



UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD DE TECNOLOGÍAS INTERACTIVAS

SISTEMA PARA LA PLANIFICACIÓN Y LA GENERACIÓN AUTOMÁTICA DEL HORARIO DOCENTE EN LA FACULTAD DE TECNOLOGÍAS INTERACTIVAS (IAHorDo)

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Katherine Borrero Meriño

Tutores: Ms.C. Yorgelys González López

Dr.C. Rey Segundo Guerrero Proenza

La Habana, 2024

*La inteligencia consiste no sólo en el conocimiento, sino también en la
destreza de aplicar los conocimientos en la práctica.
Aristóteles*



Figura 1

Que esta dedicatoria sea un reflejo de gratitud y amor hacia cada uno de ustedes. En este período de mi vida, he encontrado en Dios la guía constante y su ayuda incondicional. A mis padres, Alexis y Fidel, les dedico este trabajo por cuya fortaleza y amor han sido la base de mi crecimiento y éxito. A mi hermana Roset, que siempre ha sido mi referente. A todos aquellos que han enfrentado adversidades con valentía han demostrado que las dificultades no son barreras insuperables. Su determinación es un testimonio de que, con fe y esfuerzo, podemos superar cualquier obstáculo.

*Con gratitud y amor,
Katherine*

Agradecimientos

El estudio de esta carrera ha sido un reto para mí y muchas son las personas que me han ayudado a llegar hasta aquí de una forma u otra con sus motivaciones y sus consejos, estaré muy agradecida con cada uno de ellos. En especial a Dios que es mi sostén en todo momento.

A mis padres Lexis y Fidel que los adoro tanto y me han dado el tamaño que tengo hoy.

A mi esposo Daniel que me ha apoyado siempre.

A mi hermana Roset que me dado aliento en muchas ocasiones.

A mi mejor amiga Iliana que siempre me ofrece buenos consejos.

A mi suegra Dayana, por su corazón y comprensión, que ha sido una madre para mí.

A Caridad, por su cariño inagotable.

A mis hermanos en la fe Ismael, Marianne, Yordan, Maylín, Liliana, Elizabeth, Dayron, Elías y muchos otros que han dejado una gran huella en mi vida.

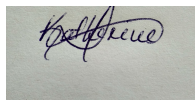
A mi intranquila compañera de cuarto Lisbet que constantemente tengo su abrazo ahí esperándome para cuando lo necesite.

Y no puedo dejar de mencionar a Helen, Tania, Alejandro Labaut y demás compañeros de clases que siempre me ha extendido su mano.

Declaración de autoría

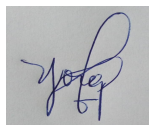
Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los 10 días del mes de diciembre del año 2024 .



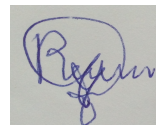
Katherine Borrero Meriño

Autor



Ms.C. Yorgelys González López

Tutora



Dr.C. Rey Segundo Guerrero Proenza

Cotutor

En la búsqueda de optimizar el proceso de confección del horario docente en la Facultad de Tecnologías Interactivas (FTI) de la Universidad de las Ciencias Informáticas (UCI), se ha identificado la necesidad imperante de un sistema automatizado. La metodología actual, realizada de manera manual, enfrenta desafíos significativos que impactan la efectividad del proceso educativo. Esta investigación propone el desarrollo de un sistema inteligente que, mediante el uso de algoritmos avanzados de optimización, genere automáticamente el horario docente. La solución propuesta se fundamenta en un análisis de los sistemas generadores de horarios y su interacción con las variables críticas del entorno académico, tales como la disponibilidad de recursos, preferencias del personal docente y limitaciones físicas de las instalaciones. Se ha seleccionado *Python* como lenguaje de programación principal, apoyado por el *framework Django* y la base de datos *PostgreSQL*, para garantizar un sistema escalable y confiable. Además, se adopta la *Extreme Programming (XP, por sus siglas en inglés)* como metodología de desarrollo, asegurando un enfoque sistemático y adaptable a los cambios. El resultado esperado es una herramienta de planificación y generación automática eficaz que facilite la toma de decisiones y promueva una distribución equitativa y coherente del horario docente. Con este sistema, se espera una mejora en la satisfacción de los estudiantes y profesores, así como una significativa reducción en los conflictos de programación, contribuyendo así a la excelencia académica de la Facultad.

Palabras clave: Algoritmo genético; generación automática; horario docente; IAHorDo.

Introducción	1
1 Fundamentos y referentes teórico-metodológicos sobre el proceso de planificación y generación automática del horario en la FTI.	7
1.1 Horarios docentes	7
1.2 Planificación del horario docente en la Facultad de Tecnologías Interactivas	9
1.3 Generación automática de horarios	10
1.4 Problema de planificación de horarios docentes	11
1.4.1 Métodos de solución	12
1.5 Análisis comparativo de módulos y sistemas automatizados existentes vinculados al campo de acción.	15
1.5.1 Sistemas y módulos de gestión de horarios nacionales	15
1.5.2 Sistemas y módulos de gestión de horarios internacionales	17
1.6 Herramientas y tecnologías	21
1.6.1 Lenguaje de modelado	21
1.6.2 Herramienta CASE	22
1.6.3 Entorno de desarrollo	23
1.6.4 Lenguaje de programación	24
1.7 Marco de trabajo para el desarrollo de la solución informática	26
1.7.1 Gestor de base de datos	27
1.8 Metodologías	27
2 Diseño de IAHorDo para la planificación y generación automática del horario en la FTI	32
2.1 Propuesta de solución	32
2.1.1 Secciones de IAHorDo	34
2.1.2 Asignación de variables	36
2.2 Algoritmo genético para la generación de horarios docentes	37
2.2.1 Requisitos no funcionales del sistema	51
2.3 Planificación del desarrollo	53
2.4 Usuarios del sistema	53

2.4.1	Historias de usuario	54
2.4.2	Estimación por esfuerzos	60
2.5	Plan de iteraciones	61
2.5.1	Plan de entregas	62
2.6	Diseño	62
2.6.1	Representación arquitectónica	63
2.6.2	Modelos de datos	64
2.7	Patrones de diseño	65
2.7.1	Patrones GRASP	65
2.7.2	Patrones GOF	66
2.7.3	Diagrama de Despliegue	67
2.7.4	Las tarjetas CRC	68
3	Validación de IAHorDo para la planificación y generación automática del horario en la FTL.	71
3.1	Tareas de Ingeniería	71
3.1.1	Estándares de codificación	75
3.2	Fase de Pruebas	77
3.2.1	Pruebas Unitarias	77
3.2.2	Resultados de las pruebas unitarias	81
3.2.3	Pruebas de Aceptación	82
	Conclusiones	91
	Recomendaciones	92
	Referencias bibliográficas	93
	Apéndices	98
A	Entrevistas	99
B	Historias de Usuario	101
C	Tarjetas CRC	115
D	Tareas de Ingeniería	121
E	Casos de pruebas de aceptación	129

1		II
2.1	Representación de las 11 secciones existentes en IAHorDo	35
2.2	Ciclo de la evolución del algoritmo genético.	38
2.3	Representación del MVT	64
2.4	Modelo de Datos	65
2.5	Diagrama de despliegue	68
3.1	Nombre de clases bajo la convención CapWord	75
3.2	Indentación del código basado en PEP8	76
3.3	Importaciones de librerías bajo PEP8	76
3.4	Prueba la modificación de una asignatura existente	78
3.5	Prueba la modificación de una asignatura que no existe.	79
3.6	Prueba la modificación con un nombre de asignatura ya existente.	79
3.7	Prueba la modificación con datos inválidos.	79
3.8	Prueba la modificación por un usuario que no es planificador.	80
3.9	Prueba la modificación por un usuario no autenticado.	80
3.10	Prueba la eliminación exitosa de un estudiante:	80
3.11	Prueba la eliminación de un estudiante que no existe:	81
3.12	Prueba la eliminación por un usuario que no es administrador:	81
3.13	Prueba la eliminación por un usuario no autenticado:	81
3.14	Resultados de las pruebas:	82
3.15	Resultados de las pruebas de aceptación	89

Índice de tablas

1.1	Tabla comparativa de las técnicas de optimización más utilizada. Fuente: Elaboración propia.	14
1.2	Resumen comparativo de sistemas y módulos nacionales. Fuente: Elaboración propia	16
1.3	Resumen comparativo de sistemas y módulos internacionales. Fuente: Elaboración propia . .	20
2.1	Restricciones y sus descripciones. Fuente: Elaboración propia.	33
2.2	Requisitos no funcionales. Fuente: Elaboración propia.	51
2.3	Usuarios del sistema. Fuente: Elaboración propia.	54
2.4	Historia de usuario # 1	55
2.5	Historia de usuario # 2	56
2.6	Historia de usuario # 3	58
2.7	Estimación de esfuerzo por historia de usuario	60
2.8	Plan de duración de las iteraciones	61
2.9	Plan de entregas. Fuente: Elaboración propia.	62
2.10	Tarjeta CRC # 1	69
2.11	Tarjeta CRC # 2	69
2.12	Tarjeta CRC # 3	70
3.1	Tarea de ingeniería # 1	72
3.2	Tarea de ingeniería # 2	72
3.3	Tarea de ingeniería # 3	72
3.4	Tarea de ingeniería # 4	72
3.5	Tarea de ingeniería # 5	73
3.6	Tarea de ingeniería # 6	73
3.7	Tarea de ingeniería # 7	73
3.8	Tarea de ingeniería # 8	74
3.9	Tarea de ingeniería # 9	74
3.10	Tarea de ingeniería # 10	74
3.11	Niveles de Pruebas. Fuente: Elaboración propia.	77
3.12	Prueba de aceptación # 1	83
3.13	Prueba de aceptación # 2	83
3.14	Prueba de aceptación # 3	84

3.15	Prueba de aceptación # 4	85
3.16	Prueba de aceptación # 5	85
3.17	Prueba de aceptación # 6	86
3.18	Prueba de aceptación # 7	87
3.19	Prueba de aceptación # 8	87
3.20	Prueba de aceptación # 9	88
B.1	Historia de usuario # 4	101
B.2	Historia de usuario # 5	102
B.3	Historia de usuario # 6	104
B.4	Historia de usuario # 7	105
B.5	Historia de usuario # 8	107
B.6	Historia de usuario # 9	108
B.7	Historia de usuario # 10	110
B.8	Historia de usuario # 11	111
B.9	Historia de usuario # 12	112
B.10	Historia de usuario # 13	113
C.1	Tarjeta CRC # 4	115
C.2	Tarjeta CRC # 5	115
C.3	Tarjeta CRC # 6	115
C.4	Tarjeta CRC # 7	116
C.5	Tarjeta CRC # 8	116
C.6	Tarjeta CRC # 9	116
C.7	Tarjeta CRC # 10	117
C.8	Tarjeta CRC # 11	117
C.9	Tarjeta CRC # 12	117
C.10	Tarjeta CRC # 13	117
C.11	Tarjeta CRC # 14	118
C.12	Tarjeta CRC # 15	118
C.13	Tarjeta CRC # 16	118
C.14	Tarjeta CRC # 17	119
C.15	Tarjeta CRC # 18	119
C.16	Tarjeta CRC # 19	119
C.17	Tarjeta CRC # 20	119
C.18	Tarjeta CRC # 21	120
D.1	Tarea de ingeniería # 11	121
D.2	Tarea de ingeniería # 12	121

D.3	Tarea de ingeniería # 13	121
D.4	Tarea de ingeniería # 14	122
D.5	Tarea de ingeniería # 15	122
D.6	Tarea de ingeniería # 16	122
D.7	Tarea de ingeniería # 17	123
D.8	Tarea de ingeniería # 18	123
D.9	Tarea de ingeniería # 19	123
D.10	Tarea de ingeniería # 20	123
D.11	Tarea de ingeniería # 21	124
D.12	Tarea de ingeniería # 22	124
D.13	Tarea de ingeniería # 23	124
D.14	Tarea de ingeniería # 24	124
D.15	Tarea de ingeniería # 25	125
D.16	Tarea de ingeniería # 26	125
D.17	Tarea de ingeniería # 27	125
D.18	Tarea de ingeniería # 28	126
D.19	Tarea de ingeniería # 29	126
D.20	Tarea de ingeniería # 30	126
D.21	Tarea de ingeniería # 31	127
D.22	Tarea de ingeniería # 32	127
D.23	Tarea de ingeniería # 33	127
D.24	Tarea de ingeniería # 34	127
D.25	Tarea de ingeniería # 35	128
E.1	Prueba de aceptación # 10	129
E.2	Prueba de aceptación # 11	129
E.3	Prueba de aceptación # 12	130
E.4	Prueba de aceptación # 13	131
E.5	Prueba de aceptación # 14	132
E.6	Prueba de aceptación # 15	132
E.7	Prueba de aceptación # 16	133
E.8	Prueba de aceptación # 17	134
E.9	Prueba de aceptación # 18	134
E.10	Prueba de aceptación # 19	135
E.11	Prueba de aceptación # 20	136
E.12	Prueba de aceptación # 21	136
E.13	Prueba de aceptación # 22	137
E.14	Prueba de aceptación # 23	138

E.15 Prueba de aceptación # 24	138
E.16 Prueba de aceptación # 25	139
E.17 Prueba de aceptación # 26	140
E.18 Prueba de aceptación # 27	140
E.19 Prueba de aceptación # 28	141
E.20 Prueba de aceptación # 29	142
E.21 Prueba de aceptación # 30	142
E.22 Prueba de aceptación # 31	143
E.23 Prueba de aceptación # 32	144
E.24 Prueba de aceptación # 33	145
E.25 Prueba de aceptación # 34	145

En la confluencia de la era digital, las Tecnologías de la Información y las Comunicaciones (TIC) han emergido como catalizadores de una transformación sin precedentes, reconfigurando el tejido de la sociedad contemporánea. Su expansión ha sido tal que han trascendido fronteras, transfigurando radicalmente diversos sectores, incluyendo el educativo, político, comercial y social. La omnipresencia de estas nuevas tecnologías ha generado cambios significativos en la forma de comunicación, interacción y realización de las tareas cotidianas. Su creciente dependencia ha dado lugar a una nueva forma de concebir la enseñanza y el aprendizaje, generando desafíos y oportunidades en la labor de los docentes universitarios. En los últimos 10 años, Cuba ha experimentado un notable avance en el uso masivo de las TIC con respecto al ámbito educacional, contribuyendo a la alfabetización digital de estudiantes y sociedad en general (Area Moreira; Gutiérrez Martín y Vidal Fernández, 2012). Estas permutaciones han originado un despliegue multidireccional en las actividades docentes, ampliando horizontes y facilitando el acceso a la información con una destreza nunca vista. En este cerco, la información y el conocimiento se han convertido en activos valiosos, marcando un antes y un después en la cultura informática y educativa (Castro Cáceres y Rodríguez Rojas, 2017).

En consonancia, el Ministerio de Educación de Cuba (MINED)¹ juega un papel fundamental en la organización y dirección del sistema educativo. El mismo tiene la misión de proponer al Estado y al Gobierno, dirigir y controlar, las políticas de educación superior referentes a la formación integral de los estudiantes de nivel superior, la educación de posgrado; y dirigir y controlar el desarrollo de la ciencia, la tecnología e innovación en las universidades y entidades de ciencias, así como la extensión de su quehacer a toda la sociedad (Educación, 2022). En el contexto de la educación superior, el proceso docente es un pilar fundamental que guía la formación académica y profesional de los estudiantes. Los docentes, como arquitectos del conocimiento, diseñan, organizan y ejecutan estrategias de enseñanza que son esenciales para el desarrollo intelectual y práctico de sus pupilos. Dentro de este cuadro, la planificación de actividades y la confección de horarios docentes emergen como tareas críticas, cuya eficacia repercute directamente en la calidad del proceso formativo. La gestión del tiempo y los recursos en la esfera educativa es una labor de gran envergadura que requiere responsabilidad y precisión. Además, la integración de las TIC en este proceso no sólo es una oportunidad para mejorar la eficiencia y la satisfacción de los involucrados, sino también un reto

¹ Refiere que el Ministerio de Educación Superior tiene la función de normar, asesorar y controlar el trabajo docente metodológico que desarrollan las instituciones de Educación Superior adscriptas a otros organismos.

que requiere un análisis detallado de las necesidades y limitaciones específicas de las facultades académicas.

