttpRequest* para hacer llamadas al servidor web, las respuestas a estas llamadas son interpretadas con un lenguaje de scripting (generalmente Javascript). Aunque en sus nombre se específica el uso de XML como lenguaje de intercambio de datos, en realidad se puede usar cualquier formato para los datos que retorna el servidor, desde *HTML*, *JSON* hasta texto plano.

#### 1.4.6.5 Librerías de Javascript

Existen varias librerías escritas en Javascript, que agrupan múltiples conjuntos de funciones y objetos, pensadas para ahorrar tiempo a los desarrolladores a la hora de crear páginas Web interactivas, entre ellas podemos citar Scriptaculous, Prototype, Yahoo User Interface, Google Web Toolkit y Extjs. Todas estas poseen características más o menos similares y son un arma potente para desarrollar aplicaciones Web de alto contenido estético, máxima funcionalidad y uso de *AJAX*, entre otras ventajas, todo esto escribiendo relativamente poco código.

Entre todas estas librerías se escogió a Extjs como la indicada para utilizarse en el desarrollo del sistema ante todo por su amplia documentación en Internet, que incluye extensos tutoriales, foros de discusión y múltiples ejemplos; además la riqueza de componentes gráficos reutilizables que posee, la experiencia acumulada por los estudiantes y profesores de nuestra universidad en el trabajo con esta plataforma y la existencia de herramientas web que permiten un rápido diseño de interfaces de usuario. (Extjs, 2009)

### 1.5 Otras herramientas a utilizar

#### 1.5.1 Notepad++ 4.0.2

Es un editor de texto multiuso que se distribuye bajo Licencia Pública General (GPL). Permite la creación y edición de archivos en formato de texto reconociendo múltiples extensiones y lenguajes de programación. Se ajusta automáticamente al tipo de archivo con que trabaja, señalando palabras clave, números de línea y caracteres de apertura y cierre, lo cual evita errores difíciles de detectar. Es ideal para el trabajo con lenguajes no compilados como *CSS, PHP*, *Javascript, XML* y *YML*. Es muy ligero y no sobrecarga al sistema aunque posea abiertos múltiples archivos en diferentes pestañas. Se le puede agregar funcionalidad mediante la instalación de complementos (*plugins*).

#### 1.5.2 Embarcadero ER/Studio 7.5

Esta es una herramienta de modelado visual de datos. Permite diseñar rápidamente bases de datos complejas, generando modelos Entidad-Relación, modelos físicos y *scripts* en SQL para la mayoría de los Sistemas de Gestión de Bases de Datos en el mercado. Además permite hacer ingeniería inversa desde bases de datos existentes y esquemas en *XML. Su* interfaz de usuario es amigable y cómoda. Como desventajas se pueden citar que solo funciona sobre plataforma Windows y su licencia es de tipo comercial.

#### 1.5.3 Phpmyadmin 2.11.2.1

Es el cliente web por excelencia de MySQL, incluido en las plataformas *WAMP*, *XAMPP* y *AppServ*. No requiere más instalación que copiarlo a un directorio donde el servidor web pueda acceder a sus archivos fuente. Es sencillo, intuitivo y ligero. Además permite la exportación de bases de datos completas a archivos *SQL, XML, YML, PDF*, entre otras; así como la importación de archivos SQL comprimidos con *gzip.* El hecho de ser una aplicación web escrita en PHP lo convierte en multiplataforma.

#### 1.5.4 Firebug 1.3.3

Es una extensión del popular navegador web Mozilla Firefox 3 que permite detectar errores en archivos *CSS* y *Javascript*, además de proveer una consola donde se muestran las llamadas

asincrónicas al servidor web, sus parámetros y su valor de retorno. Es muy útil para la depuración de páginas web en sentido general y en especial para aquellas que hacen uso de tecnología *AJAX*.

### 1.6 Conclusiones

El auge de las nuevas tecnologías ha traído consigo que en este capítulo se fundamentaran las tecnologías más apropiadas para el desarrollo de la aplicación. Teniendo en cuenta las tendencias actuales y las características del sistema a desarrollar y del equipo de desarrollo, se propone usar la metodología Extreme Programming para guiar todo el proceso. Para la construcción de la base de datos se usará MySQL 5.0.45. y para el modelado físico de la base de datos se utiliza la herramienta web Phpmyadmin 2.11.2.1. La programación del lado del servidor se realizará con PHP 5.0 usando la plataforma Symfony 1.2.4 y del lado del cliente se realizará usando la librería de Javascript: Extjs 2.2.1. Como editor de texto multipropósito se utilizará Notepad++ 4.0.2, para diseñar la base de datos se utilizará la herramienta ER/Studio 7.5 y para la depuración se utilizará la extensión de Mozilla Firefox 3: Firebug 1.3.3.

El avance de los medios enfocados a la Web se encuentra en un desarrollo vertiginoso y acelerado lo que permitió que se haya podido seleccionar de una amplia gama de productos los más adaptables al sistema que se pretende desarrollar.

# CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN

## 2.1 Introducción

En el presente capítulo se abordan las fases de exploración y planificación pertenecientes a la metodología XP (*Extreme Programming*) utilizada para el desarrollo del sistema que se propone. Además se exponen los artefactos generados durante el transcurso de dichas fases.

## 2.2 Fase de Exploración

La metodología de desarrollo XP comienza con la fase de exploración. Durante esta se realiza el proceso de identificación de las historias de usuario, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto.

### 2.2.1 Historias de Usuario

Las historias de usuario son la forma en que se especifican en *XP (Extreme Programming*) los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo. Durante la fase de exploración se identificaron 35 historias de usuario, las cuales se relacionan a continuación:

1) Navegar por el sistema
2) Gestionar universidad
3) Gestionar facultad
4) Gestionar departamento
5) Gestionar carrera
6) Gestionar local
7) Editar preferencias de local
8) Gestionar grupo
9) Editar preferencias de grupo
10) Gestionar disciplina
11) Gestionar asignatura
12) Editar plan de clases

13) Gestionar profesor
14) Editar preferencias de profesor
15) Gestionar alumno ayudante
16) Editar preferencias de alumno ayudante
17) Gestionar distribución
18) Confeccionar horario
19) Gestionar año
20) Gestionar semana
21) Gestionar turno
22) Gestionar tipo de turno
23) Gestionar tipo de local
24) Gestionar conjunto
25) Gestionar elementos de conjunto
26) Gestionar restricción
27) Gestionar parámetros de restricción
28) Gestionar Instancia de Restricción
29) Gestionar planificador
30) Gestionar seguridad
31) Autenticar
32) Cerrar sesión
33) Consultar horarios generales
34) Consultar horarios específicos
35) Consultar ayuda

Las Historias de Usuario representan las funcionalidades que el cliente desea que estén presentes en el sistema por lo tanto todo el trabajo futuro debe girar en torno a satisfacer estas expectativas. Las Historias de Usuario son la primera y fundamental entrada para el proceso de desarrollo de software

basado en la metodología *XP.* La descripción general de estas Historias de Usuario se encuentra en el Anexo I.

### 2.2.2 Personas Relacionadas con el sistema

Se denomina persona relacionada con el sistema a aquella que interactúa e intercambia con el sistema y obtiene resultados de los procesos desarrollados en la aplicación. Además aquella que interactúa con la misma sin poder hacer uso de las secciones privilegiadas del sistema. Las personas relacionadas con el sistema se infieren de una correcta lectura de las historias de usuario. A continuación se relacionan las personas vinculadas con el sistema propuesto.

Tabla 2.1 Personas relacionadas con el sistema

| **Personas relacionadas con el sistema** | **Justificación** |
|---|---|
| Usuario | Es la persona que navega por el sistema sin usar una cuenta específica y por tanto sin privilegios. Tiene la posibilidad de consultar los horarios publicados tanto de los diferentes grupos docentes como de los locales. |
| Usuario Autenticado | Es la persona que ya ha sido autenticada dentro de la aplicación y posee privilegios bajos, puede tratarse de un profesor o un alumno ayudante por lo que podrá consultar sus horarios específicos; así como acceder a los demás servicios que brinda la aplicación |
| Planificador | Es la persona encargada de llevar a cabo la administración de todo el contenido dentro de la aplicación, gestiona todo el sistema. |

### 2.2.3 Requisitos No Funcionales del Sistema

Los requisitos no funcionales del sistema no se capturan en la metodología XP en forma de historias de usuario, puesto que los clientes generalmente no conocen la terminología técnica que se utiliza para describir este tipo de requisitos.

La captura de requisitos no funcionales es por tanto una tarea del equipo de desarrollo completo, que haciendo un intercambio profundo de ideas con el cliente, determinan las mejores soluciones para crear un sistema que además de implementar correctamente las historias de usuario, cumpla con determinados estándares de calidad y con las características propias del negocio que se pretende automatizar.

#### 2.2.3.1 Requisitos de apariencia o interfaz externa

La aplicación propuesta será usada por personas que tengan un conocimiento medio de informática, por lo que la interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma en especial para aquellos que poseen pocas habilidades en el manejo de aplicaciones Web.

#### 2.2.3.2 Requisitos de usabilidad

A los administradores finales de la aplicación se les dará un adiestramiento básico en el uso de la aplicación. Estas personas tendrán un nivel de acceso amplio en la aplicación para que puedan dar respuesta a cada incidente ocurrido.

#### 2.2.3.3 Requisitos de rendimiento

Para un funcionamiento óptimo de la aplicación se seguirán las diferentes técnicas de elaboración de la Web, que faciliten el rápido acceso a sus páginas. La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo cliente/servidor, y la velocidad de la consultas de la base de datos. La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

#### 2.2.3.4 Requisitos de portabilidad

Las herramientas podrán ser usadas bajo cualquier sistema operativo de Windows NT en adelante o cualquier distribución de *Linux*. El servidor Web y el servidor de Base de Datos pueden estar en la misma computadora sin ocasionar problemas de rendimiento.

#### 2.2.3.5 Requisitos de seguridad

**Confiabilidad:** La información manejada por el sistema debe estar protegida de acceso no autorizado.

**Integridad:** La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.

**Disponibilidad:** La aplicación deberá estar disponible en todo momento para aquellas personas con acceso a la información y los mecanismos utilizados para lograr la seguridad no deben ser un obstáculo a los usuarios para obtener los datos deseados en un momento dado.

#### 2.2.3.6 Requisitos de software

En las computadoras de los usuarios solo se requiere un navegador Web moderno (Mozilla Firefox 2 o superior recomendado), pueden usar cualquier sistema operativo. En el servidor de base de datos se requiere de Windows NT en adelante o cualquier distribución de Linux y como gestor de base de datos MySQL 5.0 o PostgreSql 8.2. En el servidor Web se requiere Apache 2 o superior, PHP 5.2 con las extensiones *php_mysql*, *php_pgsql*, *php_pdo*, *php_pdo_mysql*, *php_pdo_pgsql* y *php_ldap* activadas.

#### 2.2.3.7 Requisitos de hardware

En el cliente se requiere una máquina con 128 MB de *RAM* como mínimo, el servidor Web junto con el servidor de base de datos debe tener 256 MB de *RAM* y 20 GB de disco duro mínimo y todas las máquinas implicadas en el funcionamiento de la aplicación deben estar conectadas a la red.

### 2.3 Fase de Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros del equipo de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, por ejemplo: las pruebas unitarias, la integración y refactorización del código y la preparación y ejecución de las pruebas de aceptación.

#### 2.3.1 Estimación de esfuerzo por Historia de Usuario

Para el desarrollo del sistema propuesto se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los resultados que se muestran en la Tabla 2.2.

Tabla 2.2 Estimación de esfuerzo por historia de usuario

| **Historia de Usuario** | **Puntos estimados** |
|---|---|
| Navegar por el sistema | 1 |
| Gestionar universidad | 0.5 |
| Gestionar facultad | 0.5 |
| Gestionar departamento | 0.5 |
| Gestionar carrera | 0.5 |
| Gestionar local | 0.5 |
| Editar preferencias de local | 0.5 |
| Gestionar grupo | 0.5 |
| Editar preferencias de grupo | 0.5 |
| Gestionar disciplina | 0.5 |
| Gestionar asignatura | 0.5 |
| Editar plan de clases | 0.5 |
| Gestionar profesor | 0.5 |
| Editar preferencias de profesor | 0.5 |
| Gestionar alumno ayudante | 0.5 |
| Editar preferencias de alumno ayudante | 0.5 |
| Gestionar distribución | 1 |
| Confeccionar horario | 2 |
| Gestionar año | 0.5 |
| Gestionar semana | 0.5 |

| | |
|---|---|
| Gestionar turno | 0.5 |
| Gestionar tipo de turno | 0.5 |
| Gestionar tipo de local | 0.5 |
| Gestionar conjunto | 0.5 |
| Gestionar elementos de conjunto | 0.5 |
| Gestionar restricción | 1 |
| Gestionar parámetros de restricción | 0.5 |
| Gestionar instancia de restricción | 0.5 |
| Gestionar planificador | 0.5 |
| Gestionar seguridad | 0.5 |
| Autenticar | 0.5 |
| Cerrar sesión | 0.5 |
| Consultar horarios generales | 0.5 |
| Consultar horarios específicos | 0.5 |
| Consultar ayuda | 1 |

### 2.3.2 Plan de iteraciones

Una se vez identificadas las historias de usuario del sistema y estimado el esfuerzo requerido para la implementación de cada una de estas, se procede a realizar el Plan de iteraciones del proyecto. Teniendo en cuenta el riesgo para desarrollar cada una de las historias de usuario, el tamaño del equipo de desarrollo, así como otros factores subjetivos se decidió dividir el proyecto en tres iteraciones, detalladas a continuación.