En las universidades de Cuba, y en particular en la UCI, la confección del horario docente es un proceso que se organiza cuidadosamente y que sigue presentando desafíos significativos. La tarea de asignar secuencias de clases entre profesores y estudiantes, satisfaciendo un conjunto de condiciones predefinidas, se ha convertido en un proceso complejo que demanda soluciones innovadoras. La planificación manual de estos horarios, aunque tradicional, es susceptible a errores y resulta ardua, por esta razón ha llevado a buscar soluciones en las tecnologías de la información y las comunicaciones. Por consiguiente, no sólo prometen maximizar el aprovechamiento de los recursos y el tiempo, sino también reducir los riesgos inherentes al factor humano.

La elaboración del horario docente en la FTI es un proceso llevado a cabo por un planificador utilizando herramientas como el Excel, donde se organizan todas las semanas de clases del curso escolar. Este proceso manual implica que los directivos deben considerar múltiples factores:

- El Modelo de planificación y control del proceso docente (P1), donde se recogen los elementos que se tratarán en cada clase.
- El equilibrio del balance de carga de cada año académico que maneja información sobre las tipologías de clases que se impartirán y la frecuencia de las asignaturas por semana.
- La carga docente que incluye la distribución de los profesores por asignaturas, curso, grupo y año.
- La asignación de locales según su capacidad y requisitos específicos.
- Las afectaciones que influyen en la planificación del horario.

La fase de ajustes y resolución de conflictos es especialmente crítica, pues requiere una revisión del horario preliminar para identificar y corregir posibles superposiciones de clases y la disponibilidad de los locales para los implicados. Una vez que el horario es aprobado, se vuelve accesible para los estudiantes y profesores, ya sea por correo electrónico u otros medios. Esto conlleva una responsabilidad adicional para asegurar que el horario esté siempre disponible, lo que puede generar una carga adicional para el planificador. Este proceso de confección enfrenta varios desafíos que impactan la gestión académica de la facultad. La toma de decisiones por parte de los directivos, la coordinación de la planificación y elaboración del horario es extremadamente alta. Los Profesores principales de año (PPA) desempeñan un papel fundamental en este proceso, lo que implica un gran compromiso y un conocimiento profundo de las necesidades y recursos del entorno.

El proceso manual de confección del horario presenta varias dificultades que afectan su eficiencia. A pesar de ofrecer flexibilidad para ajustes, requiere un alto esfuerzo y tiempo, especialmente cuando se presentan cambios imprevistos. La dependencia de herramientas como el Excel y la intervención directa de los planificadores incrementan la complejidad del proceso. El manejo de múltiples factores como asignaturas, carga docente y distribución de aulas aumenta la probabilidad de errores humanos, como superposiciones

de clases o asignaciones incorrectas de espacios. Estos errores generan confusión y retrasan la toma de decisiones, lo que afecta la eficiencia del proceso. Además, el enfoque manual impone una carga adicional a los planificadores, dificultando la adaptación a cambios y aumentando el riesgo de conflictos. Esto resalta la necesidad de una solución automatizada que optimice la gestión del horario docente

Ante este panorama, se plantea el siguiente **problema de la investigación** ¿Cómo generar el horario docente de la FTI de forma automática?

El proceso de confección de horarios docentes se establece como el **objeto de estudio**.

El **objetivo general** de este trabajo es desarrollar un sistema web que permita la planificación y generación automática del horario docente de la FTI.

El **campo de acción** es la Informatización de horarios docentes.

Para cumplir con el objetivo planteado en esta investigación, se definieron un conjunto de **tareas de investigación** que guían el desarrollo del trabajo de manera estructurada. Estas tareas abarcan desde la fundamentación teórica y el análisis del problema hasta el diseño y la validación de la solución propuesta. Cada una de ellas responde a un propósito específico, asegurando un abordaje integral del campo de acción. A continuación, se detallan estas tareas.

- Definición de los fundamentos teóricos relacionados con los sistemas de planificación y generación automática de horarios académicos, para establecer un marco conceptual que respalde la propuesta.
- Descripción del proceso actual de planificación y confección del horario docente en la FTI, para identificar sus limitaciones y oportunidades de mejora.
- Revisión de las soluciones homólogas vinculadas al campo de acción, para identificar buenas prácticas, aplicables al contexto de la facultad.
- Investigación de algoritmos de optimización y técnicas adecuadas para la generación automática de horarios, para seleccionar aquellos que se adapten a las necesidades del sistema.
- Implementación del sistema para automatizar la planificación de horarios docentes.
- Realizar pruebas de *software*, siguiendo la metodología de desarrollo seleccionada para asegurar que cumplan con el objetivo planteado.

La confección del horario docente en la UCI es un reflejo de la dedicación y el débito de la institución con la excelencia académica. Sin embargo, la transición hacia un sistema automatizado que utilice algoritmos avanzados y tecnologías de la información se presenta como una solución prometedora para superar los desafíos actuales y mejorar el proceso docente.

Métodos científicos de investigación:

La aplicación de métodos científicos garantiza el rigor y la sistematicidad de esta investigación. Los métodos teóricos permitieron desarrollar el marco conceptual, analizar el problema y proponer soluciones fundamen-

tadas para la planificación y generación de horarios en la FTI. Complementariamente, los métodos empíricos aportaron información práctica al observar y evaluar los procesos actuales, identificando necesidades y oportunidades de mejora. Para la realización de este trabajo, se consultaron diversas fuentes bibliográficas que sustentan el uso de los métodos científicos descritos a continuación. Según Urbano "la aplicación de métodos teóricos y empíricos es fundamental para llevar a cabo una investigación rigurosa y obtener resultados confiables"(Urbano, 2006).

Métodos teóricos:

1. Analítico-sintético: Este método se utilizó durante el proceso de confección de horarios donde se deben ejecutar diferentes actividades, las cuales serán analizadas para identificar las características, limitaciones y necesidades y luego se agrupan esos resultados para sintetizar los elementos del proceso.
2. Histórico-lógico: A través de este método, se examinó la evolución de los sistemas de planificación de horarios, comprendiendo la lógica de su desarrollo y su relación con el contexto educativo actual.
3. Modelación: Se empleó para crear representaciones mediante los artefactos ingenieriles resultantes de la aplicación de la metodología de desarrollo de software a emplear.
4. Revisión documental: Se realizó una revisión exhaustiva de la literatura existente para fundamentar teóricamente la investigación y desarrollar un marco teórico sólido.

Métodos empíricos:

1. Observación: Se observaron las tareas actuales que se llevan a cabo en la confección de horarios de la FTI para identificar áreas de mejora y entender las dinámicas y retos existentes.
2. Entrevista: Se realizaron entrevistas al personal administrativo y directivo de la facultad: dígase planificador, PPA, Jefe de Departamento y Vicedecano de formación (VDF) para recopilar información cualitativa sobre sus experiencias, necesidades y expectativas respecto al horario docente (Ver anexo 1).

Estructura capitular del documento

La estructura capitular de este trabajo organiza de manera lógica y sistemática los aspectos más relevantes de la investigación, avanzando de lo teórico a lo práctico. Cada capítulo tiene un propósito específico, diseñado para guiar al lector a través del desarrollo del sistema propuesto. La investigación está dividida en tres capítulos, cuyos contenidos se describen a continuación:

Capítulo 1: Fundamentos y referentes teórico-metodológicos del proceso de planificación y generación automática del horario en la FTI. En el primer capítulo se establecerán las bases teóricas y metodológicas para el análisis y comprensión de la investigación, teniendo presente el estado del arte mediante el estudio de las soluciones informáticas existentes tanto a nivel nacional como internacional. Se llevará a cabo la selección de la metodología de desarrollo de software, la elección de lenguajes de programación, herramientas y tecnologías apropiadas que contribuirán al desarrollo de un sistema que responda a los desafíos de la planificación horaria en la educación superior, con el fin de optimizar la gestión del tiempo y los recursos docentes en la FTI.

Capítulo 2: Diseño del sistema para la planificación y generación automática del horario docente en la FTI. En el segundo capítulo, se aborda el diseño del sistema para la planificación y generación automática del horario docente de la FTI, adoptando la metodología XP. Se detalla el proceso llevado a cabo durante las fases de planificación y diseño, con énfasis en la elaboración de las historias de usuario y la configuración del plan de iteraciones alineado al cronograma de entrega. Además, se presenta la arquitectura seleccionada, destacando la aplicación de patrones de diseño. Se aborda la creación de un modelo de datos que respalda las necesidades del sistema. Asimismo, se examinan los requisitos no funcionales, aspectos cruciales que complementan la funcionalidad del software, garantizando su rendimiento y adaptabilidad.

Capítulo 3: Validación del sistema para la planificación y generación automática del horario docente en la FTI. En el tercer capítulo, se aborda la fase de validación del sistema para la gestión del horario docente de la FTI. Durante esta etapa, se llevan a cabo las tareas de ingeniería correspondientes a cada historia de usuario definida previamente en el proceso de planificación. La validación se realiza de manera iterativa, siguiendo el plan establecido en el capítulo anterior. Cada iteración se centra en la implementación de un conjunto de funcionalidades específicas, garantizando una entrega constante de valor al cliente. Además, se presenta una propuesta detallada de las pruebas que se llevarán a cabo para validar la calidad y el correcto

funcionamiento del sistema.

Fundamentos y referentes teórico-metodológicos sobre el proceso de planificación y generación automática del horario en la FTI.

En la búsqueda de la excelencia educativa, la generación automática de horarios emerge como una solución innovadora que promete servir de apoyo a la gestión académica en la Facultad de Tecnologías Interactivas de la Universidad de las Ciencias Informáticas. Este primer capítulo se dedica a desentrañar los fundamentos teóricos y metodológicos que sustentan la creación de horarios docentes, que no sólo optimizan recursos, sino que también minimizan conflictos operativos. Con un enfoque analítico y crítico, se examina el estado del arte en la planificación de horarios, destacando las herramientas y métodos que han marcado pauta tanto a nivel internacional como nacional. Esta exploración no es meramente descriptiva, sino que se adentra en la evaluación de su aplicabilidad en el contexto específico de la institución. La metodología de desarrollo seleccionada para este proyecto no es arbitraria; responde a un análisis riguroso de los requerimientos del sistema y las dinámicas de trabajo del equipo de desarrollo. Se prioriza una metodología que no sólo se alinea con los objetivos del sistema, sino que también promueve una implementación ágil y adaptativa.

1.1. Horarios docentes

Los **horarios docentes** son fundamentales en la planificación académica, estos estructuran el proceso educativo, asegurando que las actividades se realicen de manera ordenada. Su adecuada elaboración no sólo evita conflictos y tiempos muertos, sino que también optimiza el uso de recursos clave como aulas, laboratorios y otros espacios físicos (Peralta, 2000). En el entorno universitario, gestionar estos horarios es crucial para coordinar la disponibilidad de docentes y estudiantes, garantizando una asignación equilibrada que promueva la productividad y el cumplimiento de los objetivos educativos. Además, una planificación bien estructurada favorece un ambiente de trabajo organizado, permitiendo que las responsabilidades académicas se desarrollen sin interrupciones y mejorando la experiencia tanto para profesores como para estudiantes (Zabalza, 2023).

El diseño de los horarios docentes debe considerar los siguientes aspectos:

- Recopilación de datos: se recogen las necesidades de cursos, información sobre las materias y disponibilidad de los docentes.
- Espacios físicos: aulas teóricas, salones de conferencias y laboratorios disponibles, su capacidad y equipamiento.
- Restricciones y preferencias: limitaciones de tiempo, requisitos curriculares y preferencias personales.
- Elaboración de borradores: se generan versiones preliminares que se ajustan según las necesidades y restricciones.
- Optimización: se utilizan métodos y herramientas para encontrar la mejor distribución posible.
- Publicación y ajustes: una vez establecido, el horario se publica y se hacen ajustes según sea necesario.

Es crucial que los horarios sean flexibles para responder a imprevistos y que permitan ajustes mediante una gestión y comunicación efectivas. Esto implica coordinar múltiples variables, como la capacidad de las aulas, los requisitos de las asignaturas y las necesidades de los profesores, utilizando sistemas que faciliten la planificación académica.

La **planificación** según Robbins y DeCenzo: Es el proceso que se sigue para determinar en forma exacta lo que la organización hará para alcanzar sus objetivos, es un proceso sistemático que implica la evaluación de la situación actual, la definición de objetivos y metas, la identificación de estrategias y acciones a seguir y la implementación y seguimiento de dichas estrategia (Robbins, 2002). La planificación es una actividad proactiva que busca anticiparse al futuro, estableciendo un marco de referencia que guía las decisiones y actividades hacia resultados específicos. Se ocupa generalmente tanto de los fines como de los medios, y de la manera que hay que hacerlo (Javier y Teresa, 2019).

Planificar supone analizar y estudiar los objetivos propuestos, así como la forma en la que vamos a conseguirlos. Es una herramienta de acción para decidir qué vamos a hacer y por qué, supone crear un plan. El hecho de planificar tiene varias ventajas significativas:

- Actividades ordenadas y con un propósito: La planificación asegura que todos los esfuerzos se dirijan hacia los resultados deseados, pues facilita una secuencia efectiva y sincronizada de actividades docentes como conferencias, clases prácticas, laboratorios, talleres, seminarios, corte de trabajos extraclases, defensa de trabajos de curso, pruebas parciales y pruebas finales.
- Identificación de necesidades futuras: La planificación ayuda a identificar las necesidades futuras y a determinar las funcionalidades necesarias para alcanzar los objetivos.
- Control y seguimiento: La planificación proporciona una base para el control, pues los planes establecen los estándares que permiten evaluar el progreso y asegurar que se está en la dirección correcta.
- Conocimiento de los objetivos y adaptación al medio: La planificación ayuda a mantener los objetivos de la organización en mente de todos los componentes, esto facilita la adaptación a los cambios necesarios en el entorno.

La práctica de establecer horarios docentes es tan antigua como la propia institución educativa. Sin embargo, la planificación sistemática y la optimización de horarios como la conocemos hoy comenzaron a tomar forma con la expansión de la educación formal y la necesidad de administrar eficientemente el tiempo y los recursos en las crecientes poblaciones estudiantiles.

1.2. Planificación del horario docente en la Facultad de Tecnologías Interactivas

El proceso de planificación y confección del horario docente en la Facultad de Tecnologías Interactivas de la Universidad de las Ciencias Informáticas es un procedimiento meticuloso y complejo que se lleva a cabo durante el inicio de cada período. Este proceso es fundamental para garantizar que se cumplan las necesidades educativas de los profesores, optimizando al mismo tiempo los recursos disponibles.

La planificación comienza con la recopilación del balance de carga, la carga docente, el P1 de las asignaturas, la distribución de los locales y el horario tipo (Proyecto de investigación y desarrollo (PID), Inglés, Prueba parcial (PP), secciones de clases) que son proporcionados por el Vicedecano de Formación al planificador. A continuación se aborda detalladamente dicho proceso:

1. El balance de carga es esencial, pues incluye información sobre las tipologías de clases que se impartirán y la frecuencia de las asignaturas correspondientes. La carga docente contiene la asignación de los profesores por asignaturas, curso, grupo y año académico. El P1, contiene la dosificación docente y los elementos que se tratarán en cada clase durante el curso. Este documento es elaborado por los jefes de asignatura y es accesible para todos los profesores.
2. Los actores clave que orientan este proceso son el VDF, los jefes de departamento, los profesores principales del año, decano, secretario docente y el planificador. Los PPA dedican un considerable esfuerzo y tiempo a la planificación, asegurándose de que todas las variables y restricciones se tengan en cuenta.
3. Las afectaciones pueden incluir Preparación metodológica de cada disciplina, Consejo docente, Consejo de dirección, Consejo de producción, Consejo de extensión universitaria, Reunión de la facultad, Reunión del sindicato y otros compromisos que interfieran con la disponibilidad de los profesores recogidas por los PPA. Por otro lado, las restricciones son variadas y deben ser cuidadosamente consideradas. Por ejemplo, un grupo no puede tener programada más de una actividad en un mismo turno, y un profesor no puede tener más de una clase en el mismo turno. Además, las aulas deben ser seleccionadas de acuerdo a su capacidad, asegurando que el número de estudiantes no exceda el aforo permitido.

4. Se establece la higiene horario donde generalmente, las asignaturas que tienen varias frecuencias en la semana no pueden ser programadas de manera consecutiva; se prevé que, si una asignatura tiene dos frecuencias, estas se distribuyan con al menos un día de diferencia, se deben evitar turnos libres innecesarios entre clases. Además, la planificación debe considerar los tipos de clases, diferenciando entre profesores de conferencia (C) y profesores de clase práctica (CP). Los profesores de conferencia suelen impartir clases en salones para hasta dos grupos, mientras que los de clase práctica lo hacen en aulas teóricas o laboratorios generalmente para un solo grupo, lo que añade otra capa de complejidad al proceso.
5. Las sesiones de clases pueden programarse tanto en la mañana como en la tarde, y es crucial evitar solapamientos en los horarios, las mismas se planifican según el horario tipo de la universidad. Por ejemplo, si un profesor tiene una clase de conferencia en la FTI a primera hora, se planifica que su siguiente clase en otra facultad sea en un turno diferente, evitando así conflictos.
6. Los locales utilizados para la enseñanza incluyen salones de conferencias, aulas teórica, aulas especializadas y laboratorios. La asignación de estos espacios se realiza teniendo en cuenta el año académico, el tipo de curso, tipo de clase que se impartirá y la disponibilidad de los recursos.
7. Un desafío significativo en el pasado fue el uso del sistema de gestión universitaria Akademos que aunque no generaba el horario automáticamente, enfrentó problemas de lentitud debido a la alta demanda de acceso al servidor, lo que dificultaba la planificación efectiva del horario. Esto llevó a la necesidad de buscar soluciones más eficientes para la confección del horario docente.

El proceso de planificación y confección del horario docente en la FTI es un esfuerzo colaborativo que involucra a múltiples actores y requiere una atención meticulosa a los detalles. A través de la consideración de las afectaciones y restricciones, así como la optimización de los recursos disponibles, se busca crear un horario que no sólo cumpla con las necesidades académicas, sino que también mejore la experiencia educativa de los estudiantes. La implementación de sistemas automatizados en el futuro podría facilitar aún más este proceso, reduciendo el tiempo de confección y minimizando errores.

1.3. Generación automática de horarios

En la actualidad la generación automática de horarios docentes ha evolucionado significativamente impulsada por la implementación de algoritmos y técnicas computacionales avanzadas, que han permitido obtener resultados más precisos y rápidos. Este enfoque se ha vuelto esencial en el ámbito educativo debido a la creciente complejidad y volumen de información que debe considerarse en la planificación de horarios. A diferencia de los métodos tradicionales, la automatización permite enfrentar estos desafíos de manera más eficiente, ofreciendo soluciones precisas y adaptables a las necesidades de las instituciones. El uso de herra-

mientas informáticas para la creación de horarios docentes facilita un proceso más ágil y optimizado en la asignación de clases.

Estas tecnologías permiten reducir la carga de trabajo manual, minimizando así los errores humanos que suelen ocurrir en la elaboración tradicional de horarios. Los algoritmos analizan una vasta cantidad de datos, incluyendo la disponibilidad de los docentes, las capacidades de las aulas, las preferencias de los estudiantes y las restricciones impuestas por cada institución. A través de criterios y reglas predefinidas, estos sistemas generan horarios que maximizan el uso de los recursos disponibles, cumpliendo con todas las normativas y requisitos establecidos. Además, su capacidad para adaptarse a cambios, como modificaciones de última hora o ajustes en la carga docente, convierte a la generación automática de horarios en un proceso flexible y robusto.

Los beneficios de la automatización en la generación de horarios son múltiples. En primer lugar, se traduce en un ahorro considerable de tiempo y esfuerzo, pues libera a los encargados de la planificación académica de tareas repetitivas y monótonas. Además, al aplicar algoritmos avanzados, se disminuye la probabilidad de errores y conflictos en la asignación de clases, lo que resulta en una mejora en la precisión y calidad del proceso de planificación (Nelson; Fuentes y Candelaria, 2021). Otro aspecto relevante es la optimización del uso de recursos. Los algoritmos pueden considerar elementos como la carga docente y la capacidad del aula para asignar clases sin solapamientos. Esto no sólo mejora la distribución de las materias, sino que también promueve una carga equitativa entre los docentes.

1.4. Problema de planificación de horarios docentes

En el ámbito de la organización académica, el desafío de establecer un horario docente efectivo es un proceso complejo que requiere la consideración de múltiples factores y restricciones. La Planificación de horarios docentes (PHD) se enfoca en la asignación de clases para el personal docente dentro de un marco temporal establecido, buscando cumplir con una serie de condiciones específicas.

Las restricciones en la PHD se clasifican en dos categorías principales:

- **Restricciones obligatorias:** Son aquellas que deben ser cumplidas sin excepción, pues su incumplimiento resultaría en un horario inválido.
- **Restricciones preferenciales:** Representan las preferencias del usuario y, aunque idealmente se desean cumplir, no siempre es posible hacerlo simultáneamente debido a limitaciones prácticas.

1.4.1. Métodos de solución

Según el autor Franco Baquero para abordar el Problema de la planificación de horarios docentes (PPHD), existen dos enfoques principales, los modelos matemáticos, que tienen una base matemática o se apoyan en ella, y los heurísticos, que utilizan métodos algorítmicos (Michalewicz, 1996).

- Modelos matemáticos: basados en principios matemáticos, que ofrecen una solución precisa, pero pueden ser difíciles de implementar y requieren recursos computacionales significativos, como *hardware* potente, como procesadores y memoria RAM adecuados, así como *software* especializado, como herramientas de modelado y sistemas de optimización. Además, es fundamental contar con personal capacitado en matemáticas y análisis de datos para desarrollar y ajustar estos modelos.
- Métodos heurísticos: estos procedimientos proporcionan soluciones rápidas y prácticas, aunque no garantizan la más óptima. Se destacan por su facilidad de implementación y su capacidad para generar mejores soluciones en un tiempo razonable (Pereda, 2000).

Dado que la creación de horarios académicos no demanda un uso intensivo de recursos computacionales, se eligen métodos heurísticos, debido a la necesidad de encontrar una solución viable que satisfaga las restricciones obligatorias y reduzca al mínimo las violaciones de las restricciones preferenciales en un tiempo aceptable. Entre los **métodos heurísticos**, las metaheurísticas como los algoritmos genéticos, la colonia de hormigas y la búsqueda tabú han demostrado ser efectivas. Estos métodos se basan en estrategias inteligentes que permiten mejorar procedimientos heurísticos generales, optimizando así el proceso de planificación. Los métodos de metaheurísticas son estrategias avanzadas para resolver problemas de optimización complejos, como la planificación de horarios docentes (Nelson; Fuentes y Candelaria, 2021).

Algoritmos genéticos (AG):

Los AG son inspirados en el proceso de evolución biológica. Funcionan creando una población inicial de soluciones posibles (cromosomas) de forma aleatoria, que se someten a operaciones genéticas como la selección, la cría y la mutación. La selección favorece a los individuos más aptos, mientras que la cría combina características de dos o más individuos para crear descendencia. La mutación introduce variabilidad aleatoria. Con el tiempo, la población evoluciona hacia soluciones óptimas mediante un criterio de parada adecuado (Friman, 2012; David Elliott Goldberg y David E. Goldberg, 2012; Michalewicz, 1996; Suárez; Guerrero y Castrillón, 2013).

Colonia de hormigas (CH):

La CH se basa en el comportamiento de las hormigas en la naturaleza, que encuentran el camino más corto al alimento dejando un rastro de feromonas. En el contexto de la optimización, cada hormiga representa una solución potencial. Las hormigas exploran el espacio de soluciones y depositan feromonas en las rutas

que encuentran buenas, guiando a otras hormigas hacia estas soluciones. Con el tiempo, las rutas con más feromonas se vuelven más atractivas, convergiendo hacia una solución óptima (Carcía y miguel, 2009).

Búsqueda tabú (BT):

La BT es un método que busca evitar ciclos y óptimos locales en la búsqueda de soluciones. Utiliza una lista tabú que contiene movimientos recientes que no se pueden repetir durante un cierto número de iteraciones. Esto permite explorar nuevas áreas del espacio de soluciones sin regresar a soluciones ya visitadas. La búsqueda continúa hasta que se alcanza un nivel de satisfacción o se cumple un criterio preestablecido (Chávez Bósquez; Pozos Parra y Gómez Ramos, 2015).

Programación lineal y Métodos heurísticos

La programación lineal es un **método matemático** utilizado para encontrar una solución considerable a un problema de optimización. Este método se basa en la formulación de un modelo matemático que representa el problema de manera lineal, es decir, todas las relaciones entre las variables son lineales. También busca minimizar o maximizar una función objetivo sujeta a un conjunto de restricciones lineales, es particularmente útil en situaciones donde las relaciones entre las variables pueden ser modeladas de manera lineal. Por ejemplo, en la planificación de horarios docentes, si las restricciones y las preferencias pueden expresarse de forma lineal (como la disponibilidad de profesores y la capacidad de las aulas), entonces la programación lineal puede proporcionar una solución exacta (Ortega, 2023). En problemas de planificación de horarios docentes simples, la programación lineal puede ser muy efectiva. Sin embargo, su eficacia disminuye en presencia de restricciones más complejas o no lineales, donde otras técnicas como los algoritmos genéticos pueden ser más apropiadas (Munier, 1986).

Los métodos heurísticos son estrategias de resolución de problemas que buscan encontrar soluciones aproximadas de manera rápida. A diferencia de la programación lineal, que busca soluciones exactas, los métodos heurísticos se centran en encontrar soluciones satisfactorias en un tiempo determinado. Estos métodos son caracterizados por su velocidad y simplicidad. Utilizan reglas de *thumb* o reglas del pulgar", que son principios generales aplicables en diversas situaciones, aunque no siempre precisos. Estas reglas permiten tomar decisiones o realizar cálculos aproximados basados en la experiencia previa, facilitando el proceso de búsqueda y solución en contextos donde el tiempo y la eficiencia son cruciales. Por ejemplo, en la planificación de horarios docentes, un método heurístico podría comenzar asignando a los profesores a los cursos según sus preferencias más fuertes antes de pasar a las menos fuertes. Los métodos heurísticos son altamente flexibles y pueden adaptarse a una amplia gama de problemas, incluso aquellos con múltiples restricciones y preferencias complejas. Sin embargo, a diferencia de la programación lineal, no garantizan la obtención de la solución óptima global. Aunque los métodos heurísticos pueden ser muy útiles para encontrar solu-

CAPÍTULO 1. FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL PROCESO DE PLANIFICACIÓN Y GENERACIÓN AUTOMÁTICA DEL HORARIO EN LA FTI.

ciones rápidas, no están exentos de limitaciones. No garantizan la solución óptima, y su eficacia depende en gran medida de la habilidad del diseñador del método para capturar las características clave del problema.

A continuación en la tabla, se muestra un cuadro resumen de las técnicas de optimización más utilizadas en la planificación de horarios:

Tabla 1.1. Tabla comparativa de las técnicas de optimización más utilizada. Fuente: Elaboración propia.

Técnicas de Optimización	Descripción	Utilidad según el contexto	Beneficios	Limitantes
Programación lineal	Se utiliza para asignar recursos limitados (aulas, profesores, entre otros) a actividades (clases) de manera óptima.	Ideal para horarios con restricciones lineales y objetivos claros.	Maximiza la utilización de recursos y minimiza costos.	Requiere que las restricciones sean lineales y que los datos sean precisos.
Algoritmos genéticos	Inspirados en la evolución biológica, estos algoritmos generan soluciones mediante selección, cruce y mutación.	Útiles cuando se conocen las restricciones exactas o cuando hay múltiples objetivos.	Exploran soluciones diversas y pueden encontrar óptimos globales.	Puede requerir ajuste de parámetros y es computacionalmente costoso.
<i>Simulated annealing</i>	Basado en el proceso de enfriamiento de metales, busca soluciones óptimas explorando el espacio de búsqueda.	Adecuado para problemas con muchas restricciones y soluciones locales subóptimas.	Encuentra soluciones cercanas al óptimo global y evita quedar atrapado en mínimos locales.	Requiere ajuste de parámetros y puede ser lento en converger, es decir, de dos o más elementos que se dirigen hacia un estado común.
Heurísticas constructivas	Generan soluciones paso a paso, construyendo horarios de manera incremental. Ejemplos: <i>Greedy</i> , <i>Rule-Based</i> .	Rápidas y fáciles de implementar. Útiles para horarios pequeños o con restricciones simples.	Eficientes en tiempo de ejecución y pueden proporcionar soluciones aceptables.	No garantizan la optimalidad y pueden quedar atrapadas en soluciones subóptimas.

La programación lineal, aunque eficiente para problemas con restricciones bien definidas y relaciones lineales, se enfrenta a limitaciones cuando el problema incluye múltiples preferencias y restricciones contradictorias, como disponibilidad de profesores, capacidades de aulas y normativas específicas. Por otro lado, la búsqueda tabú, aunque es eficaz para explorar el espacio de soluciones y evitar estancamientos en óptimos locales, puede ser menos eficiente en problemas con alta variabilidad y dinámicas complejas debido a su enfoque más limitado en la exploración global del espacio de búsqueda. Del mismo modo, las colonias de

hormigas, si bien son excelentes para problemas de rutas y optimización basada en grafos, pueden ser menos adecuadas para escenarios con múltiples dimensiones y restricciones dinámicas como los que se presentan en la planificación de horarios docentes.

Los AG, en contraste, ofrecen una solución evolutiva que permite generar y evaluar una amplia gama de posibles horarios, explorando eficazmente el espacio de búsqueda para encontrar combinaciones óptimas. Su enfoque basado en procesos de selección, cruce y mutación les otorga la flexibilidad necesaria para adaptarse a cambios imprevistos, como modificaciones en la disponibilidad de recursos o la cancelación de clases. Además, su capacidad para priorizar tanto la diversidad de soluciones como la optimización asegura que se cumplan las restricciones específicas impuestas al sistema, al tiempo que se consideran las preferencias individuales y colectivas.

En resumen, los AG representan una herramienta poderosa y versátil para la optimización de horarios docentes, ofreciendo soluciones que no sólo cumplen con las restricciones actuales, sino que también están preparadas para adaptarse a futuras demandas y cambios. Su flexibilidad y capacidad para optimizar hacen que sean una opción preferente en el diseño de sistemas de planificación educativa (Nelson; Fuentes y Candelaria, 2021).

1.5. Análisis comparativo de módulos y sistemas automatizados existentes vinculados al campo de acción.

Luego del análisis de los principales conceptos asociados al dominio del problema se presentan soluciones informáticas a nivel internacional y nacional relacionadas con la planificación y generación de horarios, evaluando sus características y funcionalidades. Este estudio permitirá identificar sus fortalezas, debilidades y áreas de mejora, con el propósito de determinar la viabilidad de su uso en la solución de la problemática planteada. Los resultados de este análisis servirán como base para el diseño e implementación del sistema de gestión de horarios destinado a la Facultad de Tecnologías Interactivas.