**Iteración 1**

En esta iteración se realizarán las historias de usuarios más sencillas, la realización de las mismas proporciona una idea de cómo funcionará la aplicación aunque todavía estará en sus inicios.

**Iteración 2**

En esta iteración se realizarán las historias de usuarios más complejas y que inciden críticamente en la lógica de la aplicación. Una vez concluida esta iteración el sistema se encontrará en un estado plenamente funcional.

**Iteración 3**

En esta iteración se realizarán las historias de usuarios que se consideren opcionales y que no inciden críticamente en la lógica de la aplicación. Una vez concluida esta iteración el sistema se encontrará completamente concluido y listo para su explotación.

### 2.3.3 Plan de duración de iteraciones

Como parte del ciclo de vida de un proyecto utilizando XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con que se cuenta. Este plan se encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementarán.

Tabla 2.3 Plan de duración de las iteraciones

| Iteración | Orden de las historias de usuarios | Puntos | Duración de las iteraciones |
|---|---|---|---|
| Iteración 1 | Navegar por el sistema | 1 | 8.5 semanas |
| | Gestionar universidad | 0.5 | |
| | Gestionar facultad | 0.5 | |
| | Gestionar departamento | 0.5 | |
| | Gestionar carrera | 0.5 | |
| | Gestionar tipo de local | 0.5 | |
| | Gestionar local | 0.5 | |
| | Gestionar año | 0.5 | |

| | | | |
|---|---|---|---|
| | Gestionar grupo | 0.5 | |
| | Gestionar disciplina | 0.5 | |
| | Gestionar asignatura | 0.5 | |
| | Gestionar profesor | 0.5 | |
| | Gestionar alumno ayudante | 0.5 | |
| | Gestionar semana | 0.5 | |
| | Gestionar tipo de turno | 0.5 | |
| | Gestionar turno | 0.5 | |
| Iteración 2 | Gestionar conjunto | 0.5 | 11.5 semanas |
| | Gestionar elementos de conjunto | 0.5 | |
| | Gestionar preferencias de local | 0.5 | |
| | Gestionar plan de clases | 0.5 | |
| | Gestionar preferencias de grupo | 0.5 | |
| | Gestionar preferencias de profesor | 0.5 | |
| | Gestionar preferencias de alumno ayudante | 0.5 | |
| | Gestionar distribución | 0.5 | |
| | Gestionar restricción | 1 | |

| | | | |
|---|---|---|---|
| | Gestionar parámetros de restricción | 0.5 | |
| | Gestionar instancias de restricción | 0.5 | |
| | Confeccionar horario | 2 | |
| | Gestionar planificador | 0.5 | |
| | Gestionar seguridad | 0.5 | |
| | Autenticar | 0.5 | |
| | Cerrar sesión | 0.5 | |
| | Consultar horarios generales | 0.5 | |
| | Consultar horarios específicos | 0.5 | |
| Iteración 3 | Consultar Ayuda. | 1.5 | 1.5 semanas |

### 2.3.4 Plan de entregas

A continuación se presenta el plan de entregas ideado para la fase de implementación. Como producto del mismo se harán *releases* del sistema al finalizar cada iteración en la fecha aproximada que se indica en la Tabla 2.4.

Tabla 2.4 Plan de entregas

| Sistema | Hito | Fecha Aproximada |
|---|---|---|
| Sistema de Gestión de Horarios Docentes versión 0.1 | Fin de la primera iteración | Primera semana de marzo de 2009 |
| Sistema de Gestión de Horarios Docentes versión 0.2 | Fin de la segunda iteración | Tercera semana de mayo de 2009 |
| Sistema de Gestión de Horarios | Fin de la tercera | Primera semana de junio de 2009. |

| | | |
|---|---|---|
| Docentes versión 1.0 | iteración | |

## 2.4 Conclusiones

En el presente capítulo se abordaron los temas referentes a las fases de exploración y planificación del proyecto basado en la metodología *XP*. Como resultado de la aplicación de las técnicas de ingeniería propuestas por esta metodología para estas fases se obtuvieron los artefactos necesarios para comenzar el diseño del sistema.

Una mirada exhaustiva a los artefactos aquí generados brinda el primer acercamiento al sistema a construir, aunque solo de una manera superficial. Esto no es una limitante en absoluto porque esta concebido que antes y durante del desarrollo del proyecto, el equipo de trabajo posea escaso o ningún conocimiento del negocio a modelar; a excepción del cliente quien debe conocer bien sus necesidades, fomentándose la comunicación para lograr los resultados esperados.

# CAPÍTULO 3: DISEÑO

## 3.1 Introducción

Para el diseño de las aplicaciones, la metodología *XP* no requiere la representación del sistema mediante diagramas de clases utilizando el Lenguaje Unificado de Modelado (UML), en su lugar se usan otras técnicas como las tarjetas Clase-Responsabilidad-Colaboración (CRC). No obstante el uso de diagramas UML puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante. Según XP un buen diseño es aquel que representa el problema a resolver, garantiza una correcta implementación y lo hace utilizando el menor número de clases y operaciones posibles.

En el presente capítulo se tratará la fase de diseño de la solución propuesta siguiendo las guías de la metodología XP.

## 3.2 Metáfora para el sistema

Una metáfora para el sistema es una historia que todo el mundo puede contar acerca de cómo el sistema funciona. (Beck, 1999) La tarea de elegir una metáfora para el sistema permite mantener la coherencia de nombres de todo aquello que se va a implementar. El nombre de los objetos o partes de nuestro sistema es muy importante. La tarea de “poner nombre”, sencilla a simple vista, no lo es tanto. Se debe elegir un sistema de nombres que permita que cualquiera que lo vea adivine la relación entre el objeto y aquello que representa.

En la Tabla 3.1 se muestra la metáfora para el sistema propuesto:

Tabla 3.1 Metáfora para el sistema

| Término | Significado | Clase |
|---|---|---|
| Alumno ayudante | Estudiante que imparte la docencia de una o varias asignaturas que previamente a cursado y aprobado. Es una subcategoría de docente. | Professor |
| Año | Estructura para agrupar los grupos, no es un año natural ni calendario. | Year |
| Asignatura | Unidad de enseñanza que forma parte del plan de estudios de cada carrera. | Lecture |

| | | |
|---|---|---|
| Carrera | Plan de estudios en la educación superior, se diferencian de acuerdo a las asignaturas que reciben. | Career |
| Conjunto | Unión de varios elementos de un mismo tipo. | Folder |
| Departamento | Estructura en la que se agrupan los docentes de acuerdo a las asignaturas que imparten. | Department |
| Disciplina | Estructura en la que se agrupan las asignaturas. | Discipline |
| Distribución | Unión lógica que relaciona un grupo, un docente, una asignatura y opcionalmente un tipo de turno. Responde a la pregunta: ¿Qué docente le imparte un determinado tipo de turno de una determinada asignatura a un determinado grupo? | Distribution |
| Docente | Persona que imparte la docencia. | |
| Estado de local | Estado de un local en términos de disponibilidad para desarrollar actividades docentes en un momento dado, que incluye semana, día y turno. | LocalPreference |
| Facultad | Facultad de una universidad. | Faculty |
| Grupo | Estructura en la que se agrupan los estudiantes para recibir la docencia. | Group |
| Horario | Unión lógica entre un grupo, un docente, un plan, un día, un turno y un local. Responde a las preguntas: ¿Qué actividad desarrolla un determinado docente o grupo en un momento determinado? ¿Dónde la realiza? | Schedule |

| | | |
|---|---|---|
| Instancia de restricción | Restricción con sus datos de entrada, lista para ser evaluada de acuerdo a su lógica, al valor de dichos datos y al estado del sistema. | RestrictionInstance |
| Local | Inmueble donde se imparte la docencia. | Local |
| Parámetro de Instancia | Valor que toman los parámetros de una restricción en una determinada instancia de restricción. | ParamenterInstance |
| Parámetro de restricción | Tipo de elemento que las restricciones tienen en consideración a la hora de evaluarse. | RestrictionParameter |
| Plan | Actividad programada para una asignatura. Es la unión lógica entre asignatura, tipo de turno, tipo de local y semana. Responde a la preguntas: ¿Qué actividades tiene programada una asignatura determinada para una semana determinada? ¿De qué tipo son éstas actividades? ¿En qué orden deben realizarse? ¿En qué tipo de local deben realizarse? | Planning |
| Planificador | Persona con permisos para gestionar toda la información del sistema. | Planificator |
| Preferencia de docente | Valor de felicidad de un grupo para un determinado momento, que incluye semana, día y turno; con respecto al hecho de impartir o recibir la docencia en dicho momento. | ProfessorPreference |
| Preferencia de grupo | Valor de felicidad de un grupo para un determinado momento, que incluye semana, día y turno; con respecto al hecho | GroupPreference |

| | | |
|---|---|---|
| | de impartir o recibir la docencia en dicho momento. | |
| Profesor | Persona que imparte la docencia de manera profesional, es una subcategoría de docente. | Professor |
| Restricción | Representa una particularidad del negocio que se modela en el sistema. Es una función matemática que en dependencia de los valores de sus datos de entrada, de su propia lógica y del estado del sistema; retorna un valor único. | Restriction |
| Semana | Semana de trabajo. | Week |
| Tipo de local | Diferentes categorías en las que se pueden clasificar los locales. | LocalType |
| Turno | Mínima división del día laboral. Es el tiempo en el que se desarrolla una actividad docente. | Turn |
| Universidad | Representa una universidad. | University |

### 3.3 Diseño de la base de datos

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda, el diseño la misma. El objetivo principal del diseño de bases de datos es generar tablas que modelan los registros en los que se guardará información. Un buen diseño de una base de datos evita redundancias en los datos almacenados y por consiguiente evita fallos de integridad, este aspecto es de vital importancia aunque en ocasiones, por motivos de velocidad, es conveniente tener alguna información redundante siempre y cuando su uso esté justificado.

En el diseño de la base de datos que usará el sistema propuesto se modelaron 40 tablas, considerándose una base de datos de mediana complejidad. Ver Anexo II.

### 3.4 Tarjetas Clase-Responsabilidad-Colaboración

Las tarjetas Cargo-Responsabilidad-Colaboración (por sus siglas, *CRC*) permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas *CRC* permiten que el equipo completo contribuya en la tarea del diseño. Una tarjeta *CRC* representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Al desarrollar aplicaciones web usando la plataforma *Symfony*, que hace uso del patrón Modelo-Vista-Controlador, el diseño usando tarjetas *CRC* se convierte en una tarea un tanto innecesaria una vez diseñada de manera correcta la base de datos. Debido a las comodidades que brinda *Symfony* para el manejo de bases de datos usando mapeo objeto-relacional (*ORM*), la mayoría de las clases del sistema pueden ser generadas automáticamente a partir de esquemas escritos en *YML* y a la vez estos esquemas pueden ser creados a partir de los metadatos contenidos en las bases de datos. Es decir que a partir de una base datos, *Symfony* crea toda una estructura de clases persistentes que en la mayoría de los casos son todas las clases necesarias para desarrollar el sistema completo. No obstante, algunas veces es necesario añadir nuevas clases de forma manual así como nuevas funcionalidades a las clases autogeneradas.

Durante la fase de diseño del sistema propuesto se prevé el uso de clases no persistentes y que por tanto se deben escribir completamente por el equipo de desarrollo. Las tarjetas *CRC* de estas clases se relacionan a continuación.

| **Generator** | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Obtener las opciones de clases para un grupo en un momento determinado | Group, Week, Turn, Distribution, Planning, Professor, Local |
| Obtener las opciones de clases para un docente en un momento determinado | Professor, Week, Turn, Distribution, Planning, Group, Local |
| Obtener las opciones de posibles locales para una actividad docente en un | Planning, Turn, Group, Local |

| | |
|---|---|
| momento determinado | |
| Autogenerar el horario de un grupo para una semana determinada | Group, Week, Turn, Lecture, Professor, Schedule, Evaluator |

| **Evaluator** | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Evaluar el código de una instancia de restricción | Schedule, Restriction, RestrictionInstance, Parameter Instance |

| **LDAP** | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Verificar las credenciales de un usuario del dominio | |
| Cambiar la contraseña de un usuario del dominio | |

Además se concibe agregar responsabilidades a las siguientes clases autogeneradas:

| **Local** | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Obtener disponibilidad de un local para un momento determinado | Week, Turn, LocalPreference |
| Modificar disponibilidad de un local para un momento determinado | Week, Turn, LocalPreference |

| | |
|---|---|
| Obtener información sobre el horario de un local para un momento determinado | Week, Turn, Schedule |
| Obtener el estado de ocupación de un local para un momento determinado | Week, Turn, LocalPreference, Schedule |

| **Group** | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Obtener la felicidad de un grupo para un momento determinado | Week, Turn, GroupPreference |
| Modificar la felicidad de un grupo para un momento determinado | Week, Turn, GroupPreference |
| Obtener información sobre el horario de un grupo para un momento determinado | Week, Turn, Schedule |
| Obtener el estado de ocupación de un grupo para un momento determinado | Week, Turn, GroupPreference, Schedule |
| Obtener las actividades que debe realizar un grupo para una semana determinada | Week, Distribution, Planning |
| Eliminar las actividades de un grupo para un momento determinado | Week, Turn, Schedule |
| Obtener la cantidad de actividades que debe realizar un grupo en una semana determinada que aun no han sido asignadas en el horario | Week, Schedule |

| Professor | |
|---|---|
| **Responsabilidades** | **Colaboraciones** |
| Obtener la felicidad de un profesor para un momento determinado | Week, Turn, ProfessorPreference |
| Modificar la felicidad de un profesor para un momento determinado | Week, Turn, ProfessorPreference |
| Obtener información sobre el horario de un profesor para un momento determinado | Week, Turn, Schedule |
| Obtener el estado de ocupación de un profesor para un momento determinado | Week, Turn, ProfessorPreference, Schedule |
| Eliminar las actividades de un profesor para un momento determinado | Week, Turn, Schedule |
| Obtener la cantidad de actividades que debe realizar un profesor en una semana determinada que aun no han sido asignadas en el horario | Week, Distribution, Group, Schedule |

### 3.5 Conclusiones

En el presente capítulo se llevó a cabo el diseño del sistema tomando como base las necesidades de los clientes reflejadas en las Historias de Usuario definidas en el capítulo anterior. Se procuró a toda costa realizar un diseño sencillo y general, pasando por alto los detalles más sutiles que serán resueltos en la fase de implementación.

Después de obtener los artefactos de esta fase y estructurar la base de datos que le dará soporte al sistema, el proyecto se encuentra listo para iniciar las fases de implementación y pruebas.

# CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

## 4.1 Introducción

En este capítulo se abordan las fases de Implementación y Pruebas de manera conjunta debido a que según la Metodología XP estas fases están indisolublemente ligadas. El proceso de pruebas comienza incluso antes del de implementación, ya que las pruebas unitarias deben ser escritas con anterioridad a la funcionalidad que se desea desarrollar.

Estas fases son el grueso de la metodología utilizada debido a que son las que generan el producto que utilizaran los clientes, por tanto esta etapa será la más compleja y llena de retos que el equipo debe enfrentar, donde se debe estimular la creatividad de los desarrolladores para resolver los problemas que se presenten.

## 4.2 Creación de la base de datos

El primer paso para comenzar a implementar el sistema es poner a punto la base de datos que le dará soporte. A partir del diseño de la base de datos (Ver Anexo II) se genera un *script* en lenguaje *SQL* acorde con el Sistema Gestor de Bases de Datos seleccionado; este *script*, al ser interpretado por una herramienta adecuada construye las tablas de la base de datos.

## 4.3 Generación de las clases del modelo

Las clases del modelo del sistema son generadas por *Symfony* de manera automática a partir del esquema de la base de datos. Los pasos para obtener estas clases son:

1) Generar el esquema a partir de la base de datos
2) Modificar el esquema para adaptar los nombres de las clases según convenga.
3) Generar el modelo a partir del esquema.

Los comandos usados para estas tareas se muestran en el Anexo III.

### 4.4 División estructural del sistema

Para dar cumplimiento a las exigencias de los clientes plasmadas en las historias de usuario, se divide el sistema en dos aplicaciones web:

- Planificador: es la aplicación con la que interactúan los usuarios con el rol de planificador.
- Usuario: es la aplicación con la que interactúan los usuarios autenticados para obtener información específica y los usuarios sin autenticar para obtener información general.

A su vez cada una de estas aplicaciones se divide en varios módulos desacoplados pero altamente cohesivos.

La aplicación Planificador está conformada por los módulos Años, Árbol, Asignaturas, Ayudantes, Carreras, Común, Conjuntos, Departamentos, Disciplinas, Facultades, Grupos, Instancias, Locales, Planificadores, Profesores, Restricciones, Seguridad, Semanas, Tipolocales, Tipoturnos, Turnos, Universidades y Vista. A continuación se relacionan de cada uno de estos módulos y sus funcionalidades.

**Módulo Años:**

- Listar todos los años
- Mostrar los datos de un año
- Editar los datos de un año
- Crear un nuevo año
- Eliminar un año

**Módulo Árbol:**

- Listar los elementos del sistema en forma de árbol de navegación

**Módulo Asignaturas:**

- Mostrar los datos de una asignatura
- Editar los datos de una asignatura
- Listar todas las asignaturas
- Crear una nueva asignatura
- Eliminar una asignatura

- Listar todas las asignaturas que puede recibir un grupo determinado
- Gestionar el plan de clases de una asignatura

**Módulo Ayudantes:**

- Mostrar los datos de un alumno ayudante
- Editar los datos de un alumno ayudante
- Listar todos los alumnos ayudantes
- Crear un nuevo alumno ayudante
- Eliminar un alumno ayudante
- Listar todos los alumnos ayudantes que pueden impartir clases a un grupo determinado
- Gestionar la distribución de clases de un alumno ayudante
- Gestionar el horario de un alumno ayudante

**Módulo Carreras:**

- Mostrar los datos de una carrera
- Editar los datos de una carrera
- Listar todas las carreras
- Crear una nueva carrera
- Eliminar una carrera

**Módulo Común:**

- Mostrar una interfaz para la autenticación de los usuarios
- Cambiar la contraseña del usuario autenticado
- Verificar los datos de autenticación de un usuario
- Asignar credenciales a un usuario
- Devolver las credenciales de un usuario
- Permitir cerrar la sesión al usuario autenticado

**Módulo Conjuntos:**

- Mostrar los datos de conjunto
- Listar todos los conjuntos de un determinado tipo
- Crear un nuevo conjunto de un tipo determinado
- Editar los datos de un conjunto
- Gestionar los elementos de un conjunto de acuerdo a su tipo
- Eliminar un conjunto

**Módulo Departamentos:**

- Mostrar los datos de un departamento
- Editar los datos de un departamento
- Listar todos los departamentos
- Crear un nuevo departamento
- Eliminar un departamento

**Módulo Disciplinas:**

- Mostrar los datos de una disciplina
- Editar los datos de una disciplina
- Listar todas las disciplinas
- Crear una nueva disciplina
- Eliminar una disciplina

**Módulo Facultades:**

- Mostrar los datos de una facultad
- Editar los datos de una facultad
- Listar todas las facultades
- Crear una nueva facultad
- Eliminar una facultad

**Módulo Grupos:**

- Mostrar los datos de un grupo
- Editar los datos de un grupo
- Crear un nuevo grupo
- Listar todos los grupos
- Eliminar un grupo
- Editar las preferencias de un grupo
- Gestionar la distribución de clases de un grupo
- Gestionar el horario de un grupo

**Módulo Instancias:**

- Crear una nueva instancia de una restricción
- Mostrar los valores de los parámetros de una instancia de una restricción
- Editar los valores de los parámetros de una instancia de una restricción
- Eliminar una instancia de una restricción

**Módulo Locales:**

- Mostrar los datos de un local
- Editar los datos de un local
- Listar todos los locales
- Crear un nuevo local
- Eliminar un local
- Editar la disponibilidad de un local
- Mostrar el horario de un local

**Módulo Planificadores:**

- Mostrar los datos de un planificador
- Editar los datos de un planificador
- Listar todos los planificadores
- Crear un nuevo planificador
- Eliminar un planificador
- Cambiar la contraseña de un planificador

**Módulo Profesores:**

- Mostrar los datos de un profesor
- Editar los datos de un profesor
- Listar todos los profesores
- Crear un nuevo profesor
- Eliminar un profesor
- Listar todos los profesores que pueden impartir clases a un grupo determinado
- Gestionar la distribución de clases de un profesor
- Gestionar el horario de un profesor
- Cambiar la contraseña de un profesor

**Módulo Restricciones:**

- Mostrar los datos de una restricción
- Crear una nueva restricción
- Eliminar una restricción
- Editar los datos de una restricción
- Mostrar el código fuente de una restricción
- Editar el código fuente de una restricción
- Probar el código fuente de una restricción
- Gestionar los parámetros de una restricción

**Módulo Seguridad:**

- Mostrar la configuración de seguridad del sistema
- Editar la configuración de seguridad del sistema
- Probar la configuración de seguridad del sistema

**Módulo Semanas:**

- Mostrar los datos de una semana
- Editar los datos de una semana
- Listar todas las semanas
- Crear una nueva semana
- Eliminar una semana

**Módulo Tipolocales:**

- Mostrar los datos de un tipo de local
- Editar los datos de un tipo de local
- Listar todos los tipos de local
- Crear un nuevo tipo de local
- Eliminar un tipo de local

**Módulo Tipoturnos:**

- Mostrar los datos de un tipo de turno
- Editar los datos de un tipo de turno
- Listar todos los tipos de turno
- Crear un nuevo tipo de turno
- Eliminar un tipo de turno

**Módulo Turnos:**

- Mostrar los datos de un  turno
- Editar los datos de un turno
- Listar todos los turnos
- Crear un nuevo turno
- Eliminar un turno

**Módulo Universidades:**

- Mostrar los datos de una universidad
- Editar los datos de una universidad
- Listar todas las universidades
- Crear un nueva universidad
- Eliminar una universidad

**Módulo Vista:**

- Mostrar la interfaz de usuario

La aplicación Usuario está conformada por los módulos Árbol, Grupos, Locales, Profesores, Semanas y Vista; además esta aplicación utiliza el modulo Común de la aplicación Planificador para las operaciones de seguridad. A continuación se muestran estos módulos y sus funcionalidades.

**Módulo Árbol:**

- Listar los elementos del sistema en forma de árbol de navegación de manera diferenciada para cada usuario

**Módulo Grupos:**

- Mostrar el horario de un grupo

**Módulo Locales**

- Mostrar el horario de un local

**Módulo Profesores**

- Mostar el horario de un profesor
- Mostar el horario de un alumno ayudante

**Módulo Semanas:**

- Listar todas las semanas Listar todos los conjuntos de semanas

**Módulo Vista:**

- Mostrar la interfaz de usuario

Una vez determinadas las acciones de cada módulo, es posible crear la estructura de todo el sistema (Ver Anexo IV).

### 4.5 Pruebas

Uno de los pilares fundamentales de *XP* es el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de éste en el sistema y su detección. Esto contribuye a elevar la calidad de los productos desarrollados y la seguridad de los programadores a la hora de introducir cambios o modificaciones. La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma

automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

### 4.5.1 Pruebas Unitarias

Escribir pruebas unitarias es una buena práctica de desarrollo web, aunque puede resultar difícil de poner en práctica. La metodología *XP* propone escribir estas pruebas antes de la implementación de la función a probar. *Symfony* es capaz de realizar estas pruebas de manera automática, haciendo uso de una utilidad llamada Lime, lo cual minimiza el tiempo empleado para realizar esta tarea.

Las pruebas unitarias son generalmente de caja negra, o sea que solo se observa el resultado obtenido de la llamada a las funciones a probar de acuerdo a los parámetros de entrada, si una función retorna al menos un valor no esperado se procede a su depuración paso a paso para detectar los errores que pueda contener.

Para escribir la pruebas unitarias usando Lime es conveniente estructurar correctamente los casos de prueba usando la información de la Tabla 4.1 para cada función a probar.

Tabla 4.1 Ejemplo de Caso de Prueba Unitaria

| **Clase:** Ldap | | |
|---|---|---|
| **Función:** CheckPass($username, $password) | | |
| **Valores de Entrada** | **Valores Esperados** | **Valores obtenidos** |
| Usuario, contraseña | true | true |
| usuario, contraseña_incorrecta | false | false |
| Usuario_incorrecto, contraseña | false | false |
| Usuario_incorrecto, contraseña_incorrecta | false | false |
| **Resultados de la prueba:** Prueba Satisfactoria | | |

### 4.5.2 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario. Durante las iteraciones las historias de usuarios seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable. Una historia de usuario no se considera completa hasta que no ha pasado por sus pruebas de aceptación.

Durante el desarrollo de las tres iteraciones del proyecto se realizaron las pruebas de aceptación para las 35 Historias de Usuario que conforman el sistema. Todas las pruebas de aceptación arrojaron resultados satisfactorios con lo que el cliente expresa su conformidad y satisfacción con el sistema creado. (Ver Anexo V).

## 4.6 Conclusiones

Una vez implementado el sistema y luego de un riguroso proceso de pruebas de aceptación por cada Historia de Usuario, que demuestran el grado de satisfacción de los clientes con el resultado obtenido, el proyecto se encuentra listo para ser explotado por los usuarios finales.

## CONCLUSIONES

El rápido avance de las tecnologías en el campo de la informática hace imperioso su utilización en todos los campos de la sociedad donde el ingenio del hombre puede ser asistido por computadoras. El proceso de confección de horarios docentes, por demás una tarea compleja, no es un caso aparte a esta realidad. Hoy en día, después de varios años de investigaciones en este campo, se puede decir que las tecnologías informáticas no han dicho su última palabra al resolver los problemas que esta implica.