1.5.1. Sistemas y módulos de gestión de horarios nacionales

Cuba no se encuentra ajena al desarrollo de sistemas capaces de satisfacer las necesidades en cuanto a la confección de horarios. A continuación, se presentarán algunos ejemplos de estos sistemas y módulos de gestión de horarios nacionales:

"Sistema de gestión para la planificación docente en la Facultad Regional Mártires de Artemisa"

Este sistema fue desarrollado en 2012, el mismo contiene un grupo de funcionalidades para manipular la información generada y así mejorar la integridad de la información, específicamente almacenar, modificar,

eliminar y acceder a los datos que se generan en el departamento, dígame gestión de usuarios, planificación de horarios, planificación de cuidados de prueba y control de registro de asistencias a clases. Además, destacar los aportes sociales proporcionados como: ahorro de tiempo y rapidez en la realización del trabajo, rapidez en los reportes sistemáticos, seguridad y organización de la información que se maneja. El sistema no se encuentra disponible en la actualidad, por lo que no puede tomarse como solución al problema existente en la FTL. Su desarrollo buscó aprovechar en un tiempo, las tecnologías existentes en la Facultad Regional Mártires de Artemisa y mejorar la manipulación de la documentación generada en el Departamento pero hoy no es así (Abreu Guerra, 2012).

"Módulo de gestión de horarios de Akademos"

Este es un sistema de gestión universitaria desarrollado por un equipo de trabajo de la "Dirección de Informatización de la UCI". Permite la gestión de los elementos principales que intervienen en la labor académica y presenta la opción de mitigar los riesgos que imponen el dinamismo del proceso de gestión académica. Fue utilizado para la gestión del horario docente de la universidad, proceso que se llevaba a cabo de forma manual, cabe destacar que enfrentó problemas de lentitud debido a la alta demanda de acceso al servidor, lo que dificultaba la planificación efectiva del horario.

"Módulo de generación de horarios docentes para el Sistema de Trabajo Educativo en la Facultad 1"

Durante el año 2022 la investigación del trabajo de diploma titulado "Módulo de generación de horarios docentes para el Sistema de Trabajo Educativo en la Facultad 1" se enfocó en crear una herramienta eficiente para la generación automática del horario docente, un módulo integrado a un Sistema de Trabajo Educativo (STE). El módulo permitía la gestión de distintos datos necesarios para el proceso, tales como: asignaturas, afectaciones docentes, balance de carga, locales, pruebas parciales y la carga docente. Sin embargo, este sistema no ofrece funcionalidades de edición del horario una vez generado. Este proyecto surgió para abordar las limitaciones del proceso manual existente, con el objetivo de generar múltiples opciones de horarios óptimos y reducir el tiempo y recursos necesarios para su elaboración (Blanco y Alejandro, 2022).

Luego de realizado el análisis anteriormente descrito se expone un resumen de las principales características de los sistemas analizados.

Tabla 1.2. Resumen comparativo de sistemas y módulos nacionales. Fuente: Elaboración propia

Característica	Facultad Regional Mártires de Artemisa	STE de la Facultad 1	Akademos
Libre de licencia	Sí	Sí	Sí
Continúa en la siguiente página			

Característica	Facultad Regional Mártires de Artemi- sa	STE de la Facultad 1	Akademios
Sin costo	Sí	Sí	Sí
Automatización del horario	Sí	Sí	No
Consideración de cambios do- centes	No	No	Sí
Adaptabilidad al cambio	No especificado	No	Sí
Vigencia del sistema	No especificado	No especificado	No

En la Tabla 1.2 se puede observar que los sistemas y módulos estudiados a nivel nacional no cumplen con todos los requisitos necesarios para dar solución al problema planteado en la presente investigación, el primero solamente se limitó a analizar y diseñar el sistema para la planificación no automatizada del horario docente, el segundo hoy en día no está a la disposición y el último no permite la edición del horario una vez generado.

1.5.2. Sistemas y módulos de gestión de horarios internacionales

En la actualidad se han desarrollado disímiles sistemas informáticos que permiten darle solución al PPHD, generalmente desarrolladas a la medida, solamente tienen en cuenta las características y restricciones propias de la institución a la que va dirigida el sistema. A continuación se presentan algunos de los más relevantes.

GHC

El Generador de horarios para centros de enseñanza (GHC) es una herramienta diseñada para facilitar la gestión de los horarios en centros educativos. La creación de horarios es un procedimiento complicado que requiere más que simplemente programar sesiones; debe tener en cuenta criterios pedagógicos, el uso óptimo de espacios físicos, y las necesidades y preferencias del personal docente (Penalara, 2022).

El GHC se estructura en tres componentes principales:

- **PLANIFICADOR:** permite importar datos y organizar horarios complejos de manera visual y asistida. Establece condiciones tanto estrictas como flexibles para encontrar soluciones que se ajusten a los criterios del usuario. Si las condiciones son imposibles de cumplir, el sistema ayuda a identificarlas y a depurarlas.
- **MOTOR:** una vez que el planificador ha configurado los elementos y condiciones, el motor busca automáticamente un compromiso óptimo que sea equitativo y favorable según los criterios establecidos por el usuario. Esto ayuda a crear horarios compactos y a satisfacer las preferencias del profesorado.
- **EDITOR:** esta herramienta permite presentar y modificar los horarios creados. Ofrece funciones como añadir actividades del profesorado, ajustar sesiones y transferir los horarios a otros formatos o

aplicaciones.

El planificador configura elementos como materias, grupos de alumnos, profesores, aulas y cursos, permite además definir condiciones estrictas y ventajosas como disponibilidad de docentes y preferencias para el uso de locales. El motor busca la solución más acertada atendiendo a estas configuraciones y evita superposiciones entre grupos, profesores y locales. El editor facilita la modificación y presentación de los resultados, permitiendo visualizarlos en distintos formatos y transferirlos a otras aplicaciones.

Untis

Untis se destaca en el ámbito global por su habilidad para automatizar la elaboración de horarios en entornos educativos. Su algoritmo sofisticado permite la creación de horarios que optimizan el uso de recursos y espacios, alineándose con los objetivos pedagógicos definidos. Con una presencia que abarca más de 80 países y disponible en una amplia gama de idiomas, Untis es utilizado por una gran cantidad de instituciones educativas, incluyendo colegios, universidades y centros de formación (Educaria, 2022).

Las ventajas claves de este sistema de horarios flexibles son:

- **Adaptabilidad:** permite a los empleados organizar su tiempo sin comprometer sus actividades personales, trabajando en momentos óptimos o ajustando sus horarios de entrada y salida.
- **Mejora en la salud mental:** promueve un equilibrio entre el trabajo y la vida personal, lo que puede reducir el estrés y aumentar la productividad y creatividad.
- **Sentimiento de autonomía:** la flexibilidad en los horarios otorga a los empleados un sentido de control sobre su tiempo, lo que puede mejorar la imagen corporativa y el rendimiento general.

Las desventajas pueden incluir:

- **Impacto en la comunicación:** la independencia que proporciona un horario flexible puede reducir la interacción con el equipo, lo que requiere estrategias efectivas para mantener la comunicación y la integración.
- **Desafíos organizacionales:** aunque los resultados deben entregarse puntualmente, la dependencia de un solo empleado puede tener consecuencias para todo el equipo. una organización deficiente puede llevar a la falta de responsabilidad y resultados negativos.
- **Extensión del tiempo laboral:** la flexibilidad puede generar malentendidos sobre las expectativas laborales, haciendo que los empleados se sientan obligados a responder a solicitudes fuera del horario laboral normal.

Untis ofrece una solución integral para la gestión de horarios educativos, pero su implementación debe ser cuidadosamente evaluada para asegurar que se alinee con las necesidades específicas del centro.

Additio App

Additio App es una plataforma integral de gestión escolar que ofrece soluciones todo en uno para proyectos educativos. Presenta dos versiones principales, *Additio for Teachers* y *Additio for Schools*, esta plataforma ha sido adoptada por más de 200 000 docentes y 3 000 escuelas en sus clases (AppAdditio, 2014).

Las principales ventajas que podemos obtener al utilizar este *software* son:

- La plataforma automatiza procesos complejos como la evaluación, planificación y gestión escolar, lo que reduce el tiempo y esfuerzo requerido por los docentes.
- Facilita la realización de tareas administrativas y educativas, permitiendo a los docentes centrarse más en la enseñanza y menos en la burocracia.
- Ofrece una solución integral que reúne todas las herramientas necesarias en una sola plataforma, lo que mejora la cohesión y la comunicación dentro del centro educativo.
- Implementa medidas estrictas para proteger la información personal y confidencial de los estudiantes y docentes.

Desventajas:

- Requiere acceso a internet y dispositivos tecnológicos, lo que puede ser un obstáculo en áreas con infraestructura limitada o en escuelas con recursos reducidos.
- Puede haber costos asociados con la adopción y mantenimiento de la plataforma, lo que podría ser un factor limitante para algunas instituciones educativas.
- Los usuarios pueden necesitar tiempo para familiarizarse con todas las funciones de la plataforma, lo que podría afectar temporalmente la eficiencia.

El Generador de horarios con Inteligencia artificial (IA) es una herramienta clave dentro de *Additio App* que optimiza la asignación de clases, profesorado y espacios. Utiliza inteligencia artificial para crear horarios rápidamente y realizar cambios en tiempo real, entre sus funcionalidades destacadas están:

- Genera horarios rápidamente, eliminando el tiempo invertido en cuadrarlos manualmente.
- Permite realizar cambios y ajustes en tiempo real.
- Permite a múltiples usuarios trabajar simultáneamente en los horarios, manteniendo la sincronización en tiempo real.
- Se integra con otras plataformas a través de una Application Programming Interface (API, por sus siglas en inglés), permitiendo la importación y exportación de datos.

Additio App es una solución integral que mejora la gestión escolar, aumenta la productividad y facilita la comunicación entre docentes, estudiantes y familias. Su capacidad para adaptarse a las necesidades específicas y su facilidad de uso lo convierten en una opción valiosa para cualquier institución educativa.

El análisis de los sistemas y módulos mencionados revela que tienen varias limitaciones que los hacen inadecuados para la aplicación deseada en la investigación. Entre las principales deficiencias, se destaca que suelen ser *software* de propiedad. Aunque los sistemas internacionales destacan por su capacidad para generar automáticamente el horario docente, esta ventaja no es suficiente para superar los demás inconvenientes que presentan. Sin embargo, GHC ofrece una funcionalidad que puede tomarse como referencia en la implementación del sistema a desarrollar como la configuración del marco horario.

Donde se establece:

- El número de turnos.
- La parada de mediodía.
- La hora de entrada y de salida de cada uno de los turnos.
- El marco horario podrá ser modificado durante la configuración del horario.

A continuación, se expondrá una tabla comparativa de las principales características de los sistemas y módulos de gestión de horarios internacionalmente.

Tabla 1.3. Resumen comparativo de sistemas y módulos internacionales. Fuente: Elaboración propia

Característica	Untis	GHC	Additio App
Licencia	Propietario	Propietario	Propietario
Automatización	Sí	Sí	Sí
Impacto en docentes	No	Sí	No
Adaptabilidad	Sí	No	Sí
Generación automática de horarios	Sí	Sí	Sí
Técnicas de asignación y optimización	Algoritmos genéticos	Optimización de tiempo de ejecución	Optimización de tiempo de ejecución de acuerdo con restricciones y preferencias específicas.

Luego del estudio realizado, se puede concluir que estos *softwares*, Untis, GHC, y *Additio App*, han sido de gran ayuda en la búsqueda de soluciones para la gestión del horario docente en la FTI. Sin embargo, debido a que son *software* propietario y requieren una licencia de pago, no pueden ser utilizados en su totalidad como solución al problema existente. A pesar de esto, es importante destacar que todos estos *softwares* comparten características clave que son relevantes para el sistema a desarrollar. Todos ofrecen la generación automática del horario, lo cual es una funcionalidad fundamental para optimizar el proceso de asignación de clases y actividades académicas.

Además, utilizan técnicas de optimización, como algoritmos genéticos para buscar soluciones óptimas. Estas técnicas permiten abordar los desafíos de la programación de horarios, como evitar conflictos, optimizar

recursos y maximizar la eficiencia. Se tomará como referencia para implementar el sistema para la gestión del horario docente de la Facultad de Tecnologías Interactivas los algoritmos genéticos, en aras de emplear las mejores prácticas de estas técnicas para desarrollar una solución personalizada que se ajuste a las necesidades y restricciones de la facultad.

El estudio de los *softwares* anteriormente descritos demostró que la planificación de un horario docente gira en torno al uso de métodos que permitan optimizar los horarios, teniendo en cuenta diferentes criterios, como la distribución equitativa de las asignaturas, la disponibilidad de recursos, la minimización de conflictos de horarios, la maximización de la utilización de aulas y la satisfacción de profesores y estudiantes. La aplicación de algoritmos genéticos asegura que los horarios generados sean los más adecuados y satisfactorios posibles.

1.6. Herramientas y tecnologías

Las herramientas y tecnologías en el proceso de desarrollo de *software* son fundamentales al representar los elementos clave del proceso. Dichas herramientas también se relacionan con descripciones que ayudan a los involucrados a comprender las acciones y tareas necesarias para llevar a cabo el trabajo. Su adecuada selección puede mejorar la calidad y eficiencia del desarrollo, mientras que las tecnologías apropiadas garantizan que la solución cumpla con los requisitos y desafíos específicos del proyecto (Pressman, 2006). En los subepígrafes siguientes se realizará la descripción de estas, explicando el porqué de su selección.

1.6.1. Lenguaje de modelado

El Lenguaje de Modelado Unificado (UML) es una herramienta estándar utilizada para representar visualmente sistemas y procesos mediante diagramas. Su principal aplicación es en la ingeniería de *software*, pero también se utiliza en áreas como la gestión de proyectos y la modelación de procesos de negocio. Desarrollado por la *Object Management Group* (*OMG*, por sus siglas en inglés), el UML permite a los profesionales del desarrollo de sistemas representar la estructura, el comportamiento y la interacción de los componentes de un *software*.

El UML facilita la comunicación entre equipos y mejora la comprensión de sistemas complejos, a través de diagramas estructurales y conductuales, como los de clases, casos de uso y secuencias. Su adopción global lo ha convertido en un estándar clave para el diseño y documentación de *software* y otros sistemas. Además, es un lenguaje de modelado estandarizado que permite representar sistemas, procesos y relaciones mediante diagramas gráficos. Su versatilidad lo convierte en una opción popular en diversos dominios, como el desarrollo de *software*, la arquitectura de sistemas y la gestión de proyectos. Respaldado por la *OMG*, el UML goza de reconocimiento global y se ha convertido en un estándar de facto (Lamarca, 2016).

Ofrece una variedad de diagramas, cada uno con un propósito específico:

- Diagramas de casos de uso: estos describen las interacciones entre actores (usuarios, sistemas externos) y el sistema en estudio. Son fundamentales para definir los requisitos funcionales y comprender cómo los usuarios interactúan con la aplicación.
- Diagramas de clases: representan las clases, sus atributos y relaciones. Son ideales para diseñar la estructura del *software* y modelar la herencia, la agregación y la asociación entre objetos.
- Diagramas de secuencia: capturan la secuencia temporal de interacciones entre objetos. Son útiles para visualizar procesos complejos y escenarios de ejecución.
- Diagramas de actividades: modelan flujos de trabajo, procesos y algoritmos. Son esenciales para la planificación y el diseño detallado.

El UML no sólo es valioso en la etapa de planificación, sino también durante la implementación y el mantenimiento del *software*. Algunas de sus aplicaciones clave son:

- Documentación: los diagramas UML sirven como documentación visual para equipos de desarrollo y partes interesadas. Facilitan la comprensión y la comunicación.
- Diseño detallado: al modelar clases, relaciones y comportamientos, el UML guía el diseño de sistemas coherentes y bien estructurados.
- Análisis de requisitos: los diagramas de casos de uso ayudan a definir los requisitos funcionales y a alinear las expectativas de los usuarios.
- Optimización de procesos: los diagramas de actividad permiten identificar cuellos de botella y mejorar la eficiencia.