Durante el desarrollo de este proyecto se realizó un estudio que permitió tener un conocimiento de la situación actual y las tendencias de los sistemas de planificación docente en el mundo, en nuestro país y específicamente en nuestra universidad. Este estudio demostró la necesidad de desarrollar un sistema capaz de gestionar la información pertinente a la planificación docente de la facultad 5 de la Universidad de las Ciencias Informáticas y lo suficientemente flexible para adaptarse a las necesidades de prácticamente cualquier centro de la enseñanza superior cubana.

Se presenta una solución web que permite gestionar la información referente a universidades, facultades, departamentos, carreras, disciplinas, asignaturas, grupos, profesores, alumnos ayudantes y locales; brindando al planificador docente una herramienta centralizada para realizar su trabajo. Además brinda la posibilidad a todas las personas implicadas en la docencia a obtener la información actualizada en todo momento referente a sus horarios docentes y sus preferencias. El sistema creado consta con un mecanismo que permite extender sus funcionalidades aun después de estar explotándose, lo que hace que pueda adaptarse a muchos centros docentes con características diferentes.

El desarrollo del proyecto utilizando las guías de la metodología *XP* permitió minimizar el tiempo en que los usuarios finales obtuvieron un producto listo para utilizarse, sin sacrificar la calidad de la documentación generada y dejando las puertas abiertas para futuras mejoras.

Se documentó todo el trabajo desde sus inicios lo que facilitará su estudio por parte de futuros desarrolladores, permitiendo una comprensión rápida y sencilla de la concepción general del sistema.

## RECOMENDACIONES

Al concluir el presente trabajo de diploma se recomienda:

- Continuar en el estudio de este tipo de aplicaciones con el fin de añadir mejoras funcionales enfocadas al beneficio del usuario.
- Incorporar funcionalidades que involucren técnicas de Inteligencia Artificial y generación automática de horarios.
- Codificar un conjunto más amplio de restricciones para facilitar la implantación del software en centros con características diferentes a nuestra universidad.
- Recoger las diversas opiniones de estudiantes y profesores con el objetivo de enriquecer y perfeccionar la aplicación.
- Elaborar archivos de ayuda más extensos y completos para facilitar el uso del sistema por personal inexperto.

**BIBLIOGRAFÍA**

**Beck, Kent. 1999.** Embracing Change with Extreme Programming (Revista de la IEEE Computer Society). 1999, Vol. 32, 10.

**Ciberaula. 2006.** ¿Qué es Java? . [En línea] 2006. [Citado el: 16 de marzo de 2009.] http://java.ciberaula.com/articulo/que_es_java/.

**Colectivo de autores. 2008.** Entorno Virtual de Aprendizaje. [En línea] 2008. [Citado el: 27 de marzo de 2009.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=16179.

**—. 2007.** Entorno Virtual de Aprendizaje. *El Lenguaje XHTML.* [En línea] 2007. [Citado el: 12 de abril de 2009.] http://eva.uci.cu/mod/resource/view.php?id=10531.

**—. 2008.** Entorno Virtual de Aprendizaje. *Introducción a las hojas de estilo.* [En línea] 2008. [Citado el: 12 de abril de 2009.] http://eva.uci.cu/mod/resource/view.php?id=10544.

**—. 2008.** Entorno Virtual de Aprendizaje. *Introducción al HTML Dinámico (DHTML). Lenguaje JavaScript.* [En línea] 2008. [Citado el: 19 de abril de 2009.] http://eva.uci.cu/mod/resource/view.php?id=10546.

**Espinoza, Humberto. 2005.** Consultores Informáticos Integrales. *PostgreSQL.* [En línea] 2005. [Citado el: 28 de marzo de 2009.] http://www.openworldconsult.com.ve/psql.

**Extjs. 2009.** API Documentation. [En línea] 2009. [Citado el: 19 de marzo de 2009.] http://www.extjs.com/docs.

**Fowler, Chad. 2006.** *Rails Recipes.* New York : The Pragmatic Bookshelf, 2006. 978-0-9776-1660-2.

**Fowler, Martin. 2003.** ProgramacionExtrema.org. *La Nueva Metodología.* [En línea] 2003. [Citado el: 16 de marzo de 2009.] http://www.programacionextrema.org/articulos/newMethodology.es.html.

**Hernán, S. M. 2004.** *Tesis de Grado en Ingeniería en Informática.* Ciudad de la Habana : Instituto Superior Politécnico José Antonio Echeverría, 2004.

**Jacobson, Ivar, Booch, Grady y Rumbaugh, J. 2000.** *El Proceso Unificado de Desarrollo de software.* Madrid : Addison-Wesley, 2000.

**Kabir, Mohammed J. 1998.** *La Biblia del Servidor Apache .* Madrid : Anaya, 1998.

**Microsoft Corporation. 2008.** SQL Server 2008. [En línea] 2008. [Citado el: 20 de marzo de 2009.] http://www.microsoft.com/sqlserver/2008/en/us/product-information.aspx.

**Molina, María Onelia. 1995.** *Higiene de la actividad docente.* Ciudad de la Habana : Pueblo y Educación, 1995.

**Netcraft. 2009.** Netcraft. *Market Share for Top Servers Across All Domains August 1995 - March 2009.* [En línea] 2009. [Citado el: 16 de marzo de 2009.] http://news.netcraft.com.

**Peñalara. 2009.** Generador de Horarios Para Centros de Enseñanza. [En línea] 2009. [Citado el: 10 de marzo de 2009.] www.penalara.com/ghc.asp.

**Potencier, Fabien y Zaninotto, Francois. 2008.** *Symfony la guía definitiva.* Madrid : www.librosweb.com, 2008.

**Programación en Castellano. 2005.** Programación en Castellano. *Programación en Castellano. ¿Que es ASP.NET?* [En línea] 2005. [Citado el: 15 de marzo de 2009.] www.programacion.com/asp/articulo/aspnet_quees.

**Rodríguez, C. C. 2003.** *Diseño y desarrollo de aplicaciones Web multidispositivo.* Madrid : Addison-Wesley, 2003.

**Sánchez, M. A. Mendoza. 2004.** Metodologías De Desarrollo De Software. *Informatizate.net.* [En línea] 2004. [Citado el: 12 de marzo de 2009.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

**Timetab. 2007.** Timetab. [En línea] 2007. [Citado el: 22 de marzo de 2009.] http://personal.telefonica.terra.es/web/timetab/espannol/index.html.

**Untis. 2009.** Untis. [En línea] 2009. [Citado el: 22 de marzo de 2009.] http://www.grupet.at/espanol/produkte/stundenplan/uebersicht.php.

**ANEXO I HISTORIAS DE USUARIO**

**Historia de Usuario: Navegar por el sistema**

| Historia de Usuario | |
|---|---|
| Número: 1 | Nombre historia: Navegar por el sistema |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 1 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que navegue por el sistema haciendo uso de un árbol de navegación, donde podrá seleccionar los diferentes elementos presentes en el sistema para ver su vista correspondiente en el panel principal. | |

**Historia de Usuario: Gestionar universidad**

| Historia de Usuario | |
|---|---|
| Número: 2 | Nombre historia: Gestionar universidad |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar universidades del sistema, además de editar y mostrar los datos de cada universidad presente en el sistema. | |
| Observaciones: Si existe una universidad con un nombre igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar facultad**

| Historia de Usuario | |
|---|---|
| Número: 3 | Nombre historia: Gestionar facultad |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar facultades del sistema, además de editar y mostrar los datos de cada facultad presente en el sistema. | |
| Observaciones: Si existe una facultad con un nombre igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar departamento**

| Historia de Usuario | |
|---|---|
| Número: 4 | Nombre historia: Gestionar departamento |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar departamentos del sistema, además de editar y mostrar los datos de cada departamento presente en el sistema. | |
| Observaciones: Si existe un departamento con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar carrera**

| Historia de Usuario | |
|---|---|
| Número: 5 | Nombre historia: Gestionar carrera |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar carreras universitarias del sistema, además de editar y mostrar los datos de cada carrera universitaria presente en el sistema. | |
| Observaciones: Si existe una carrera universitaria con un nombre igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar local**

| Historia de Usuario | |
|---|---|
| Número: 6 | Nombre historia: Gestionar local |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar locales del sistema, además de editar y mostrar los datos de cada local presente en el sistema. | |
| Observaciones: Si existe un local con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Editar preferencias de local**

| Historia de Usuario | |
|---|---|
| Número: 7 | Nombre historia: Editar preferencias de local |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite las preferencias de los locales del sistema, seleccionando la semana, el día y el turno y asignando un valor de disponibilidad para el mismo | |

**Historia de Usuario: Gestionar grupo**

| Historia de Usuario | |
|---|---|
| Número: 8 | Nombre historia: Gestionar grupo |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar grupos de estudiantes del sistema, además de editar y mostrar los datos de cada grupo presente en el sistema. | |
| Observaciones: Si existe un grupo con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Editar preferencias de grupo**

| Historia de Usuario | |
|---|---|
| Número: 9 | Nombre historia: Editar preferencias de grupo |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite las preferencias de los grupos de estudiantes del sistema, seleccionando la semana, el día y el turno y asignando un valor de felicidad para el mismo | |

**Historia de Usuario: Gestionar disciplina**

| Historia de Usuario | |
|---|---|
| Número: 10 | Nombre historia: Gestionar disciplina |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar disciplinas del sistema, además de editar y mostrar los datos de cada disciplina presente en el sistema. | |
| Observaciones: Si existe una disciplina con un nombre igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar asignatura**

| Historia de Usuario | |
|---|---|
| Número: 11 | Nombre historia: Gestionar asignatura |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar asignaturas del sistema, además de editar y mostrar los datos de cada asignatura presente en el sistema. | |
| Observaciones: Si existe una asignatura con un nombre igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Editar plan de clases**

| Historia de Usuario | |
|---|---|
| Número: 12 | Nombre historia: Editar plan de clases |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite los planes de clase de las asignaturas del sistema, agregando, editando o eliminando las clases programadas por semana. | |

**Historia de Usuario: Gestionar profesor**

| Historia de Usuario | |
|---|---|
| Número: 13 | Nombre historia: Gestionar profesor |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar profesores del sistema, además de editar y mostrar los datos de cada profesor presente en el sistema. | |
| Observaciones: Si existe un profesor con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Editar preferencias de profesor**

| Historia de Usuario | |
|---|---|
| Número: 14 | Nombre historia: Editar preferencias de profesor |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite las preferencias de los profesores del sistema, seleccionando la semana, el día y el turno y asignando un valor de felicidad para el mismo | |

**Historia de Usuario: Gestionar alumno ayudante**

| Historia de Usuario | |
|---|---|
| Número: 15 | Nombre historia: Gestionar alumno ayudante |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar alumnos ayudantes del sistema, además de editar y mostrar los datos de cada alumno ayudante presente en el sistema. | |
| Observaciones: Si existe un alumno ayudante con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Editar preferencias de alumno ayudante**

| Historia de Usuario | |
|---|---|
| Número: 16 | Nombre historia: Editar preferencias de alumno ayudante |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite las preferencias de los alumnos ayudantes del sistema, seleccionando la semana, el día y el turno y asignando un valor de felicidad para el mismo | |

**Historia de Usuario: Gestionar distribución**

| Historia de Usuario | |
|---|---|
| Número: 17 | Nombre historia: Gestionar distribución |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 1 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar distribuciones docentes, además de editar y mostrar los datos de cada distribución presente en el sistema. | |
| Observaciones: Las distribuciones docentes serán la relación existente entre los grupos, profesores, alumnos ayudantes, asignaturas, y tipos de turno. | |

**Historia de Usuario: Confeccionar horario**

| Historia de Usuario | |
|---|---|
| Número: 18 | Nombre historia: Confeccionar horario |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 2 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador a que edite los horarios docentes de grupos, profesores y alumnos ayudantes para que estos se ajusten a las necesidades de la organización. | |
| Observaciones: El proceso de edición de horarios docentes irá acompañado de una fuerte validación para evitar inconsistencias. | |

**Historia de Usuario: Gestionar año**

| Historia de Usuario | |
|---|---|
| Número: 19 | Nombre historia: Gestionar año |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar años del sistema, además de editar y mostrar los datos de cada año presente en el sistema. | |
| Observaciones: Si existe un año con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar semana**

| Historia de Usuario | |
|---|---|
| Número: 20 | Nombre historia: Gestionar semana |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar semanas del sistema, además de editar y mostrar los datos de cada semana presente en el sistema. | |
| Observaciones: Si existe una semana con un nombre y/o número igual a la que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar turno**

| Historia de Usuario | |
|---|---|
| Número: 21 | Nombre historia: Gestionar turno |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar turnos de clase del sistema, además de editar y mostrar los datos de cada turno presente en el sistema. | |
| Observaciones: Si existe un turno con un nombre y/o número igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar tipo de turno**

| Historia de Usuario | |
|---|---|
| Número: 22 | Nombre historia: Gestionar tipo de turno |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar tipos de turno de clase del sistema, además de editar y mostrar los datos de cada tipo de turno presente en el sistema. | |
| Observaciones: Si existe un tipo de turno con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar tipo de local**

| Historia de Usuario | |
|---|---|
| Número: 23 | Nombre historia: Gestionar tipo de local |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 1 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar tipos de local de clase del sistema, además de editar y mostrar los datos de cada tipo de local presente en el sistema. | |
| Observaciones: Si existe un tipo de local con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar conjunto**

| Historia de Usuario | |
|---|---|
| Número: 24 | Nombre historia: Gestionar conjunto |
| Usuario: Planificador | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar conjuntos al sistema, además de editar y mostrar los datos de cada conjunto presente en el sistema. | |
| Observaciones: Si existe un conjunto con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar elementos de conjunto**

| Historia de Usuario | |
|---|---|
| Número: 25 | Nombre historia: Gestionar elementos de conjunto |
| Usuario: Planificador | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de adicionar y eliminar elementos en los conjuntos del sistema, además de mostrar los elementos de cada conjunto presente en el sistema. | |
| Observaciones: No debe permitirse a un mismo conjunto poseer varias entradas de un mismo elemento | |

**Historia de Usuario: Gestionar restricción**

| Historia de Usuario | |
|---|---|
| Número: 26 | Nombre historia: Gestionar restricción |
| Usuario: Planificador | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Alto |
| Puntos estimados: 1 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar restricciones del sistema, además de editar y mostrar los datos de cada restricción presente en el sistema. Las restricciones poseerán parámetros que podrán ser gestionados y una lógica escrita en lenguaje PHP que podrá ser editada por el planificador. | |
| Observaciones: Si existe una restricción con un nombre igual al que se desea crear entonces no podrá crearse. | |

**Historia de Usuario: Gestionar parámetros de restricción**

| Historia de Usuario | |
|---|---|
| Número: 27 | Nombre historia: Gestionar parámetros de restricción |
| Usuario: Planificador | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de adicionar y eliminar parámetros a las restricciones del sistema, además se podrán editar y mostrar los tipos de datos de cada parámetro. | |
| Observaciones: Debe proporcionarse una lista de posibles valores para los tipos de datos de los parámetros. | |

**Historia de Usuario: Gestionar Instancia de Restricción**

| Historia de Usuario | |
|---|---|
| Número: 28 | Nombre historia: Gestionar instancia de restricción |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar instancias de las restricciones del sistema, proporcionando diferentes valores a los parámetros de estas. | |
| Observaciones: Cada instancia de restricción deberá tener un juego de valores único en sus parámetros. | |

**Historia de Usuario: Gestionar planificador**

| Historia de Usuario | |
|---|---|
| Número: 29 | Nombre historia: Gestionar planificador |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de crear y eliminar otros planificadores del sistema, además de editar y mostrar los datos de cada tipo de turno presente en el sistema. | |
| Observaciones: Si existe un planificador con un nombre igual al que se desea crear entonces no podrá crearse. Si el planificador intenta eliminar su propia cuenta el sistema debe impedirlo. | |

**Historia de Usuario: Gestionar seguridad**

| Historia de Usuario | |
|---|---|
| Número: 30 | Nombre historia: Gestionar seguridad |
| Usuario: Planificador | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador de que realice las acciones de de editar y mostrar los datos de la configuración de seguridad presente en el sistema. | |

**Historia de Usuario: Autenticar**

| Historia de Usuario | |
|---|---|
| Número: 31 | Nombre historia: Autenticar |
| Usuario: Usuario | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se brinda la posibilidad al usuario a que introduzca sus datos (nombre de usuario y contraseña) con la finalidad de verificarlos y otorgarle los permisos según el rol que cumpla dentro de la aplicación. | |
| Observaciones: El usuario y la contraseña deben ser válidos. Se le otorgarán los permisos de acuerdo a la configuración de seguridad del sistema. | |

**Historia de Usuario: Cerrar sesión**

| Historia de Usuario | |
|---|---|
| Número: 32 | Nombre historia: Cerrar sesión |
| Usuario: Planificador, Usuario Autenticado | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se le brinda la posibilidad al Planificador y al Usuario Autenticado a que cierre la sesión y continúe navegando por el sistema sin privilegios. | |

**Historia de Usuario: Consultar horarios generales**

| Historia de Usuario | |
|---|---|
| Número: 33 | Nombre historia: Consultar horarios generales |
| Usuario: Usuario | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se brinda la posibilidad al usuario a que acceda a los horarios publicados en el sistema. | |
| Observaciones: Se deben mostrar los horarios de acuerdo a la semana del curso correspondiente a la fecha actual y permitirse la selección de otras semanas. | |

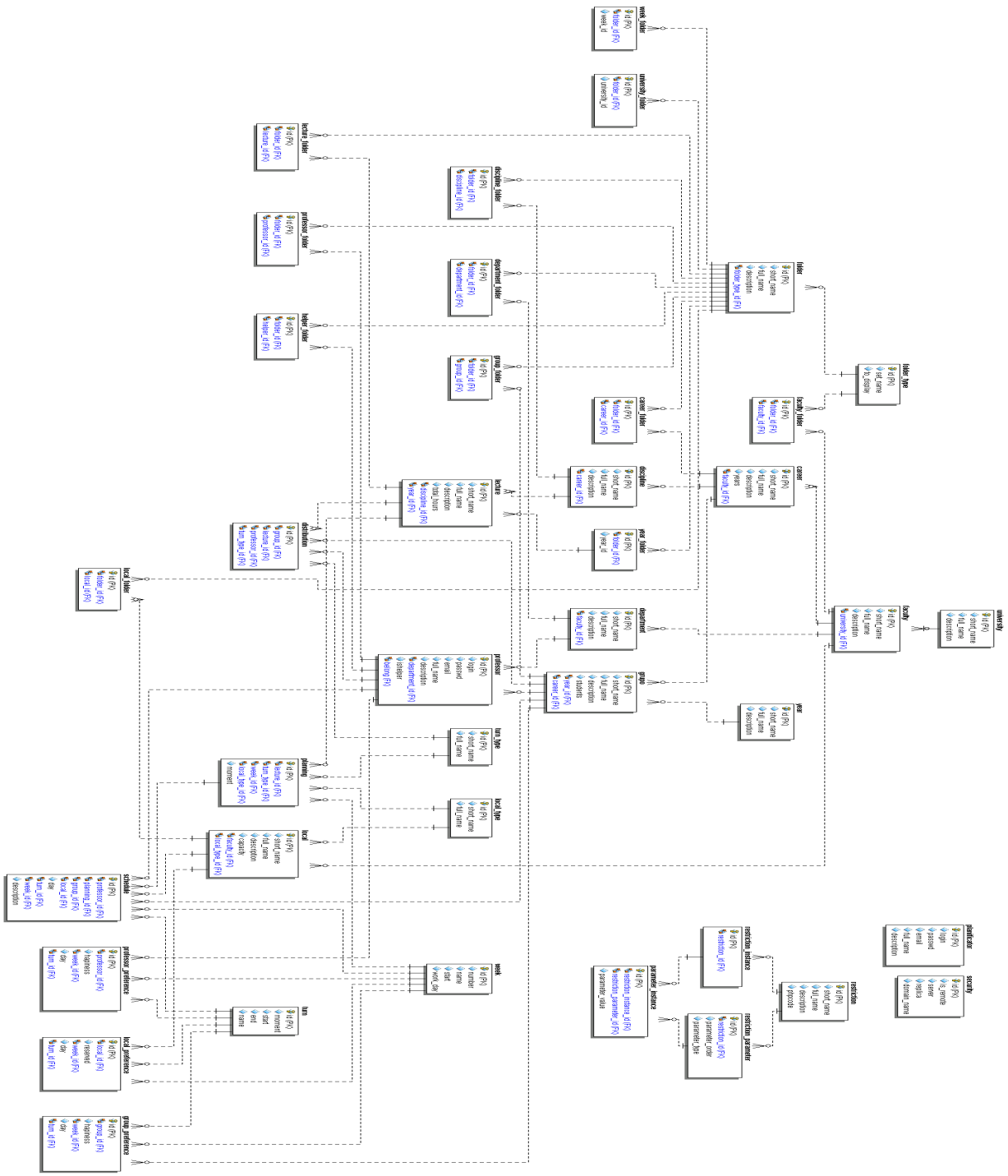
**Historia de Usuario: Consultar horarios específicos**

| Historia de Usuario | |
|---|---|
| Número: 34 | Nombre historia: Consultar horarios específicos |
| Usuario: Usuario Autenticado | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 0.5 | Iteración asignada: 2 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se brinda la posibilidad al usuario autenticado a que acceda a sus propios horarios publicados en el sistema. | |
| Observaciones: Se deben mostrar los horarios de acuerdo a la semana del curso correspondiente a la fecha actual y permitirse la selección de otras semanas. | |

**Historia de Usuario: Consultar ayuda**

| Historia de Usuario | |
|---|---|
| Número: 35 | Nombre historia: Consultar ayuda |
| Usuario: Planificador | |
| Prioridad en negocio: Baja | Riesgo en desarrollo: Medio |
| Puntos estimados: 2 | Iteración asignada: 3 |
| Programador responsable: Yerandy Reyes Fabregat | |
| Descripción: Se brinda la posibilidad al Planificador a que acceda a la ayuda del sistema. | |

## ANEXO II DISEÑO DE LA BASE DE DATOS



## ANEXO III SESIÓN DE TRABAJO CON SYMFONY DESDE LA CONSOLA DE WINDOWS

- **Elegir el directorio de trabajo:**

```
>cd g:\horarios

>g:
```

- **Crear el proyecto**

```
>%php% %symfony% generate:project horarios
```

- **Crear aplicaciones**

```
>%php% %symfony% generate:app planificador

>%php% %symfony% generate:app usuario
```

- **Crear módulos**

```
>%php% %symfony% generate:module planificador arbol

>%php% %symfony% generate:module usuario vista
```

- **Configurar la base de datos**

```
>%php% %symfony% configure:database mysql:dbname=schedules;host=localhost root password
```

- **Crear el esquema de la base de datos**

```
>%php% %symfony% propel:build-schema
```

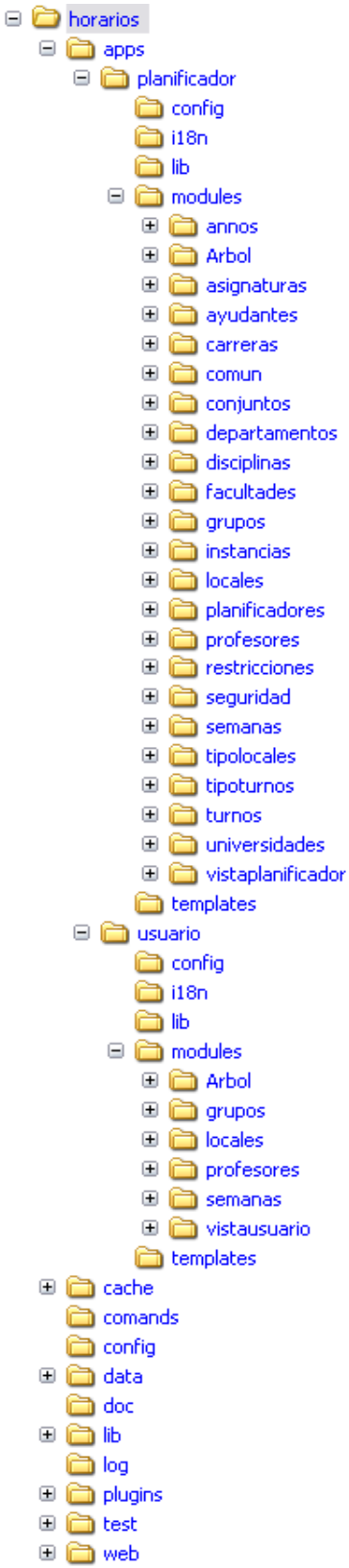
- **Crear las clases del modelo**

```
>%php% %symfony% propel:build-model
```

- **Borrar la caché del sistema**

```
>%php% %symfony% cc
```

## ANEXO IV ESTRUCTURA DE DIRECTORIOS DEL PROYECTO



## ANEXO V PRUEBAS DE ACEPTACIÓN DE LA PRIMERA ITERACIÓN

**Prueba de aceptación para la historia de usuario Navegar por el sistema.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU1_P1 | **Historia de Usuario:** Navegar por el sistema |
| **Nombre:** Navegar por el sistema | |
| **Descripción:** Probar que los hipervínculos funcionan correctamente, así como que el árbol de navegación es generado de manera correcta. | |
| **Condiciones de Ejecución:** | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación y se navega por todos los elementos del mismo. | |
| **Resultado Esperado:** El árbol de navegación se expandirá correctamente mostrando los elementos adecuados según el nodo en que se encuentre. Los elementos de tipo universidad, facultad, carrera, disciplina, grupo, asignatura, departamento, profesor, alumno ayudante, local, tipo de local, año, semana, turno y tipo de turno al ser accedidos deben crear una nueva pestaña en el panel principal, mostrando los datos correspondientes. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Navegar por el sistema.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU1_P2 | **Historia de Usuario:** Navegar por el sistema |
| **Nombre:** Navegar por el sistema | |
| **Descripción:** Probar que al eliminar un elemento que se encuentre abierto en el panel principal su pestaña se cierre automáticamente. | |

| |
|---|
| **Condiciones de Ejecución:**<br><br>Se eliminarán elementos de cada nodo padre en el árbol de navegación |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abren todos los elementos del árbol en el panel principal, se eliminan los elementos del árbol mediante su menú contextual. |
| **Resultado Esperado:** Al eliminar un elemento en el árbol de navegación cuyo contenido se encuentre mostrado en una pestaña del panel principal, esta se cierra automáticamente. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar universidad**.