1.6.2. Herramienta CASE

La elección de las herramientas correctas en el desarrollo de *software* puede transformar profundamente el rumbo de un proyecto. En este escenario, las herramientas Computer Aided Software Engineering (CASE, por sus siglas en inglés) emergen como una solución factible, permitiendo abordar con éxito las tareas de diseño, análisis y gestión en proyectos de desarrollo de *software* (Pressman, 2006).

Visual Paradigm 8.0

Visual Paradigm for UML 8.0 Enterprise Edition es una herramienta de modelado y diseño de *software* que permite crear representaciones gráficas de alta calidad utilizando el UML y otras notaciones estándar. Su interfaz sencilla e intuitiva facilita la creación de diagramas UML, siguiendo los estándares de la industria. Pero ¿qué hace que *Visual Paradigm* destaque entre la multitud? Con *Visual Paradigm* se puede diseñar la arquitectura, los componentes, los procesos, los casos de uso, las clases, los objetos, las secuencias, las actividades, los estados y otros elementos del *software* mediante diagramas gráficos interactivos. La herramienta ofrece una amplia gama de símbolos y elementos predefinidos, lo que facilita la creación de modelos

precisos y detallados (VisualParadigm, 2024).

Una de las características más valiosas de *Visual Paradigm* es su capacidad para generar automáticamente código fuente en diferentes lenguajes de programación a partir de los modelos UML. Esto agiliza el proceso de desarrollo y reduce la posibilidad de errores manuales.

Visual Paradigm está disponible para *Windows*, *macOS* y *Linux*, lo que brinda flexibilidad en cuanto a la elección de la plataforma de ejecución. Los equipos de desarrollo pueden trabajar en su entorno preferido sin restricciones, lo que fomenta la colaboración y la productividad.

Visual Paradigm ofrece funciones avanzadas que potencian la eficiencia y la calidad del desarrollo. Algunas de estas características incluyen:

- Simulación de procesos de negocio: permite evaluar el comportamiento de los procesos antes de implementarlos, identificando posibles problemas y optimizando flujos de trabajo.
- Análisis de requisitos: facilita la captura y gestión de requisitos, asegurando que el *software* cumpla con las expectativas de los usuarios y las partes interesadas.
- Diseño de interfaz de usuario: proporciona herramientas para crear prototipos de interfaces de usuario, mejorando la experiencia del usuario final.
- Control de versiones: *Visual Paradigm* integra sistemas de control de versiones, permitiendo un seguimiento preciso de los cambios en el código y los modelos.
- Colaboración en equipo: facilita la colaboración entre miembros del equipo, incluso en proyectos distribuidos.
- Generación de documentación: automatiza la creación de documentación técnica y manuales de usuario.

Visual Paradigm es una herramienta completa que combina potencia, flexibilidad y funcionalidades avanzadas. Su adopción puede elevar la calidad de los proyectos de desarrollo de *software* como el de IAHorDo y contribuir al éxito de las organizaciones que buscan innovar y crear soluciones efectivas.

1.6.3. Entorno de desarrollo

Visual Studio Code

Visual Studio Code, desarrollado por *Microsoft*, es un Entorno de Desarrollo Integrado (IDE) de código abierto que ofrece un espacio de trabajo inteligente para programadores y desarrolladores. Su popularidad se debe a una combinación de peculiaridades fundamentales y una interfaz amigable. El corazón de *VS Code* es su editor de código. Con resaltado de sintaxis, autocompletado y sugerencias inteligentes, escribir código se convierte en una experiencia fluida. Además, las características de edición avanzadas, como la refactorización y la búsqueda rápida, hacen que la programación sea más ágil. La herramienta admite una

amplia gama de extensiones y complementos creados por la comunidad. ¿Necesitas soporte para un lenguaje específico o una funcionalidad personalizada? Existe una extensión para eso. Esta flexibilidad permite a los desarrolladores adaptar su entorno según sus necesidades. Está disponible para *Windows*, *macOS* y *Linux*; la accesibilidad es clave para una experiencia de desarrollo sin obstáculos. Aunque *VS Code* es conocido por su afinidad con lenguajes *web* como *JavaScript*, Lenguaje de Marcado de Hipertexto (HTML) y Hojas de Estilo en Cascada (CSS), su versatilidad va más allá. Admite, además, una variedad de lenguajes de programación, desde *Python* hasta *Java* (VisualStudioCode, 2024).

1.6.4. Lenguaje de programación

Un lenguaje de programación es una herramienta creada para transmitir instrucciones a un ordenador, permitiendo así la creación de programas que controlan tanto el comportamiento físico como lógico de una máquina. Estos lenguajes son fundamentales para expresar algoritmos con precisión y también sirven como medio de comunicación entre humanos y máquinas. A través de ellos, se puede especificar con detalle cómo debe manejarse la información, cómo debe almacenarse o transmitirse, y qué acciones deben tomarse según diversas condiciones posibles. Estos lenguajes intentan acercarse al lenguaje humano o natural, facilitando la interacción con las máquinas (Suárez; Guerrero y Castrillón, 2013). Según Roque Cuevas, el conjunto de órdenes escritas por el programador, conocido como lenguaje fuente, se traduce posteriormente al lenguaje máquina que entiende el ordenador. Cada lenguaje de programación cuenta con una sintaxis y reglas específicas que definen su estructura (Contributors, 2024).

PHP

PHP es un popular lenguaje multiplataforma, interpretado y de secuencias de comandos de propósito general que es especialmente adecuado para el desarrollo *web* dinámico. Rápido, flexible y pragmático, PHP potencia todo, desde su *blog* hasta los sitios *web* más populares del mundo. PHP se centra principalmente en las secuencias de comandos del lado del servidor *server-side scripting*, por lo que puede hacer cualquier cosa que pueda hacer cualquier otro programa CGI, como recopilar datos de formulario, generar contenido de página dinámico o enviar y recibir *cookies*, maneja un enorme volumen de módulos llamados *ext's*, y puede establecer conexión con una alta variedad de motores de base de datos actualmente como *MySQL* y *PostgreSQL* (Suárez; Guerrero y Castrillón, 2013).

Python

Python es un lenguaje de programación interpretado de alto nivel que se destaca por su legibilidad y facilidad de uso. Su sintaxis clara y sencilla facilita la escritura y comprensión del código, esto lo convierte en una excelente opción tanto para principiantes como para desarrolladores experimentados. Con el paso de los años se ha convertido en un lenguaje omnipresente en la industria tecnológica pues se utiliza en diversas

áreas como el desarrollo *web*, inteligencia artificial, análisis de datos y automatización de tareas debido a su gran cantidad de bibliotecas y *frameworks*. Estas herramientas predefinidas permiten a los desarrolladores acceder a funcionalidades listas para usar. Este lenguaje permite la reutilización de módulos en diferentes programas, los cuales cubren tareas comunes como manejo de archivos, matemáticas y comunicación en red, esto ahorra tiempo en el desarrollo y reduce la necesidad de escribir código desde cero. Además, cuenta con una comunidad activa de desarrolladores que contribuyen con documentación, tutoriales, librerías adicionales y foros de discusión.

Python es compatible con los principales sistemas operativos como *Windows*, *macOS* y *Linux*, lo que facilita la portabilidad y la colaboración entre equipos. También se puede integrar fácilmente con otros lenguajes y tecnologías, permitiendo su empleo como lenguaje de *scripting* en aplicaciones más grandes y la combinación con bibliotecas de otras tecnologías (Python, 2024).

JavaScript

JavaScript es un lenguaje de programación utilizado en el desarrollo *web* que se ejecuta en el navegador del usuario. Se caracteriza por su versatilidad y capacidad para crear aplicaciones *web* interactivas y dinámicas. Mediante este, los desarrolladores pueden manipular el contenido y la estructura de una página *web* en tiempo real, gestionar eventos, realizar validación de datos en el lado del cliente y establecer comunicación con el servidor. Su flexibilidad y capacidad para interactuar con otros elementos de la *web* lo convierten en una herramienta poderosa para los programadores (JavaScripts, 2024).

HTML5

HTML es el lenguaje fundamental utilizado en el desarrollo *web* para crear el esquema y el contenido de una página *web*. Se utiliza para definir elementos y etiquetas que representan textos, imágenes, enlaces, formularios y otros elementos que componen una página *web*. Cada elemento HTML se organiza jerárquicamente en un árbol de documentos, lo que permite definir la estructura de la página. Además, permite la inclusión de metadatos, como títulos, descripciones y enlaces a hojas de estilo y *scripts*. HTML5 es la quinta versión del HTML. Es conocido por sus mejoras en la semántica, la accesibilidad y la capacidad de crear aplicaciones *web* más interactivas. HTML5 admite nuevas etiquetas y atributos que facilitan la creación de contenido multimedia y aplicaciones *web* dinámicas (HTML5, 2019).

CSS3

CSS3 es un lenguaje de diseño utilizado en el desarrollo de aplicaciones *web* para controlar la apariencia y presentación de las páginas. Permite a los desarrolladores definir cómo se verán los elementos HTML en un navegador. Mediante reglas de estilo, se especifican propiedades como colores, fuentes, márgenes,

espaciado y diseño de página, lo que permite lograr una estética coherente y atractiva en un sitio *web*. CSS es crucial para la separación de contenido y presentación, adecuando la creación de sitios *web* responsivos y compatibles con múltiples dispositivos.

CSS3 es la tercera versión del lenguaje de CSS. Es conocido por sus mejoras en la estilización y la capacidad de crear diseños más complejos y personalizados. CSS3 admite nuevas propiedades y selectores que facilitan la creación de diseños *web* más atractivos y responsivos (McFarland, 2012).

Luego del estudio de los lenguajes de programación expuestos y siguiendo los requerimientos con que debe cumplir el sistema se opta por el uso de la programación *web*, usando *Python* como lenguaje de programación para la implementación del algoritmo de optimización teniendo en cuenta su versatilidad, facilidad de uso y capacidades integradas para manipular datos y realizar cálculos matemáticos. Esto lo convierte en una herramienta efectiva para abordar problemas complejos, como la planificación de horarios docentes debido a la capacidad que posee de integración con bases de datos y sistemas *web*, además de ser de código abierto y altamente flexible en la programación. También trabaja en conjunto con HTML5 y CSS3 para mejorar el trabajo en entornos *web*, permitiendo crear interfaces amigables para los usuarios del sistema.

1.7. Marco de trabajo para el desarrollo de la solución informática

Django

Django es un popular *framework* de desarrollo, de código abierto que se caracteriza por su alto nivel de productividad y seguridad. Está escrito en *Python* y ofrece una amplia gama de herramientas y bibliotecas que simplifican el proceso de creación de aplicaciones *web* robustas. Se destaca por su enfoque en el principio "baterías incluidas", lo que significa que proporciona una serie de componentes listos para usar, como autenticación de usuarios, administración de bases de datos y enrutamiento de *Uniform Resource Locator* (URL, por sus siglas en inglés). Esto acelera el desarrollo y permite a los desarrolladores centrarse en la lógica de la aplicación en lugar de preocuparse por detalles técnicos. Además, promueve prácticas de seguridad sólidas, lo que lo convierte en una elección confiable para aplicaciones que manejan datos sensibles, como las soluciones *web* para la planificación y generación de horarios docentes (Django, 2024).

Bootstrap 5

Bootstrap 5 es un *framework* de CSS y *JavaScript* que se utiliza para crear aplicaciones *web* con diseños responsivos y atractivos. Es conocido por su facilidad de uso y su capacidad para crear diseños *web* complejos con un mínimo de código. *Bootstrap 5* admite una amplia variedad de componentes y utilidades que facilitan la creación de aplicaciones *web* (A. Cruz, 2023). Son varias las ventajas que nos ofrece *Bootstrap* a la hora de utilizarlo, como por ejemplo:

- Primer enfoque móvil, el marco consiste en los primeros estilos móviles en toda la biblioteca en lugar de ellos en archivos separados.
- Diseño *responsive*, CSS3 *responsive* de *Bootstrap* se ajusta a *Desktops*, *Tablets* y *Móviles*.
- Es compatible con todos los navegadores.
- Es de código abierto.

1.7.1. Gestor de base de datos

PostgreSQL 14.6

El gestor de base de datos seleccionado es *PostgreSQL 14.6*, Sistema de Gestión de Bases de Datos Relacional de Código Abierto (RDBMS) ampliamente utilizado. Es además, una elección popular para aplicaciones que requieren una base de datos relacionales confiables, extensibles y de alto rendimiento. Se adapta tanto a pequeñas bases de datos locales como a sistemas de datos empresariales a gran escala (Postgresql, 2024). Ofrece una serie de características, tales como:

- Consultas complejas: *PostgreSQL* ofrece un sistema de consulta avanzado que permite realizar consultas complejas y agregaciones de datos.
- Seguridad: proporciona funciones de seguridad, como autenticación y autorización avanzadas, lo que lo hace adecuado para aplicaciones que manejan datos sensibles.
- Replanteamiento: ofrece capacidades de replicación para garantizar la disponibilidad y escalabilidad de los datos.

1.8. Metodologías

Las metodologías de desarrollo de *software* constituyen un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones informáticas. Entre ellas, las metodologías ágiles y tradicionales representan enfoques diferentes en la gestión de proyectos. El enfoque ágil se distingue por su flexibilidad y capacidad de adaptación, dividiendo los proyectos en iteraciones manejables y fomentando una colaboración continua y una respuesta rápida a los cambios. Esto se traduce, por ejemplo, en la planificación de horarios docentes, permitiendo ajustar los horarios según las necesidades cambiantes de profesores y estudiantes. Por otro lado, el enfoque tradicional, también conocido como método en cascada, sigue un proceso lineal y secuencial con una planificación detallada desde el principio. Se espera que se siga el plan original de manera estricta, lo que puede ser desafiante ante cambios inesperados en la confección de horarios docentes.

Dentro de las metodologías ágiles se encuentran Proceso Unificado Ágil (AUP), Scrum y XP.

Scrum

Este método de desarrollo ágil de *software* fue concebido por Jeff Sutherland y su equipo a principios de los años 90. Los principios de Scrum son congruentes con el manifiesto ágil y se utilizan para guiar las actividades de desarrollo dentro de un proceso de análisis que incluye las siguientes actividades estructurales: requerimientos, análisis, diseño, evolución y entrega (Beck y Andres, 2004). Dentro de cada actividad estructural, las tareas del trabajo se organizan en un patrón de proceso llamado *sprint*. El trabajo realizado dentro de un *sprint* se adapta al problema en cuestión y se define y modifica con frecuencia en tiempo real por parte del equipo de desarrollo. Scrum enfatiza el uso de un conjunto de patrones efectivos para proyectos con plazos de entrega muy ajustados, requisitos cambiantes y negocios críticos. Cada uno de estos patrones de proceso define un grupo de acciones de desarrollo.

AUP

El Proceso Unificado Ágil adopta una filosofía secuencial para las actividades macro y un enfoque iterativo para las actividades micro a fin de construir sistemas basados en computadora. Ofrece una estructura secuencial (por ejemplo, una serie de actividades de ingeniería de *software* lineales) que permite al equipo visualizar el flujo general del proceso de un proyecto de *software*. Sin embargo, dentro de cada actividad, el equipo reitera con el objetivo de alcanzar la agilidad y entregar lo más rápido posible incrementos significativos de *software* a los usuarios finales. Cada iteración del AUP aborda actividades como modelado, implementación, pruebas, despliegue, configuración, gestión de proyectos y administración del entorno (Pressman, 2010).