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU2_P1 | **Historia de Usuario:** Gestionar universidad |
| **Nombre:** Crear una universidad | |
| **Descripción:** Probar que al crear un elemento de tipo universidad el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *university* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo universidad en el árbol, se despliega el menú contextual del nodo Universidades y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear una nueva universidad en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva universidad debe ser la palabra “universidad” seguido del identificador autoincrementado. El árbol | |

| |
|---|
| debe automáticamente expandir el nodo Universidades mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar universidad**.

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU2_P2 | **Historia de Usuario:** Gestionar universidad |
| **Nombre:** Editar una universidad | |
| **Descripción:** Probar que al editar un elemento de tipo universidad los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Universidades en el árbol de navegación, se selecciona un elemento de tipo universidad, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la universidad editada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar universidad.**

| **Caso de Prueba de Aceptación** |
|---|

| **Código:** HU2_P3 | **Historia de Usuario:** Gestionar universidad |
|---|---|
| **Nombre:** Editar una universidad | |
| **Descripción:** Probar que al editar un elemento de tipo universidad los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *university* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Universidades en el árbol de navegación, se selecciona un elemento de tipo universidad, se modifica el nombre completo de la universidad, haciéndolo coincidir con el nombre completo de otro elemento universidad que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar universidad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU2_P4 | **Historia de Usuario:** Gestionar universidad |
| **Nombre:** Editar una universidad | |
| **Descripción:** Probar que al editar un elemento de tipo universidad los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:** | |

| |
|---|
| La base de datos del sistema debe contener al menos dos entradas en la tabla *university* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Universidades en el árbol de navegación, se selecciona un elemento de tipo universidad, se modifica el nombre corto de la universidad, haciéndolo coincidir con el nombre corto de otro elemento universidad que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar universidad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU2_P5 | **Historia de Usuario:** Gestionar universidad |
| **Nombre:** Eliminar una universidad | |
| **Descripción:** Probar que al eliminar un elemento de tipo universidad los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Universidades en el árbol de navegación, se abre el menú contextual de un elemento de tipo universidad, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con esta universidad, deben ser | |

| |
|---|
| eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Universidades mostrando los elementos de tipo universidad remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU3_P1 | **Historia de Usuario:** Gestionar facultad |
| **Nombre:** Crear una facultad | |
| **Descripción:** Probar que al crear un elemento de tipo facultad, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *faculty.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo facultad en el árbol, se despliega el menú contextual del nodo Facultades perteneciente a un nodo de tipo universidad y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear una nueva facultad en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva facultad debe ser la palabra “facultad” seguido del identificador autoincrementado, el valor de la universidad a la que pertenece debe ser el identificador de la universidad del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Facultades perteneciente a la universidad en cuestión mostrando el elemento | |

| |
|---|
| creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU3_P2 | **Historia de Usuario:** Gestionar facultad |
| **Nombre:** Editar una facultad | |
| **Descripción:** Probar que al editar un elemento de tipo facultad los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br>La base de datos del sistema debe contener al menos una entrada en la tabla *faculty* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo facultad, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la facultad editada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** |
|---|

| **Código:** HU3_P3 | **Historia de Usuario:** Gestionar facultad |
|---|---|
| **Nombre:** Editar una facultad | |
| **Descripción:** Probar que al editar un elemento de tipo facultad los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *faculty* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo facultad, se modifica el nombre completo de la facultad, haciéndolo coincidir con el nombre completo de otro elemento facultad que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU3_P4 | **Historia de Usuario:** Gestionar facultad |
| **Nombre:** Editar una facultad | |
| **Descripción:** Probar que al editar un elemento de tipo facultad los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *faculty* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo facultad, se modifica el nombre corto de la facultad, haciéndolo coincidir con el nombre corto de otro elemento facultad que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU3_P5 | **Historia de Usuario:** Gestionar facultad |
| **Nombre:** Mover una facultad | |
| **Descripción:** Probar que al mover un elemento de tipo facultad en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *faculty* y dos entradas en la tabla *university.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo facultad, se selecciona la opción Cortar, se abre el menú contextual del nodo Facultades de otra universidad, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento facultad seleccionado para moverse, cambiando la universidad a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Facultades fuente y destino para mostrar la modificación realizada. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar facultad.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU3_P6 | **Historia de Usuario:** Gestionar facultad |
| **Nombre:** Eliminar una facultad | |
| **Descripción:** Probar que al eliminar un elemento de tipo facultad los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *faculty* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo facultad, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con esta facultad, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Facultades al que pertenece el elemento borrado mostrando los elementos de tipo facultad remanentes en la base de datos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU4_P1 | **Historia de Usuario:** Gestionar departamento |
| **Nombre:** Crear un departamento | |
| **Descripción:** Probar que al crear un elemento de tipo departamento, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university* y una en la tabla *faculty.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *department.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo departamento en el árbol, se despliega el menú contextual del nodo Departamentos perteneciente a un nodo de tipo facultad y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo departamento en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo departamento debe ser la palabra “departamento” seguido del identificador autoincrementado, el valor de la facultad a la que pertenece debe ser el identificador de la facultad del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Departamentos perteneciente a la facultad en cuestión mostrando el elemento creado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU4_P2 | **Historia de Usuario:** Gestionar departamento |

| |
|---|
| **Nombre:** Editar un departamento |
| **Descripción:** Probar que al editar un elemento de tipo departamento los datos del mismo se actualicen correctamente en la base de datos del sistema |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *department* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo departamento, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del departamento editado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU4_P3 | **Historia de Usuario:** Gestionar departamento |
| **Nombre:** Editar un departamento | |
| **Descripción:** Probar que al editar un elemento de tipo departamento los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *department* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo departamento, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento departamento que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU4_P4 | **Historia de Usuario:** Gestionar departamento |
| **Nombre:** Editar un departamento | |
| **Descripción:** Probar que al editar un elemento de tipo departamento los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *department* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo departamento, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento departamento que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU4_P5 | **Historia de Usuario:** Gestionar departamento |
| **Nombre:** Mover un departamento | |
| **Descripción:** Probar que al mover un elemento de tipo departamento en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *department* y dos entradas en la tabla *faculty.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo departamento, se selecciona la opción Cortar, se abre el menú contextual del nodo Departamentos de otra facultad, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento departamento seleccionado para moverse, cambiando la facultad a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Departamentos fuente y destino para mostrar la modificación realizada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar departamento.**

| **Caso de Prueba de Aceptación** |
|---|

| **Código:** HU4_P6 | **Historia de Usuario:** Gestionar departamento |
|---|---|
| **Nombre:** Eliminar un departamento | |
| **Descripción:** Probar que al eliminar un elemento de tipo departamento los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *department* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo departamento, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este departamento, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Departamentos al que pertenece el elemento borrado mostrando los elementos de tipo departamento remanentes en la base de datos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P1 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Crear una carrera | |
| **Descripción:** Probar que al crear un elemento de tipo carrera, el mismo se inserte | |

| |
|---|
| correctamente en la base de datos del sistema |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university* y una en la tabla *faculty.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *career.* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo carrera en el árbol, se despliega el menú contextual del nodo Carreras perteneciente a un nodo de tipo facultad y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear una nueva carrera en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva carrera debe ser la palabra “carrera” seguido del identificador autoincrementado, el valor de la facultad a la que pertenece debe ser el identificador de la facultad del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Carreras perteneciente a la facultad en cuestión mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P2 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Editar una carrera | |
| **Descripción:** Probar que al editar un elemento de tipo carrera los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *career* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo carrera, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la carrera editada. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P3 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Editar una carrera | |
| **Descripción:** Probar que al editar un elemento de tipo carrera los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *career* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo carrera, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento carrera que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P4 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Editar una carrera | |
| **Descripción:** Probar que al editar un elemento de tipo carrera los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *career* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo carrera, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento carrera que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P5 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Mover una carrera | |

| |
|---|
| **Descripción:** Probar que al mover un elemento de tipo carrera en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *career* y dos entradas en la tabla *faculty.* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo carrera, se selecciona la opción Cortar, se abre el menú contextual del nodo Carreras de otra facultad, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe actualizar el elemento carrera seleccionado para moverse, cambiando la facultad a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Carreras fuente y destino para mostrar la modificación realizada. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar carrera.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU5_P6 | **Historia de Usuario:** Gestionar carrera |
| **Nombre:** Eliminar una carrera | |
| **Descripción:** Probar que al eliminar un elemento de tipo carrera los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *career* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo carrera, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base | |

| |
|---|
| de datos. Si existen elementos relacionados con esta carrera, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Carreras al que pertenece el elemento borrado mostrando los elementos de tipo carrera remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU23_P1 | **Historia de Usuario:** Gestionar tipo de local |
| **Nombre:** Crear un tipo de local | |
| **Descripción:** Probar que al crear un elemento tipo de local, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *local_type.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Local en el árbol de navegación comprobando la no existencia de elementos tipo de local, se despliega el menú contextual del nodo Tipos de Local y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo tipo de local en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo tipo de local debe ser la cadena “tipo de local” seguido del identificador autoincrementado. El árbol debe automáticamente expandir el nodo Tipos de Local mostrando el elemento creado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar tipo de local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU23_P2 | **Historia de Usuario:** Gestionar tipo de local |
| **Nombre:** Editar un tipo de local | |
| **Descripción:** Probar que al editar un elemento tipo de local los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *local_type* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Local en árbol de navegación, se selecciona un elemento tipo de local, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del tipo de local editado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar tipo de local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU23_P3 | **Historia de Usuario:** Gestionar tipo de local |
| **Nombre:** Editar un tipo de local | |
| **Descripción:** Probar que al editar un elemento tipo de local los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |

| |
|---|
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *local_type* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Local en el árbol de navegación, se selecciona un elemento tipo de local, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento tipo de local que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU23_P4 | **Historia de Usuario:** Gestionar tipo de local |
| **Nombre:** Editar un tipo de local | |
| **Descripción:** Probar que al editar un elemento tipo de local los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *local_type* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Local en el árbol de navegación, se selecciona un elemento tipo de local, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento tipo de local que se encuentre en el sistema, se | |

| |
|---|
| presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU23_P5 | **Historia de Usuario:** Gestionar tipo de local |
| **Nombre:** Eliminar un tipo de local | |
| **Descripción:** Probar que al eliminar un elemento tipo de local los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *local_type* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Local en el árbol de navegación, se abre el menú contextual de un elemento tipo de local, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este tipo de local, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Tipos de Local mostrando los elementos tipo de local remanentes en la base de datos. | |
| | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P1 | **Historia de Usuario:** Gestionar local |
| **Nombre:** Crear un local | |
| **Descripción:** Probar que al crear un elemento de tipo local, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *university* y una en la tabla *faculty.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *local.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo local en el árbol, se despliega el menú contextual del nodo Locales perteneciente a un nodo de tipo facultad y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo local en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo local debe ser la palabra “local” seguido del identificador autoincrementado, el valor de la facultad a la que pertenece debe ser el identificador de la facultad del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Locales perteneciente a la facultad en cuestión mostrando el elemento creado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P2 | **Historia de Usuario:** Gestionar local |
| **Nombre:** Editar un local | |
| **Descripción:** Probar que al editar un elemento de tipo local los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *local* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo local, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del local editado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P3 | **Historia de Usuario:** Gestionar local |
| **Nombre:** Editar un local | |
| **Descripción:** Probar que al editar un elemento de tipo local los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *local* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el | |

| |
|---|
| árbol de navegación, se selecciona un elemento de tipo local, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento local que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P4 | **Historia de Usuario:** Gestionar local |
| **Nombre:** Editar un local | |
| **Descripción:** Probar que al editar un elemento de tipo local los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *local* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo local, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento local que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. | |
| | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P5 | **Historia de Usuario:** Gestionar local |
| **Nombre:** Mover un local | |
| **Descripción:** Probar que al mover un elemento de tipo local en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *local* y dos entradas en la tabla *faculty.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo local, se selecciona la opción Cortar, se abre el menú contextual del nodo Locales de otra facultad, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento local seleccionado para moverse, cambiando la facultad a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Locales fuente y destino para mostrar la modificación realizada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar local.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU6_P6 | **Historia de Usuario:** Gestionar local |

| |
|---|
| **Nombre:** Eliminar un local |
| **Descripción:** Probar que al eliminar un elemento de tipo local los datos del mismo se eliminen correctamente en la base de datos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *local* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo local, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este local, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Locales al que pertenece el elemento borrado mostrando los elementos de tipo local remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar año.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU19_P1 | **Historia de Usuario:** Gestionar año |
| **Nombre:** Crear un año | |
| **Descripción:** Probar que al crear un elemento de tipo año, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *year.* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Años en el árbol de navegación comprobando la no existencia de elementos de tipo año en el árbol, se despliega el menú contextual del nodo Años y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear un nuevo año en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo año debe ser la palabra "año" seguido del identificador autoincrementado. El árbol debe automáticamente expandir el nodo Años mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar año.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU19_P2 | **Historia de Usuario:** Gestionar año |
| **Nombre:** Editar un año | |
| **Descripción:** Probar que al editar un elemento de tipo año los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *year* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Años en el árbol de navegación, se selecciona un elemento de tipo año, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del año editado. | |
| | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar año.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU19_P3 | **Historia de Usuario:** Gestionar año |
| **Nombre:** Editar un año | |
| **Descripción:** Probar que al editar un elemento de tipo año los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *year* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Años en el árbol de navegación, se selecciona un elemento de tipo año, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento año que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar año.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU19_P4 | **Historia de Usuario:** Gestionar año |
| **Nombre:** Editar un año | |

| |
|---|
| **Descripción:** Probar que al editar un elemento de tipo año los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *year* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Años en el árbol de navegación, se selecciona un elemento de tipo año, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento año que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar año.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU19_P5 | **Historia de Usuario:** Gestionar año |
| **Nombre:** Eliminar un año | |
| **Descripción:** Probar que al eliminar un elemento de tipo año los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:** La base de datos del sistema debe contener al menos una entrada en la tabla *year* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Años en el árbol de navegación, se abre el menú contextual de un elemento de tipo año, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |

| |
|---|
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este año, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Años al que pertenece el elemento borrado mostrando los elementos de tipo año remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P1 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Crear un grupo | |
| **Descripción:** Probar que al crear un elemento de tipo grupo, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en las tablas *university, faculty, career.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *grupo.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo grupo en el árbol, se despliega el menú contextual del nodo Grupos perteneciente a un nodo de tipo carrera y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo grupo en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo grupo debe ser la palabra “grupo” seguido del identificador autoincrementado, el valor de la carrera a la que pertenece debe ser el identificador de la carrera del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Grupos perteneciente | |

| |
|---|
| a la carrera en cuestión mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P2 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Editar un grupo | |
| **Descripción:** Probar que al editar un elemento de tipo grupo los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *grupo* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo grupo, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del grupo editado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P3 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Editar un grupo | |
| **Descripción:** Probar que al editar un elemento de tipo grupo los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de | |

| |
|---|
| los nombres completos únicos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *grupo* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo grupo, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento grupo que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P4 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Editar un grupo | |
| **Descripción:** Probar que al editar un elemento de tipo grupo los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *grupo* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo grupo, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento grupo que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los | |

| |
|---|
| cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P5 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Mover un grupo | |
| **Descripción:** Probar que al mover un elemento de tipo grupo en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *grupo* y dos entradas en la tabla *career.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo grupo, se selecciona la opción Cortar, se abre el menú contextual del nodo Grupos de otra carrera, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento grupo seleccionado para moverse, cambiando la carrera a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Grupos fuente y destino para mostrar la modificación realizada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar grupo.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU8_P6 | **Historia de Usuario:** Gestionar grupo |
| **Nombre:** Eliminar un grupo | |
| **Descripción:** Probar que al eliminar un elemento de tipo grupo los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *grupo* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo grupo, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este grupo, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Grupos al que pertenece el elemento borrado mostrando los elementos de tipo grupo remanentes en la base de datos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P1 | **Historia de Usuario:** Gestionar disciplina |
| **Nombre:** Crear una disciplina | |
| **Descripción:** Probar que al crear un elemento de tipo disciplina, el mismo se inserte correctamente en la base de datos del sistema | |

| |
|---|
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en las tablas *university, faculty, career.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *discipline.* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo disciplina en el árbol, se despliega el menú contextual del nodo Disciplinas perteneciente a un nodo de tipo carrera y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear una nueva disciplina en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva disciplina debe ser la palabra “disciplina” seguido del identificador autoincrementado, el valor de la carrera a la que pertenece debe ser el identificador de la carrera del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Disciplinas perteneciente a la carrera en cuestión mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P2 | **Historia de Usuario:** Gestionar disciplina |
| **Nombre:** Editar una disciplina | |
| **Descripción:** Probar que al editar un elemento de tipo disciplina los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *discipline* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo disciplina, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la disciplina editada. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P3 | **Historia de Usuario:** Gestionar disciplina |
| **Nombre:** Editar una disciplina | |
| **Descripción:** Probar que al editar un elemento de tipo disciplina los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br>La base de datos del sistema debe contener al menos dos entradas en la tabla *discipline* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo disciplina, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento disciplina que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P4 | **Historia de Usuario:** Gestionar disciplina |
| **Nombre:** Editar una disciplina | |
| **Descripción:** Probar que al editar un elemento de tipo disciplina los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *discipline* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo disciplina, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento disciplina que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P5 | **Historia de Usuario:** Gestionar disciplina |

| |
|---|
| **Nombre:** Mover una disciplina |
| **Descripción:** Probar que al mover un elemento de tipo disciplina en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *discipline* y dos entradas en la tabla *career.* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo disciplina, se selecciona la opción Cortar, se abre el menú contextual del nodo Disciplinas de otra carrera, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe actualizar el elemento disciplina seleccionado para moverse, cambiando la carrera a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Disciplinas fuente y destino para mostrar la modificación realizada. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar disciplina.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU10_P6 | **Historia de Usuario:** Gestionar disciplina |
| **Nombre:** Eliminar una disciplina | |
| **Descripción:** Probar que al eliminar un elemento de tipo disciplina los datos del mismo se eliminen correctamente en la base de datos. | |

| |
|---|
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *discipline* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo disciplina, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con esta disciplina, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Disciplinas al que pertenece el elemento borrado mostrando los elementos de tipo disciplina remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU11_P1 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Crear una asignatura | |
| **Descripción:** Probar que al crear un elemento de tipo asignatura, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en las tablas *university, faculty, career y discipline.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *lecture.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo asignatura en | |

| |
|---|
| el árbol, se despliega el menú contextual del nodo Asignaturas perteneciente a un nodo de tipo disciplina y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear una nueva asignatura en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva asignatura debe ser la palabra “asignatura” seguido del identificador autoincrementado, el valor de la disciplina a la que pertenece debe ser el identificador de la disciplina del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Asignaturas perteneciente a la disciplina en cuestión mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU11_P2 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Editar una asignatura | |
| **Descripción:** Probar que al editar un elemento de tipo asignatura los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *lecture* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo asignatura, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la disciplina editada. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU11_P3 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Editar una asignatura | |
| **Descripción:** Probar que al editar un elemento de tipo asignatura los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *lecture* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo asignatura, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento asignatura que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU11_P4 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Editar una asignatura | |

| |
|---|
| **Descripción:** Probar que al editar un elemento de tipo asignatura los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *lecture* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo asignatura, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento asignatura que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU11_P5 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Mover una asignatura | |
| **Descripción:** Probar que al mover un elemento de tipo asignatura en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *lecture* y dos entradas en la tabla *discipline.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo asignatura, se selecciona la opción Cortar, se abre el menú contextual del nodo Asignaturas de otra | |

| |
|---|
| disciplina, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe actualizar el elemento asignatura seleccionado para moverse, cambiando la disciplina a la que pertenece por la indicada en la acción Pegar. El árbol de navegación debe actualizar los nodos Asignaturas fuente y destino para mostrar la modificación realizada. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar asignatura.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU11_P6 | **Historia de Usuario:** Gestionar asignatura |
| **Nombre:** Eliminar una asignatura | |
| **Descripción:** Probar que al eliminar un elemento de tipo asignatura los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *lecture* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo asignatura, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con esta asignatura, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Asignaturas al que pertenece el elemento borrado mostrando los elementos de tipo asignatura remanentes en la base de datos. | |
| | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU13_P1 | **Historia de Usuario:** Gestionar profesor |
| **Nombre:** Crear un profesor | |
| **Descripción:** Probar que al crear un elemento de tipo profesor, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en las tablas *university, faculty, department.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *professor.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo profesor en el árbol, se despliega el menú contextual del nodo Profesores perteneciente a un nodo de tipo departamento y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo profesor en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo profesor debe ser la palabra “profesor” seguido del identificador autoincrementado, el valor del departamento al que pertenece debe ser el identificador del departamento del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Profesores perteneciente al departamento en cuestión mostrando el elemento creado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU13_P2 | **Historia de Usuario:** Gestionar profesor |
| **Nombre:** Editar un profesor | |
| **Descripción:** Probar que al editar un elemento de tipo profesor los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *professor* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo profesor, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del profesor editado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU13_P3 | **Historia de Usuario:** Gestionar profesor |
| **Nombre:** Editar un profesor | |
| **Descripción:** Probar que al editar un elemento de tipo profesor los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:** | |

| |
|---|
| La base de datos del sistema debe contener al menos dos entradas en la tabla *professor* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo profesor, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento profesor que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU13_P4 | **Historia de Usuario:** Gestionar profesor |
| **Nombre:** Editar un profesor | |
| **Descripción:** Probar que al editar un elemento de tipo profesor los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br>La base de datos del sistema debe contener al menos dos entradas en la tabla *professor* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo profesor, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento profesor que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |

| |
|---|
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU13_P5 | **Historia de Usuario:** Gestionar profesor |
| **Nombre:** Mover un profesor | |
| **Descripción:** Probar que al mover un elemento de tipo profesor en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *professor* y dos entradas en la tabla *department.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo profesor, se selecciona la opción Cortar, se abre el menú contextual del nodo Profesores de otro departamento, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento profesor seleccionado para moverse, cambiando el departamento al que pertenece por el indicado en la acción Pegar. El árbol de navegación debe actualizar los nodos Profesores fuente y destino para mostrar la modificación realizada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar profesor.**

| |
|---|
| **Caso de Prueba de Aceptación** |

| **Código:** HU13_P6 | **Historia de Usuario:** Gestionar profesor |
|---|---|
| **Nombre:** Eliminar un profesor | |
| **Descripción:** Probar que al eliminar un elemento de tipo profesor los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *professor* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo profesor, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este profesor, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Profesores al que pertenece el elemento borrado mostrando los elementos de tipo profesor remanentes en la base de datos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P1 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Crear un alumno ayudante | |
| **Descripción:** Probar que al crear un elemento de tipo alumno ayudante, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:** | |

| |
|---|
| La base de datos del sistema debe contener al menos una entrada en las tablas *university, faculty, department.*<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *professor.* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación comprobando la no existencia de elementos de tipo alumno ayudante en el árbol, se despliega el menú contextual del nodo Alumnos Ayudantes perteneciente a un nodo de tipo departamento y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear un nuevo alumno ayudante en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo alumno ayudante debe ser la cadena “alumno ayudante” seguido del identificador autoincrementado, el valor del departamento al que pertenece debe ser el identificador del departamento del nodo al que pertenece en el árbol de navegación. El árbol debe automáticamente expandir el nodo Alumnos Ayudantes perteneciente al departamento en cuestión mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P2 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Editar un alumno ayudante | |
| **Descripción:** Probar que al editar un elemento de tipo alumno ayudante los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *profesor* y este debe tener el valor del campo *is_helper* igual a verdadero | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo | |

| |
|---|
| el árbol de navegación, se selecciona un elemento de tipo alumno ayudante, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del alumno ayudante editado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P3 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Editar un alumno ayudante | |
| **Descripción:** Probar que al editar un elemento de tipo alumno ayudante los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *profesor,* ambas con valor del campo *is_helper* igual a verdadero. | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo alumno ayudante, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento alumno ayudante que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P4 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Editar un alumno ayudante | |
| **Descripción:** Probar que al editar un elemento de tipo alumno ayudante los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *profesor,* ambas con valor del campo *is_helper* igual a verdadero. | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se selecciona un elemento de tipo profesor, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento alumno ayudante que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P5 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Mover un alumno ayudante | |
| **Descripción:** Probar que al mover un elemento de tipo alumno ayudante en el árbol de navegación, los datos del mismo se actualicen correctamente en la base de datos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *profesor,* con valor del campo *is_helper* igual a verdadero y dos entradas en la tabla *department.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo alumno ayudante, se selecciona la opción Cortar, se abre el menú contextual del nodo Alumnos Ayudantes de otro departamento, se selecciona la opción Pegar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe actualizar el elemento alumno ayudante seleccionado para moverse, cambiando el departamento al que pertenece por el indicado en la acción Pegar. El árbol de navegación debe actualizar los nodos Alumnos Ayudantes fuente y destino para mostrar la modificación realizada. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar alumno ayudante.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU15_P6 | **Historia de Usuario:** Gestionar alumno ayudante |
| **Nombre:** Eliminar un alumno ayudante | |

| |
|---|
| **Descripción:** Probar que al eliminar un elemento de tipo alumno ayudante los datos del mismo se eliminen correctamente en la base de datos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *profesor,* con valor en el campo *is_helper* igual a verdadero |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande todo el árbol de navegación, se abre el menú contextual de un elemento de tipo alumno ayudante, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este alumno ayudante, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Alumnos Ayudantes al que pertenece el elemento borrado mostrando los elementos de tipo alumno ayudante remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar semana.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU20_P1 | **Historia de Usuario:** Gestionar semana |
| **Nombre:** Crear una semana | |
| **Descripción:** Probar que al crear un elemento de tipo semana, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *week.* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Semanas en el árbol de navegación comprobando la no existencia de elementos de tipo semana en el árbol, se despliega el menú contextual del nodo Semanas y se selecciona la opción Nuevo. |
| **Resultado Esperado:** El sistema debe crear una nueva semana en la base de datos, con los valores por defecto en sus datos, el nombre de la nueva semana debe ser la palabra "semana" seguido del identificador autoincrementado. El árbol debe automáticamente expandir el nodo Semanas mostrando el elemento creado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar semana.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU20_P2 | **Historia de Usuario:** Gestionar semana |
| **Nombre:** Editar una semana | |
| **Descripción:** Probar que al editar un elemento de tipo semana los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *week* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Semanas en el árbol de navegación, se selecciona un elemento de tipo semana, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos de la semana editada. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar semana.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU20_P3 | **Historia de Usuario:** Gestionar semana |
| **Nombre:** Editar un semana | |
| **Descripción:** Probar que al editar un elemento de tipo semana los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *week* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Semanas en el árbol de navegación, se selecciona un elemento de tipo semana, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento semana que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar semana.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU20_P4 | **Historia de Usuario:** Gestionar semana |
| **Nombre:** Editar una semana | |