XP

Esta es una metodología ágil de desarrollo de *software* que se centra en la entrega temprana y continua de *software* funcional de alta calidad. Fue creada por Kent Beck a fines de la década de 1990 y se ha convertido en una de las metodologías ágiles más populares y ampliamente utilizadas. XP enfatiza la simplicidad en el diseño y la toma de decisiones basada en pruebas y datos empíricos. La metodología se adapta bien a proyectos donde los requisitos pueden cambiar con frecuencia, pues su enfoque iterativo permite ajustes y adaptaciones continuas (Penadés y Letelier Torres, 2006).

A continuación, se presentan valores claves de esta metodología:

- Simplicidad: XP busca la solución más simple para cada problema, desarrollando solo las funciones necesarias en cada momento en lugar de posibles consideraciones futuras y evitando el trabajo anticipado.
- Comunicación: en XP, es esencial una comunicación directa y clara entre los ingenieros de *software*, los colaboradores en el desarrollo y los clientes. Esto facilita un entendimiento mutuo y un trabajo

colaborativo eficiente, utilizando metáforas ¹ para explicar conceptos importantes, fomentando la retroalimentación continua y evitando el uso de documentación extensa como medio de comunicación.

- Valentía: XP fomenta la valentía de los miembros del equipo para señalar errores y problemas de manera abierta, sin disculpas, con el fin de abordarlos de manera proactiva.
- Retroalimentación continua: XP se basa en ciclos cortos de desarrollo y presentación de avances al cliente, lo que permite recibir retroalimentación y corregir errores rápidamente.
- Respeto: al adherirse a estos valores, el equipo ágil fomenta el respeto entre sus miembros, así como entre otros participantes y los integrantes del equipo, lo que también se extiende de manera indirecta hacia el *software* mismo. A medida que logra entregar con éxito incrementos de *software*, el equipo cultiva un mayor respeto por el proceso.

Beneficios de XP

- Calidad del *software*: las prácticas de ingeniería, como las pruebas automatizadas y la programación en parejas, aseguran la entrega de *software* de alta calidad.
- Adaptabilidad: la capacidad de XP para adaptarse rápidamente a los cambios en los requisitos del cliente lo convierte en una metodología ideal para entornos dinámicos.
- Satisfacción del cliente: la estrecha colaboración y la entrega frecuente de *software* funcional mejoran significativamente la satisfacción del cliente.
- Productividad del equipo: la comunicación efectiva, la simplicidad y la comunicación continua aumentan la productividad del equipo de desarrollo.

El ciclo de vida óptimo de XP se compone de seis etapas:

Fase I Exploración: En esta etapa, los clientes presentan las historias de usuario relevantes para la primera entrega del producto. Simultáneamente, el equipo de desarrollo se familiariza con las herramientas y tecnologías que utilizará, probando la tecnología y explorando la arquitectura del sistema a través de un prototipo. Esta fase puede durar desde unas pocas semanas hasta varios meses, dependiendo del tamaño del proyecto y la experiencia del equipo con la tecnología.

Fase II Planificación de la entrega: Durante esta fase, el cliente prioriza las historias de usuario y los programadores estiman el esfuerzo necesario para cada una. Se acuerda el contenido de la primera entrega y se establece un cronograma conjunto, que no debería exceder los tres meses. Las estimaciones se realizan en puntos, donde un punto equivale a una semana ideal de programación. La planificación puede basarse en tiempo o alcance, utilizando la velocidad del proyecto para determinar cuántas historias se pueden implementar.

¹En el contexto de XP, una metáfora es una historia que cada cliente, programador o gerente narra, acerca de cómo funciona el sistema (Pressman, 2010). Martin Fowler en 2001 explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema.

Fase III Iteraciones: Esta fase incluye múltiples iteraciones antes de la entrega final. Cada iteración dura un máximo de tres semanas y busca establecer una arquitectura que pueda ser utilizada en el resto del proyecto. El cliente decide qué historias se implementarán en cada iteración, priorizando el valor comercial. Al final de la última iteración, el sistema está listo para producción.

Fase IV Producción: La fase de producción implica pruebas adicionales y revisiones de rendimiento antes de que el sistema sea entregado al cliente. También se toman decisiones sobre nuevas características a incluir, y es posible reducir el tiempo de cada iteración a una semana. Las sugerencias y propuestas son documentadas para su futura implementación.

Fase V Mantenimiento: Mientras la primera versión está en producción, el equipo debe mantener el sistema operativo y desarrollar nuevas iteraciones. Esto puede requerir soporte adicional al cliente y ajustes en la estructura del equipo, dado que la velocidad de desarrollo puede disminuir tras la entrega inicial.

Fase VI Muerte del proyecto: Esta fase ocurre cuando no hay más historias que agregar al sistema. Se asegura que se cumplan las necesidades del cliente en términos de rendimiento y confiabilidad. Se genera la documentación final y no se realizan más cambios arquitectónicos. La muerte del proyecto también puede suceder si el sistema no genera los beneficios esperados o si no hay presupuesto para su mantenimiento.

Esta metodología se adapta perfectamente al proceso de planificación y generación de horarios docentes, pues permite responder rápidamente a los cambios en los requisitos, involucrar estrechamente al cliente (en este caso, los responsables de la planificación y confección de horarios) y entregar de forma iterativa soluciones que se ajusten a sus necesidades cambiantes teniendo en cuenta que el plazo de entrega es corto y el tamaño del equipo pequeño.

Conclusiones parciales

En el presente capítulo, se ha analizado la planificación de horarios docentes, subrayando su importancia para optimizar la asignación de recursos y minimizar conflictos en el proceso formativo de los estudiantes. Esto permitió llegar a las siguientes conclusiones:

- El empleo de tecnologías y herramientas es fundamental para el desarrollo de un sistema rentable que facilite este proceso. Tecnologías como *Python*, debido a su legibilidad y versatilidad, y el *framework Django*, que permite crear aplicaciones *web* robustas, junto con *PostgreSQL*, que asegura un excelente manejo de los datos, son indispensables para este fin.
- El estudio de soluciones similares reveló que muchos sistemas existentes, aunque útiles, carecen de la flexibilidad necesaria para adaptarse a las particularidades del entorno educativo. Esta observación subraya la necesidad de una solución personalizada.

- Es esencial desarrollar un sistema personalizado que integre algoritmos genéticos para abordar las complejidades de la planificación horaria. Este enfoque no sólo busca mejorar la satisfacción de los docentes, sino también contribuir a una mejor gestión del tiempo y los recursos disponibles.
- La elección de la metodología XP se ajustó al ciclo de vida del proyecto, garantizando un desarrollo ágil y flexible, permitiendo adaptaciones rápidas a los cambios y mejorando continuamente la calidad del sistema.

Diseño de IAHorDo para la planificación y generación automática del horario en la FTI

En este capítulo, se profundiza el diseño del sistema orientado a la planificación y generación automática del horario docente de la Facultad de Tecnologías Interactivas, adoptando la metodología XP como pilar fundamental. Se detalla el proceso meticuloso llevado a cabo durante las fases de planificación y diseño, enfocándose específicamente en la elaboración y definición de las historias de usuario, así como en la configuración de un plan de iteraciones alineado con el cronograma de entrega del proyecto. Este enfoque asegura una gestión del desarrollo del sistema organizada. También, se ofrece una visión detallada de la arquitectura elegida, destacando la aplicación de patrones de diseño GRASP y GOF. Se aborda la creación de un modelo de datos que respalde las necesidades del sistema, garantizando así una integración fluida entre los componentes del *software*. Finalmente, se discute la adopción de la arquitectura Modelo Vista Template (MVT), un patrón de arquitectónico que facilita la separación de responsabilidades dentro del sistema y se examinan los requisitos no funcionales, aspectos cruciales que complementan la funcionalidad del *software*, asegurando su rendimiento óptimo y su adaptabilidad a futuras necesidades.

2.1. Propuesta de solución

La Facultad de Tecnologías Interactivas se encuentra en vanguardia en la digitalización de sus procesos administrativos, con el desarrollo de **IAHorDo**, un sistema para la planificación y generación automática del horario que busca optimizar la gestión académica. Esta herramienta informática, diseñada para ser accesible tanto para los PPA como para el vicedecano de formación y el decano, representa un avance significativo en la precisión de la planificación académica.

La funcionalidad central de IAHorDo reside en su capacidad para gestionar automáticamente la elaboración de horarios docentes, basándose en una variedad de datos críticos para el proceso, incluyendo sesiones, asignaturas, profesores, grupos, afectaciones de profesores, afectaciones de turnos, balance de carga, indis-

ponibilidad de locales, actividades y la carga docente. Este enfoque automatizado no sólo reduce la carga manual y el riesgo de errores, sino que también permite una mayor adaptación en la planificación de horarios, respondiendo a las necesidades cambiantes de la comunidad educativa. Una vez finalizado el horario, el sistema facilita su distribución a las partes interesadas, mejorando así la comunicación y la transparencia en el proceso. La implementación de dicha propuesta promete ofrecer múltiples ventajas y beneficios, no sólo en términos de eficiencia operativa, sino también en la satisfacción de los usuarios finales, incluidos estudiantes, profesores y la institución en su conjunto.

El diseño de sistemas automatizados para la generación de horarios, impulsados por algoritmos genéticos, presenta una oportunidad innovadora en las instituciones educativas. Este marco se centra en la incorporación y gestión efectiva de restricciones clave. Las restricciones se definen como las limitaciones y condiciones que deben ser cumplidas por el sistema y sus componentes durante el proceso de generación de horarios. La inclusión meticulosa de estas restricciones en el modelo permite al algoritmo genético tomar decisiones informadas enfocándose en la identificación de soluciones que satisfagan todas las exigencias establecidas. Este enfoque asegura que los horarios generados sean viables y equitativos, considerando los factores críticos. La fase inicial del proyecto implica la recolección y preparación de datos relevantes, que incluyen detalles sobre asignaturas, profesores, capacidades de locales, restricciones de horarios y preferencias individuales. Este proceso de preparación de datos es fundamental, pues sirve como base para el desarrollo del modelo que alimentará al algoritmo genético.

A continuación se muestra una tabla que resume las restricciones mencionadas y sus descripciones:

Tabla 2.1. Restricciones y sus descripciones. Fuente: Elaboración propia.

Restricción	Descripción
Sesiones	Define si cada sesión de clase se imparte en la mañana o la tarde.
Aulas fijas	Mantener aulas específicas fijas para ciertos bloques de tiempo o asignaturas. Por ejemplo: • Las conferencias se imparten en salones generalmente para dos grupos. • Las clases prácticas y talleres se imparten en aulas teóricas únicamente para un solo grupo. • Los laboratorios se imparten en laboratorios para un solo grupo.
Continúa en la siguiente página	

Restricción	Descripción
Frecuencia de asignaturas	Asegurar que las asignaturas sean impartidas según la frecuencia programada para el curso de acuerdo al balance de carga de cada semana. Además, de evitar solapamientos.
Coincidencia de grupos	Evitar que dos grupos sean atendidos por el mismo profesor en diferentes aulas al mismo tiempo.
Coincidencia de locales	Evitar que dos grupos coincidan en la misma aula, a menos que sea un salón específico.
Horario de almuerzo	No se deben programar el tercer y cuarto turno de clases en el mismo día para los mismos profesores o grupos, respetando el horario de almuerzo.
Afectaciones personales	Considerar las preferencias personales de los profesores sobre los turnos en los que prefieren impartir sus clases.
Exámenes	Mantener los exámenes en aulas diferentes durante todo el año, en el mismo horario.

Esta tabla proporciona una visión clara de las restricciones claves que deben ser consideradas en el diseño y optimización del sistema a desarrollar. Cada restricción juega un papel crucial en la búsqueda de una solución que satisfaga las necesidades y expectativas de todos los involucrados en el proceso educativo.

2.1.1. Secciones de IAHorDo

El sistema contará con 11 secciones que brindarán soluciones integrales al problema planteado. Cada sección cumple una función específica y se complementa con los demás para lograr un sistema efectivo.



Figura 2.1. Representación de las 11 secciones existentes en IAHorDo

1. Sección de Marco horario: En esta sección de configuración del marco horario muestra una ventana se establecen: los elementos fundamentales como el tipo de curso, el año académico, el número de turnos por sección, la duración en minutos de cada uno de los turnos, tiempo entre los turnos y parada de mediodía. En la plantilla general puede indicar las posiciones sin actividad en la facultad, que quedarán inhabilitadas en todas las plantillas particulares. El marco horario podrá ser modificado durante la configuración del horario.
2. Sección de Balance de carga : El profesor principal del año y la vicedecana utilizarán este módulo para distribuir equitativamente las clases y la carga de trabajo entre los profesores. Por ejemplo, se podrá registrar que la asignatura tiene una frecuencia de 3 clases semanales, con 2 clases teóricas y 1 práctica, impartidas durante las semanas 1 a 16 del semestre.
3. Sección de Carga docente: Se establece la relación entre profesores, asignaturas y grupos, gestionando la carga docente de cada uno. Esto garantiza que se asignen las cargas docentes de manera equilibrada y acorde a las capacidades y disponibilidades de los profesores.
4. Sección de Afectaciones: Se registran las afectaciones de cada profesor y de los turnos involucrados, incluyendo el profesor afectado, día, semana de la afectación y turno. Esto permite considerar las ausencias o disponibilidades limitadas de los profesores y los turnos en la generación de horarios.
5. Sección de Usuarios : Se mantiene un registro de todos los usuarios que interactúan con el sistema teniendo en cuenta el rol que desempeñan, para el rol de estudiante, se registran los siguientes datos: su nombre, apellidos, nombre de usuario y el año académico al que pertenecen, para el rol de profesor

se registra: nombre, apellidos, nombre de usuario, categoría docente, formación académica, residente, para el rol de planificador se incluyen: nombre, apellidos, nombre de usuario, año académico y facultad a la que pertenece, en el rol de decano se tiene en cuenta: nombre, apellidos, nombre de usuario y facultad. Esta información es esencial para la asignación de cargas docentes y la generación de horarios que respeten las disponibilidades y preferencias de los profesores.

6. Sección de Asignaturas: Esta sección permite al profesor principal del año y a la vicedecana gestionar la información de las asignaturas que se impartirán en la facultad. Por ejemplo, se podrán registrar datos como el nombre de la asignatura, la sesión en la que se imparte (mañana o tarde), el año académico y el semestre correspondiente.
7. Sección de Locales: Se administran los datos de los locales disponibles en la facultad, como el nombre Aula 201 y la disponibilidad de cada uno. Esto permite optimizar la asignación de espacios físicos para las clases.
8. Sección de Grupos: Esta sección gestiona la información de los grupos existentes en la facultad, como el nombre IDFI401 o IEFI401 y el año académico al que pertenecen. Contar con esta información centralizada facilita la asignación de asignaturas, profesores y aulas a los grupos correspondientes.
9. Sección de Actividades: Se registran los datos de los actividades incluyéndose además, los exámenes planificados para cada asignatura, incluyendo el día, turno y semana correspondiente. Esto facilita la coordinación y planificación de las actividades y los exámenes dentro del horario docente.
10. Sección de Horario: Esta sección permite generar automáticamente el horario docente, considerando las restricciones y preferencias definidas, además de visualizarlo una vez generado, proceder a la edición del mismo en cuestión de mantener una higiene horaria y el filtrado de horarios por grupo, año y tipo de curso. Esto optimiza el proceso de generación de horarios, reduciendo significativamente el tiempo y esfuerzo requeridos. La integración y el flujo de información entre estos módulos es crucial para lograr una solución integral que satisfaga las necesidades de la institución. Además, de visualizar el horario docente generado, dando la opción de filtrar por grupo, año y semana. Esto brinda a los usuarios una interfaz clara y fácil de usar para consultar los horarios.