| |
|---|
| **Descripción:** Probar que al editar un elemento de tipo semana los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *week* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Semanas en el árbol de navegación, se selecciona un elemento de tipo semana, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento semana que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar semana.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU20_P5 | **Historia de Usuario:** Gestionar semana |
| **Nombre:** Eliminar una semana | |
| **Descripción:** Probar que al eliminar un elemento de tipo semana los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:** La base de datos del sistema debe contener al menos una entrada en la tabla *week* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Semanas en el árbol de navegación, se abre el menú contextual de un elemento de tipo semana, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |

| |
|---|
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con esta semana, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Semanas al que pertenece el elemento borrado mostrando los elementos de tipo semana remanentes en la base de datos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU22_P1 | **Historia de Usuario:** Gestionar tipo de turno |
| **Nombre:** Crear un tipo de turno | |
| **Descripción:** Probar que al crear un elemento tipo de turno, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *turn_type.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Turno en el árbol de navegación comprobando la no existencia de elementos tipo de turno en el árbol, se despliega el menú contextual del nodo Tipos de Turno y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo tipo de turno en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo tipo de turno debe ser la cadena “tipo de turno” seguido del identificador autoincrementado. El árbol debe automáticamente expandir el nodo Tipos de Turno mostrando el elemento creado. | |
| | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de turno.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU22_P2 | **Historia de Usuario:** Gestionar tipo de turno |
| **Nombre:** Editar un tipo de turno | |
| **Descripción:** Probar que al editar un elemento tipo de turno los datos del mismo se actualicen correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *turn_type* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Turno en el árbol de navegación, se selecciona un elemento tipo de turno, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. | |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del tipo de turno editado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar tipo de turno.**

| | |
|---|---|
| **Caso de Prueba de Aceptación** | |
| **Código:** HU22_P3 | **Historia de Usuario:** Gestionar tipo de turno |
| **Nombre:** Editar un tipo de turno | |

| |
|---|
| **Descripción:** Probar que al editar un elemento tipo de turno los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres completos únicos. |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *turn_type* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Turno en el árbol de navegación, se selecciona un elemento tipo de turno, se modifica el nombre completo del elemento, haciéndolo coincidir con el nombre completo de otro elemento tipo de turno que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres completos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU22_P4 | **Historia de Usuario:** Gestionar tipo de turno |
| **Nombre:** Editar un tipo de turno | |
| **Descripción:** Probar que al editar un elemento tipo de turno los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres cortos únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *turn_type* | |

| |
|---|
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Turno en el árbol de navegación, se selecciona un elemento tipo de turno, se modifica el nombre corto del elemento, haciéndolo coincidir con el nombre corto de otro elemento tipo de turno que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres cortos repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar tipo de turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU22_P5 | **Historia de Usuario:** Gestionar tipo de turno |
| **Nombre:** Eliminar un tipo de turno | |
| **Descripción:** Probar que al eliminar un elemento tipo de turno los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:** La base de datos del sistema debe contener al menos una entrada en la tabla *turn_type* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Tipos de Turno en el árbol de navegación, se abre el menú contextual de un elemento tipo de turno, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este tipo de turno, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Tipos de Turno al que pertenece el elemento borrado mostrando los elementos tipo de turno remanentes en la base de datos. | |

| |
|---|
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU21_P1 | **Historia de Usuario:** Gestionar turno |
| **Nombre:** Crear un turno | |
| **Descripción:** Probar que al crear un elemento de tipo turno, el mismo se inserte correctamente en la base de datos del sistema | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema no debe contener ninguna entrada en la tabla *turn.* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Turnos en el árbol de navegación comprobando la no existencia de elementos de tipo turno en el árbol, se despliega el menú contextual del nodo Turnos y se selecciona la opción Nuevo. | |
| **Resultado Esperado:** El sistema debe crear un nuevo turno en la base de datos, con los valores por defecto en sus datos, el nombre del nuevo turno debe ser la cadena “turno” seguido del identificador autoincrementado. El árbol debe automáticamente expandir el nodo Turnos mostrando el elemento creado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU21_P2 | **Historia de Usuario:** Gestionar turno |

| |
|---|
| **Nombre:** Editar un turno |
| **Descripción:** Probar que al editar un elemento de tipo turno los datos del mismo se actualicen correctamente en la base de datos del sistema |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos una entrada en la tabla *turn* |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Turnos en el árbol de navegación, se selecciona un elemento turno, se editan los campos de este elemento en el panel principal, se presiona en el botón guardar para hacer válidos los cambios realizados y se actualiza la pestaña mediante su menú contextual. |
| **Resultado Esperado:** El sistema debe mostrar los nuevos datos del turno editado. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU21_P3 | **Historia de Usuario:** Gestionar turno |
| **Nombre:** Editar un turno | |
| **Descripción:** Probar que al editar un elemento de tipo turno los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los nombres únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *turn* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Turnos en el árbol de navegación, se selecciona un elemento turno, se modifica el nombre del elemento, haciéndolo coincidir con el nombre de otro | |

| |
|---|
| elemento turno que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando nombres repetidos. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

**Prueba de aceptación para la historia de usuario Gestionar turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU21_P4 | **Historia de Usuario:** Gestionar turno |
| **Nombre:** Editar un turno | |
| **Descripción:** Probar que al editar un elemento de tipo turno los datos del mismo se actualicen correctamente en la base de datos del sistema sin violar las restricciones de los números únicos. | |
| **Condiciones de Ejecución:**<br><br>La base de datos del sistema debe contener al menos dos entradas en la tabla *turn* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Turnos en el árbol de navegación, se selecciona un elemento de tipo turno, se modifica el número del elemento, haciéndolo coincidir con el número de otro elemento turno que se encuentre en el sistema, se presiona en el botón guardar para hacer válidos los cambios realizados. | |
| **Resultado Esperado:** El sistema debe mostrar un mensaje de error, evitando números repetidos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

**Prueba de aceptación para la historia de usuario Gestionar turno.**

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Código:** HU21_P5 | **Historia de Usuario:** Gestionar turno |
| **Nombre:** Eliminar un turno | |
| **Descripción:** Probar que al eliminar un elemento de tipo turno los datos del mismo se eliminen correctamente en la base de datos. | |
| **Condiciones de Ejecución:** La base de datos del sistema debe contener al menos una entrada en la tabla *turn* | |
| **Entrada / Pasos de ejecución:** Se abre la aplicación Planificador, se expande el nodo Configuración/Turnos en el árbol de navegación, se abre el menú contextual de un elemento turno, se selecciona la opción Eliminar, se acepta en el diálogo de confirmación. | |
| **Resultado Esperado:** El sistema debe eliminar el elemento seleccionado de la base de datos. Si existen elementos relacionados con este turno, deben ser eliminados en cascada para evitar inconsistencias. Si el elemento se encuentra abierto en el panel principal, su pestaña debe cerrarse de manera automática. El árbol de navegación debe actualizar el nodo Turnos al que pertenece el elemento borrado mostrando los elementos turno remanentes en la base de datos. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

## GLOSARIO

**AJAX:** (Asynchronous JavaScript And XML) JavaScript asíncrono y XML. Técnica de desarrollo web que permite la creación de aplicaciones interactivas.

**ANSI SQL 92:** Estándar definido en el año 1992 por el Instituto Nacional Americano de Estándares (ANSI). Ver SQL.

**ASP:** (Active Server Pages) Tecnología web para la creación de páginas web dinámicas creada por Microsoft Corporation.

**ASP.NET:** Plataforma para aplicaciones web desarrollada y comercializada por Microsoft Corporation. Se considera la evolución de ASP.

**C:** Lenguaje de programación de alto nivel creado en 1972, en sus inicios orientado a la creación de Sistemas Operativos.

**C#:** (C Sharp) Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft Corporation como parte de la Plataforma .Net.

**CPU:** (Central Processing Unit) Unidad central de procesamiento. Componente fundamental de una computadora digital.

**CSS:** (Cascading Style Sheet) Hoja de estilo en casada. Lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

**FTP:** (File Transfer Protocol) Protocolo para transferencia de archivos. Pertenece a la familia de protocolos de internet. Se utiliza para intercambiar archivos entre computadoras conectadas a una red.

**Gzip:** (GNU Zip) Software libre utilizado para la compresión de archivos.

**Hosting:** Servicio para los usuarios de internet que permite el alojamiento de información como sitios webs en servidores remotos.

**HTML:** (Hypertext Markup Language) lenguaje de marcas de hipertexto. Usado para escribir documentos como páginas web.

**HTTP:** (Hypertext Transfer Protocol) Protocolo de transferencia de hipertexto. Protocolo que permite el intercambio de información variada entre clientes y servidores. Es el protocolo fundamental de la web.

**J2EE:** (Java 2 Enterprise Edition) Plataforma de desarrollo de nivel empresarial. Utiliza el lenguaje Java con una arquitectura distribuida.

**Javascript:** Lenguaje interpretado similar al Java ampliamente utilizado para agregar funcionalidades a páginas web.

**JSON:** (JavaScript Object Notation) Formato sencillo para intercambiar datos. Consiste básicamente en un arreglo asociativo de Javascript que se utiliza para incluir información de objetos.

**Ldap:** (Lightweight Directory Access Protocol) Protocolo ligero de acceso a directorios. Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

**LINUX:** Sistema operativo libre, altamente configurable. Se considera el producto insignia del Software Libre.

**Log:** Archivo de registro donde se almacena información sobre los determinados eventos que ocurren en una aplicación, sistema operativo o dispositivo.

**LTS:** (Long Term Support) Soporte por largo tiempo. Política que garantiza el soporte de un determinado producto por parte de sus productores o distribuidores aun después de que versiones más avanzadas del producto sean lanzadas.

**MVC:** (Model-View-Controller) Modelo-Vista-Controlador. Patrón de diseño que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

**ORM:** (Object-Relational Mapping) Mapeo objeto-relacional. Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

**PDF:** (Portable Document Format) Formato de documento portable. Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Está especialmente ideado para documentos susceptibles de ser impresos.

**Pdo:** (PHP Data Objects) Extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

**Pdo_mysql:** Extensión de PHP 5 para el trabajo con bases de datos almacenadas en gestores MySQL. Ver Pdo.

**Perl:** Lenguaje de programación interpretado creado en 1987. Es ampliamente utilizado para crear páginas web dinámicas.

**PHP:** (**P**hp **H**eader **P**reprocessor) lenguaje de programación interpretado creado en 1994, diseñado originalmente para la creación de páginas web dinámicas. Actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

**PL/SQL:** Lenguaje de programación incrustado en los gestores de bases de datos Oracle y PostgreSQL, soporta todas las características de SQL y añade nuevas funcionalidades.

**RAM:** (Ramdom Access Memory) Memoria de acceso aleatorio. Componente de una computadora digital que permite el almacenamiento rápido y volátil de datos.

**Release:** Término que identifica a un producto de software entregado a los usuarios finales para su explotación.

**Script:** Guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Usualmente es un archivo de texto.

**Sistema experto:** Sistemas informáticos que emulan el comportamiento de un experto en un dominio concreto. Son creados a partir de técnicas de Inteligencia Artificial.

**SMTP:** (Simple Mail Transfer Protocol) Protocolo simple de transferencia de correo. Protoclo para el intercambio de correo electrónico.

**SQL:** (Structured Query Language) Lenguaje estructurado de consultas. Es un lenguaje de acceso a bases de datos.

**UML:** (Unified Modeling Language) Lenguaje unificado de modelado. Lenguaje gráfico para modelar sistemas informáticos.

**UNIX:** Sistema Operativo creado en 1969. Es la base muchos sistemas operativos modernos como Linux.

**Visual Basic:** Lenguaje de programación orientado a objetos creado por Microsoft Corporation en 1991.

**Web:** Sistema de documentos interconectados por enlaces de hipertexto, disponibles en Internet.

**WINDOWS:** Familia de sistemas operativos creados por Microsoft Corporation desde 1985 hasta la fecha.

**XML:** (E**x**tensible **M**arkup **L**anguage) Lenguaje de marcas extensible. Metalenguaje que permite especificar la gramática de otros lenguajes específicos. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

**YAML:** (**Y**aml Ain´t **A**nother **M**arkup **L**anguage) Formato de serialización de datos legible por humanos inspirado en lenguajes como XML y C. No se considera un lenguaje de marcas. Su uso fundamental es el almacenamiento de información descriptiva.