2.1.2. Asignación de variables

La gestión de horarios docentes en la FTI requiere una comprensión profunda de cómo asignar variables de manera estratégica. La asignación de variables es un componente crítico en la formulación de soluciones óptimas, pues estas variables representan los elementos fundamentales que se manipulan para alcanzar un horario docente equilibrado. En este contexto, las variables clave incluyen:

- Grupos de estudiantes (A): Representan los diferentes niveles o grupos de estudiantes que recibirán instrucción.
- Locales disponibles (B): Refieren a los espacios físicos designados para la enseñanza, como salones de conferencia, aulas teóricas y laboratorios.
- Asignaturas (C): Indican el contenido específico que se impartirá a los estudiantes.
- Profesores (D): Son los instructores responsables de enseñar las asignaturas.
- Sesiones académicas (E): Se refieren al período específico del año en el que se imparten las clases.
- Días de la semana (F): Determinan cuándo se llevarán a cabo las clases.
- Turnos (G): Definen el momento del día en el que se impartirán las clases.
- Afectaciones (G): Definen el momento del día en el que se manifieste una afectación tanto para profesores como para los turnos de clases.

La asignación cuidadosa de estas variables permite explorar una amplia gama de combinaciones posibles, evaluando cada opción contra un conjunto de restricciones y preferencias específicas. Entre estas restricciones se encuentran la necesidad de evitar solapamientos de grupos, maximizar el uso de aulas, y asegurar una distribución justa de las asignaturas entre los profesores. Además, se deben considerar factores como las sesiones, las aulas fijas, la frecuencia de las asignaturas, la coincidencia de grupos, y las afectaciones de los profesores. Una asignación de variables exitosa tiene el potencial de generar horarios docentes que no sólo cumplan con las restricciones establecidas, sino que también optimicen varios objetivos importantes, como reducir los conflictos de horario, aumentar la eficiencia en el uso de recursos, y garantizar una distribución equitativa de las cargas de trabajo entre los profesores. Por otro lado, una asignación inapropiada o insuficiente de variables puede dar lugar a soluciones subóptimas o incluso inviables, lo que subraya la importancia de un enfoque meticuloso y estratégico en esta etapa crucial del proceso de planificación de horarios docentes.

2.2. Algoritmo genético para la generación de horarios docentes

Una vez que se han establecido las restricciones que deben considerarse para la generación de horarios, se da inicio a la primera fase del algoritmo genético, paso fundamental para abordar el problema de la generación de horarios docente, para una mejor comprensión se muestra el ciclo de la evolución del AG (Reyes; Hugo; Galaviz; Carlos; Armenta y José, 2012).

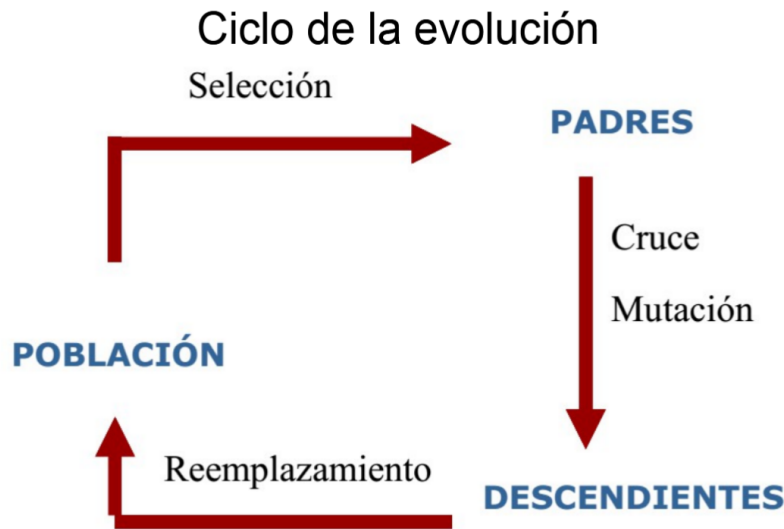


Figura 2.2. Ciclo de la evolución del algoritmo genético.

Paso 1: Definición de la población inicial

En la fase inicial, se establece la población de horarios docentes como punto de partida para el algoritmo genético. La generación de la población inicial se realiza de manera aleatoria, asignando asignaturas, profesores, aulas y horarios de forma que cada horario generado cumpla con las restricciones básicas, como la capacidad de las aulas y la disponibilidad de profesores. Para ello se importan varias librerías esenciales, como *random* para generar valores aleatorios, *deap* para definir la estructura del algoritmo genético, *datetime* para manejar fechas y tiempos, y *pandas* para la manipulación de datos (Suárez; Guerrero y Castrillón, 2013). Las librerías específicas del proyecto, (*administrador.models*, *planificador.models*, *planificador.utils*) se utilizan para interactuar con los modelos y utilidades definidos en la aplicación. El número 7 en la línea `population = self.toolbox.population(n=7)` se refiere al tamaño de la población inicial. Este valor se elige para proporcionar una diversidad suficiente en los horarios iniciales, pero también se puede ajustar según las necesidades específicas. Usar un número menor podría limitar la diversidad y, por lo tanto, la efectividad del algoritmo genético, mientras que un número mayor podría incrementar el tiempo de procesamiento sin necesariamente mejorar los resultados. La *toolbox* es una parte fundamental del código y se configura para manejar la generación y manipulación de individuos (horarios). Se registran métodos para generar atributos (`attr_turno`), individuos (`individual`) y la población inicial (`population`). Además, se registran operadores genéticos como evaluación (`evaluate`), cruce (`mate`), mutación (`mutate`) y selección (`select`) (Abramson, 1991).

Código fuente 2.1. Ejemplo de código en Python

```
1 print('definiendo clase')
2 % Clase para generar turnos y horarios con parametros
3 class GeneradorHorario:
```

```
4     def __init__(self, periodo:int, grupo:pl_models.Grupo,cant_turnos:int,lunes:
      datetime,semana:int):
5         self.periodo = periodo
6         self.grupo = grupo
7         self.cant_turnos = cant_turnos
8         self.lunes = lunes
```

Código fuente 2.2. Ejemplo de código en Python

```
1
2     print('[] Encontrar el mejor horario generado')
3     best_ind = tools.selBest(population, 1)[0]
```

Paso 2: Evaluación de aptitud *fitness*

En esta etapa, se evalúa la calidad de los horarios generados en la población inicial basándose en varios criterios que determinan su adecuación en términos de restricciones y preferencias. La puntuación de aptitud, o penalización, se acumula con cada incumplimiento de restricciones establecidas. La matriz de horario representa el horario semanal, mientras que el balance asegura la correcta distribución de sesiones. Se registra la cantidad de turnos asignados por asignatura en ‘cant_turnos_x_asign’ y se controlan las asignaturas impartidas cada día con ‘asignaturas_dia’. Las penalizaciones se aplican por varias razones: indisponibilidad de locales o profesores, asignaciones de sesiones incorrectas (mañana/tarde), desajustes en el balance de carga y tipología de la clase, superposiciones de locales, y días sin asignaturas. Por ejemplo, una penalización de 100 se aplica por desajustes en la tipología de clases, una penalización de 50 por falta de balance de carga, y una penalización significativa de 10 000 por turnos libres, desincentivando fuertemente días sin asignaturas.

El código inicia estableciendo la penalización en cero y convirtiendo el horario (individuo) en una matriz semanal. Se crea un diccionario para guardar el balance de carga de cada asignatura y, para cada día y turno, se verifica si el turno tiene información de local, profesor y asignatura. Si es así, se suman penalizaciones por indisponibilidad del local o profesor, actividades de grupo programadas, y sesiones incorrectas. Se verifica el balance de carga para asegurar la correcta distribución de clases y su tipología, sumando penalizaciones de 100 por desajuste y 50 si no hay balance de carga. La superposición de locales con horarios anteriores se evita sumando penalizaciones proporcionales al número de coincidencias. La variable ‘asignaturas_dia’ registra las asignaturas impartidas cada día, penalizando con 10 000 los días sin asignaturas. La caja de herramientas (*toolbox*) facilita la creación y manipulación de individuos (horarios) en la población inicial, configurando métodos para generar atributos, individuos y la población, además de operadores genéticos para evaluar, cruzar, mutar y seleccionar individuos. Usar números como 100 o 10 000 permite ajustar la severidad de las penalizaciones. Valores menores podrían no ser suficientemente disuasorios para evitar soluciones subóptimas, mientras que valores mayores aseguran el cumplimiento estricto de restricciones, guiando el algoritmo genético hacia horarios más viables y eficientes (Abramson, 1991).

Código fuente 2.3. Ejemplo de código en Python

```
1  def eval_schedule(self, individual):
2      penalizacion = 0
3      horario = [[individual[i + j * 5] for i in range(5)] for j in range(self.
4                  cant_turnos)]
5      balance = {aqt.id: pl_models.BalanceCarga.objects.filter(asignatura=aqt).
6                  order_by('frecuencia') for aqt in pl_models.Asignatura.objects.filter(
7                      anno_academico=self.grupo.anno_academico, periodo=self.periodo)}
8
9      cant_turnos_x_asign = {}
10     asignaturas_dia = [set() for _ in range(5)] %Conjunto para llevar el
11         control de asignaturas por dia
12
13     for dia in range(5):
14         fecha = utils.sumar_dias(fecha=self.lunes, dias=dia)
15         for turno in range(self.cant_turnos):
16             if isinstance(horario[turno][dia]['turno'], dict):
17                 local = horario[turno][dia]['turno'].get('local')
18                 profesor = horario[turno][dia]['turno'].get('profesor')
19                 selected_asignatura = horario[turno][dia]['turno'].get('
20                     asignatura')
21
22             %Penalizaciones
23             penalizacion += pl_models.Indisponibilidad_Local.objects.filter
24                 (dia=fecha, turno=turno, local=local).count()
25             penalizacion += pl_models.Afectacion_Profesor.objects.filter(
26                 dia=fecha, turno=turno, profesor_id=profesor).count()
27             actividades = pl_models.Actividad.objects.filter(dia=fecha)
28             actividad_id_grupo = pl_models.Grupos_Actividad.objects.filter(
29                 grupo=self.grupo, actividad_id__in=actividades).values_list(
30                     'actividad_id', flat=True)
31             penalizacion += pl_models.Turnos_Actividad.objects.filter(
32                 actividad_id__in=actividad_id_grupo, turno=turno).exists()
33
34             %Verificacion de sesion
35             sesion_correcta = 'Tarde' if turno > self.cant_turnos / 2 else
36                 'Manana'
37             penalizacion += (selected_asignatura.sesion != sesion_correcta)
38
39             %Balance de carga
40             asignatura_id = selected_asignatura.id
41             if asignatura_id in balance:
42                 mh = list(balance[asignatura_id].values_list('tipologia',
43                     flat=True))
44             cant_turnos_x_asign[asignatura_id] = cant_turnos_x_asign.
45                 get(asignatura_id, 0) + 1
```

```
33
34         %Verificacion de tipologia
35         pos = cant_turnos_x_asign[asignatura_id]%len(mh)
36         penalizacion += (mh[pos] != horario[turno][dia]['turno']).
            get('tipo_clase')) * 100
37     else:
38         %Penalizacion por no tener balance de carga
39         penalizacion += 50 %Ajusta esta penalizacion segun sea
            necesario
40
41     %Superposicion de local
42     ultimos_horarios = pl_models.Horario.objects.filter(semana=self
        .semana)
43     for uh in ultimos_horarios:
44         coincidencias_locales = pl_models.Turno.objects.filter(
            horario_id=uh.id, dia=fecha, turno=turno, local=local)
45         if coincidencias_locales.exists():
46             penalizacion += len(coincidencias_locales.exclude(
                asignatura=selected_asignatura, profesor_id=profesor
            ))
47             penalizacion += len(coincidencias_locales.filter(
                asignatura=selected_asignatura, profesor_id=profesor
            ))
48
49     %Registrar asignatura en el día
50     asignaturas_dia[dia].add(asignatura_id)
51
52     else:
53         %Penalizar solo si no hay asignaturas programadas para ese dia
54         if len(asignaturas_dia[dia]) < 1: %No se ha asignado ninguna
            asignatura
55             penalizacion += 10000 %Penalizacion por turno libre
```

Paso 3: Selección

En esta etapa, se implementará el operador de selección para elegir horarios de la población actual en función de su aptitud. Se utiliza un método de selección por torneo, donde los horarios con una mayor puntuación de aptitud tienen una mayor probabilidad de ser seleccionados. Este operador se utiliza por su simplicidad y efectividad, permitiendo mantener diversidad genética y evitando la convergencia prematura a soluciones subóptimas. La función ‘generar_turno’, selecciona aleatoriamente turnos docentes asegurando que cumplan con las restricciones establecidas. En un ciclo ‘while’, se intenta generar un turno hasta 1 000 veces, donde el contador ‘cant_intentos’ se incrementa con cada intento. Si no se cumplen las condiciones después de 1 000 intentos, se retorna ‘Free’, indicando un turno libre. La selección aleatoria comienza con la elección de una tipología de turno entre las opciones C, CP, L, T, S, y TE. Posteriormente, se selecciona aleatoriamente

un profesor del año académico actual, se obtiene su carga docente, y se adopta la asignatura asignada para ese periodo y profesor. La distribución de locales se filtra según la tipología del turno, y se selecciona un local libre. Los tipos de las variables (profesor, asignatura, tipología de turno, local) se verifican antes de retornar un diccionario que contiene la información del turno.

Usar el número 1 000 en ‘cant_intentos’ define un límite razonable de intentos para evitar bucles infinitos, asegurando eficiencia sin comprometer la búsqueda de combinaciones válidas. Un valor menor podría resultar en menos probabilidades de encontrar un turno válido, mientras que un valor mayor podría incrementar el tiempo de ejecución sin beneficios adicionales significativos. Los valores 1 y 0 son utilizados para inicializar contadores y verificaciones de estado lógico. La caja de herramientas (*toolbox*) en esta etapa facilita la generación y manipulación de atributos y individuos del algoritmo genético. Se configura con métodos para generar turnos, individuos y la población inicial, además de operadores genéticos para evaluar, cruzar, mutar y seleccionar individuos (David Elliott Goldberg y David E. Goldberg, 2012).

Código fuente 2.4. Ejemplo de código en Python

```
1      %Seleccionar una tipologia de turno aleatoriamente
2      completed = False
3      cant_intentos = 0
4      while not completed:
5          selected_tipologia_turno = random.choice(["C", "CP", "L", "T", "S", "TE
6              "])
7          cant_intentos += 1
8          if cant_intentos == 1000:
9              %Retornar 'Free' si no se cumplen las condiciones
10             return {
11                 'turno': 'Free'
12             }
13
14     selected_profesor = None
15     selected_asignatura = None
16     selected_local = None
17
18     try:
19         %Obtener los profesores del anno academico actual
20         profesores_del_anio = pl_models.carga_annoAcademico.objects.filter(
21             anno=self.grupo.anno_academico
22         ).values_list('cargadocente', flat=True).distinct()
23
24         profesores_del_anio = pl_models.CargaDocente.objects.filter(
25             id__in=profesores_del_anio
26         ).values_list('profesor', flat=True)
27
28         profesores_del_anio = list(admin_models.Profesor.objects.filter(
```

```
28         id__in=profesores_del_anio
29     ))
30
31     %Verificar que la lista de profesores no está vacía
32     if not profesores_del_anio:
33         raise Exception("No hay profesores disponibles para el anno
34             academico especificado.")
35
36     %Seleccionar un profesor aleatoriamente
37     selected_profesor = random.choice(profesores_del_anio)
38
39     %Obtener la carga docente del profesor seleccionado
40     carga_docente = pl_models.CargaDocente.objects.get(profesor=
41         selected_profesor)
42
43     carga_asignaturas = pl_models.carga_Asignaturas.objects.filter(
44         cargadocente=carga_docente
45     ).values_list('asignatura', flat=True)
46
47     asignaturas_anio = pl_models.Asignatura.objects.filter( id__in=
48         carga_asignaturas, anno_academico=self.grupo.anno_academico,
49         periodo=self.periodo ).distinct()
50
51     %Verificar que la lista de asignaturas no está vacía
52     if not asignaturas_anio:
53         raise Exception("No hay asignaturas disponibles para el anno
54             academico y periodo especificados.")
55
56     %Seleccionar una asignatura aleatoriamente
57     selected_asignatura = random.choice(asignaturas_anio)
58
59
60     %Definir la distribucion de locales por tipo de turno
61     distribucion = {
62         "C": ['Salon de conferencia', 'Aula especializada', 'Teatro'],
63         "CP": ['Aula especializada', 'Aula teorica'],
64         "L": ['Laboratorio'],
65         "T": ['Aula teorica', 'Salon de conferencia'],
66         "S": ['Aula teorica', 'Salon de conferencia'],
67         "TE": ['Aula teorica'],
68         "PP": ['Salon de conferencia'],
69         "PF": ['Salon de conferencia'],
70         "TC": ['Salon de conferencia', 'Aula teorica']
71     }
72
73     %Filtrar los locales disponibles segun la tipologia de turno
```

```

        seleccionada
69         locales = pl_models.Local.objects.filter(designacion__in=
            distribucion[selected_tipologia_turno])
70         selected_local = None
71
72         %Verificar que la lista de locales no este vacia
73         if locales.exists():
74             selected_local = random.choice(list(locales)) %Escoger local
                aleatoriamente
75     except Exception as e:
76         completed = False
77
78     %Verificar los tipos de las variables antes de retornar el resultado
79     if isinstance(selected_profesor, admin_models.Profesor) and isinstance(
        selected_asignatura, pl_models.Asignatura):
80         if isinstance(selected_tipologia_turno, str) and isinstance(
            selected_local, pl_models.Local):
81             return {
82                 'turno': {
83                     "profesor": selected_profesor,
84                     "asignatura": selected_asignatura,
85                     "tipo_clase": selected_tipologia_turno,
86                     "local": selected_local,
87                 }
88             }
```

Paso 4: Cruce

El cruce es un operador genético fundamental que combina partes de dos horarios padres seleccionados para crear uno o más horarios hijos (Michalewicz, 1999). Este proceso se maneja con una probabilidad de cruce (*cxpb*) establecida en 0.5, lo que significa que hay un 50 % de probabilidad de que dos individuos seleccionados se crucen para formar descendencia. Este valor asegura una adecuada mezcla de genes sin sobrecargar el proceso con demasiados cruces. Usar un valor menor reduciría la variabilidad genética, limitando la eficacia del algoritmo, mientras que un valor mayor podría llevar a una excesiva recombinación, dificultando la estabilización de soluciones óptimas. Durante el cruce, se seleccionan aleatoriamente dos horarios padres y un punto de cruce en ambos horarios. Luego, se intercambian las asignaciones de asignaturas, profesores y aulas a partir del punto de cruce para generar dos nuevos horarios hijos. Estos horarios hijos reemplazarán a algunos de los horarios en la población actual para la próxima generación.

Código fuente 2.5. Ejemplo de código en Python

```

1     %Aplicar el algoritmo genético
2     print('[] Aplicar el algoritmo genético')
3     algorithms.eaSimple(population, self.toolbox, cxpb=0.5, mutpb=0.2, ngen=40,
```

```
stats=None, halloffame=None, verbose=True)
```

Paso 5: Mutación

La mutación es un operador genético que introduce pequeñas modificaciones aleatorias en los horarios para ayudar a mantener la diversidad en la población y explorar soluciones nuevas. Por ejemplo, se puede seleccionar aleatoriamente un horario y cambiar la asignación de una asignatura a un profesor o aula diferente, siempre y cuando se cumplan las restricciones básicas. La probabilidad de mutación debe ser baja para evitar cambios drásticos en los horarios. En este código, la función 'mutar_turno' gestiona el proceso de mutación de los horarios. Si un número aleatorio es mayor que la probabilidad de mutación 'indpb', se procede con la mutación; de lo contrario, se retorna el horario sin cambios. Se utiliza la probabilidad de mutación *mutpb*=0.2, lo que significa que hay un 20 % de probabilidad de que ocurra una mutación. Este valor es lo suficientemente bajo para evitar alteraciones drásticas en los horarios, pero lo suficientemente alto para mantener la diversidad en la población (Michalewicz, 1999).

Durante la mutación, se recorren todos los turnos del horario. Se verifican restricciones, como la disponibilidad de locales y profesores, y se genera un nuevo turno si no hay afectaciones. La selección de profesores y asignaturas se basa en la disponibilidad y carga docente. Los locales disponibles se filtran según la tipología del turno y se asigna uno aleatoriamente. Si se encuentra un mejor turno, se actualiza el horario; de lo contrario, se marca como 'Free'. Usar un valor menor para *mutpb* podría reducir la variabilidad genética, limitando la exploración de nuevas soluciones. Un valor mayor, en cambio, podría introducir demasiadas modificaciones, dificultando la estabilidad y eficacia del algoritmo. Así, el valor de 0.2 es un equilibrio que permite mantener la diversidad genética sin desestabilizar el proceso evolutivo.

Código fuente 2.6. Ejemplo de código en Python

```
1 algorithms.eaSimple(population, self.toolbox, cxpb=0.5, mutpb=0.2, ngen=40,
stats=None, halloffame=None, verbose=True)
```

Código fuente 2.7. Ejemplo de código en Python

```
1 def mutar_turno(self, individual, indpb):
2     if not random.random() > indpb:
3         return individual,
4     else:
5         turnos_x_dia = {}%Para controlar la cantidad de turnos diarios
6         distribucion = {
7             "C": ['Salon de conferencia', 'Aula especializada', 'Teatro'],
8             "CP": ['Aula especializada', 'Aula teorica'],
9             "L": ['Laboratorio'],
10            "T": ['Aula teorica', 'Salon de conferencia'],
11            "S": ['Aula teorica', 'Salon de conferencia'],
```

```
12         "TE": ['Aula teorica'],
13         "PP": ['Salon de conferencia'],
14         "PF": ['Salon de conferencia'],
15         "TC": ['Aula teorica']
16     }
17
18     %Crear diccionario de combinaciones de turnos por asignatura
19     contexto = {} %<asignaturas | lista de profesores && turnos generados>
20     asignaturas = pl_models.Asignatura.objects.filter(periodo=self.periodo,
21                                                         anno_academico=self.grupo.anno_academico)
22     for asignatura in asignaturas:
23         asignatura_id = str(asignatura.id)
24         %verificar que hayan profesores para esa asignatura (luego se ve si
25         esta disponible)
26         profesores_de_asignatura = pl_models.carga_Asignaturas.objects.
27             filter(asignatura=asignatura).values_list('cargadocente', flat=
28             True).distinct()
29         profesores_de_asignatura = pl_models.CargaDocente.objects.filter(
30             id__in=profesores_de_asignatura).values_list('profesor', flat=
31             True)
32         profesores_de_asignatura = admin_models.Profesor.objects.filter(
33             id__in=profesores_de_asignatura).distinct()
34         if profesores_de_asignatura:
35             %generar los turnos para esa asignatura
36             %obtener el balance de carga de la asignatura y ordenarlo segun
37             la frecuencia, obteniendo solamente el tipo de clase
38             balance_carga = list(pl_models.BalanceCarga.objects.filter(
39                 asignatura=asignatura).order_by('frecuencia').values_list('
40                 tipologia', flat=True))
41             balance_carga = [elemento for elemento in balance_carga if
42                             elemento != 'N']
43             if balance_carga:
44                 contexto[asignatura_id]={}
45                 contexto[asignatura_id]['assign']=asignatura
46                 contexto[asignatura_id]['profesores']=
47                     profesores_de_asignatura
48                 contexto[asignatura_id]['bc']=balance_carga
49
50     %Recorrer todos los turnos del horario
51     for idx in range(len(individual)):
52         dia_index = idx // self.cant_turnos
53         turno_index = (idx%self.cant_turnos) + 1%de 1 a cant_turnos
54         %Verificar restricciones de medio día
55         mid_day_turn = self.cant_turnos // 2
56         mejor_turno = None
```

```
46         turno_preado = None
47         if turno_index == mid_day_turn + 1 and isinstance(individual[idx -
48             1]['turno'], dict):
49             pass
50         else:
51             %verificar si no hay afectacion de turno
52             afectaciones_turno = pl_models.Afectacion_Turno.objects.filter(
53                 dia=utils.sumar_dias(fecha=self.lunes, dias=dia_index)).
54                 distinct()
55             afectaciones_turno = pl_models.Turnos_Afectados_Turno.objects.
56                 filter(afectacion_id__in=afectaciones_turno, turno=
57                 turno_index).values_list('afectacion_id', flat=True).distinct
58                 ()
59             afectaciones_turno = pl_models.Grupos_Afectados_Turno.objects.
60                 filter(afectacion_id__in=afectaciones_turno, grupo_id=self.
61                 grupo).distinct()
62
63             actividades_turno = pl_models.Actividad.objects.filter(dia=
64                 utils.sumar_dias(fecha=self.lunes, dias=dia_index)).distinct
65                 ()
66             actividades_turno = pl_models.Turnos_Actividad.objects.filter(
67                 turno=turno_index, actividad_id__in=actividades_turno).
68                 values_list('actividad_id', flat=True).distinct()
69             actividades_turno = pl_models.Grupos_Actividad.objects.filter(
70                 actividad_id__in=actividades_turno, grupo=self.grupo)
71
72             if not afectaciones_turno and not actividades_turno:
73                 for asignatura_id, data in contexto.items():
74                     sesion_correcta = 'Tarde' if turno_index >= self.
75                         cant_turnos // 2 else 'Manana'
76
77                     if data['asign'].sesion != sesion_correcta:
78                         continue
79                     if not data['bc']:
80                         continue
81
82                     turno_preado = {
83                         'tipo_clase': data['bc'][0], %str%Tomamos el primer
84                         turno disponible
85                         'asignatura': data['asign'], %Asignatura
86                     }
87                     selected_asignatura = turno_preado['asignatura']
88                     selected_tipo_clase = turno_preado['tipo_clase']
89
90                     if str(dia_index) in turnos_x_dia:
91                         if selected_asignatura.id in turnos_x_dia[str(
```

```

    dia_index)]:
77         continue
78         %ir a la siguiente asignatura si esta ya se dio
            hoy
79     else:
80         turnos_x_dia[str(dia_index)]=[]
81         %si no se ha dado ninguna clase ponerle al dia una
            lista vacia,
82         %hasta que se vayan agregando los ids de las
            asignaturas dadas.
83     selected_profesor = None
84     for prf in data['profesores']:
85         if not selected_profesor is None:
86             pass
87             ready = True
88             %ver si el profesor tiene alguna afectacion para
                este turno
89             afectacion_profesor = pl_models.Afectacion_Profesor
                .objects.filter(dia=utils.sumar_dias(fecha=self.
                    lunes,dias=dia_index),turno=turno_index,
                    profesor_id=prf)
90             if afectacion_profesor:
91                 ready = False
92
93             %si el turno actual es uno de los del medio dia
94             %ver si el profesor tiene doble turno al medio dia.
95             if turno_index == mid_day_turn + 1 and ready ==
                True:
96                 if isinstance(individual[idx-1]['turno'],dict):
97                     if individual[idx-1]['turno']['profesor'].
                        id == prf.id:
98                         ready = False
99
100             %ver si tiene algun otro turno pendiente en otro
                horario
101             posible_turno = pl_models.Turno.objects.filter(
                profesor_id=prf,dia=utils.sumar_dias(fecha=self.
                    lunes,dias=dia_index),turno=turno_index)
102             if posible_turno.exists():
103                 ready = False
104             if ready:
105                 selected_profesor = prf
106             if selected_profesor is None:
107                 continue
108
109         turno_precreado['profesor']=selected_profesor
```

```
110
111
112         %Filtrar locales disponibles
113         todos_locales = pl_models.Local.objects.all()
114         locales_disponibles = [
115             local for local in todos_locales
116             if local.designacion in distribucion.get(
117                 turno_preactado['tipo_clase'], []) and
118                 not pl_models.Indisponibilidad_Local.objects.filter(
119                     (local=local, dia=utils.sumar_dias(fecha=self.
120                     lunes, dias=dia_index), turno=turno_index).
121                     exists()
122                 ]
123
124         %Verificar disponibilidad de locales y afectaciones del
125         profesor
126         if locales_disponibles:
127             local_asignado = random.choice(locales_disponibles)
128             turno_preactado['local'] = local_asignado
129             mejor_turno = turno_preactado
130             break
131
132     if mejor_turno:
133         individual[idx] = {
134             'turno': {
135                 'profesor': mejor_turno['profesor'],
136                 'asignatura': mejor_turno['asignatura'],
137                 'tipo_clase': mejor_turno['tipo_clase'],
138                 'local': mejor_turno['local']
139             }
140         }
141     turnos_x_dia[str(dia_index)].append(mejor_turno['asignatura'].
142     id)
143     contexto[str(mejor_turno['asignatura'].id)]['bc'].pop(0)
144     %Eliminar la frecuencia del balance de carga de la asignatuea
145     if not contexto[str(mejor_turno['asignatura'].id)]['bc']:
146         del contexto[str(mejor_turno['asignatura'].id)] %Eliminar
147         la asignatura si no quedan mas turnos
148
149     else:
150         individual[idx] = {'turno': 'Free'}
151
152     return individual,
```

Paso 6: Reemplazo

En esta etapa, se decide cómo se reemplazarán los horarios en la población actual con los nuevos horarios

generados a través del cruce y la mutación. En el contexto del código, el reemplazo se gestiona mediante la línea `algorithms.eaSimple(population, self.toolbox, cxpb=0.5, mutpb=0.2, ngen=40, stats=None, halloffame=None, verbose=True)`. Este proceso se asegura de mantener el tamaño de la población constante mientras evoluciona la calidad de los horarios. Durante cada generación del algoritmo genético, los individuos de la población actual se seleccionan, cruzan y mutan para generar una nueva población de horarios. La nueva población resultante de estos operadores genéticos reemplaza a la población anterior. Esta estrategia de reemplazo se basa en mantener los individuos más aptos (con mejores puntuaciones de aptitud), permitiendo que solo los mejores horarios prevalezcan en cada nueva generación (Abramson, 1991). Así, a medida que las generaciones progresan, la población evoluciona hacia horarios más óptimos y eficientes, cumpliendo mejor las restricciones y preferencias establecidas. La elección del número 40 para la cantidad de generaciones en el algoritmo genético se basa en encontrar un equilibrio entre la exploración del espacio de soluciones y el tiempo de procesamiento.

Con 40 generaciones, se busca permitir suficiente tiempo para que la población evolucione y converja hacia soluciones óptimas, mejorando la calidad de los horarios en cada iteración. Este número se considera adecuado para proporcionar una evolución progresiva y suficiente, donde los operadores de cruce y mutación pueden realizar cambios beneficiosos sin llevar el proceso a un tiempo de cálculo innecesariamente largo. Un número menor de generaciones podría no permitir que el algoritmo explore suficientemente el espacio de soluciones y refinar adecuadamente los individuos, lo que podría resultar en horarios subóptimos. Por otro lado, un número mayor de generaciones podría incrementar significativamente el tiempo de procesamiento sin ofrecer mejoras proporcionales adicionales en la calidad de los horarios generados.

Código fuente 2.8. Ejemplo de código en Python

```
1 algorithms.eaSimple(population, self.toolbox, cxpb=0.5, mutpb=0.2, ngen=40,
    stats=None, halloffame=None, verbose=True)
```

Paso 7: Iteración

La iteración es un componente crucial de cualquier algoritmo genético, pues permite que la población evolucione hacia soluciones óptimas a lo largo del tiempo. En cada iteración, se repiten una serie de pasos críticos, incluyendo la selección de individuos, el cruce para combinar sus características y la mutación para introducir variaciones. Estos pasos se llevan a cabo durante varias generaciones para explorar diferentes combinaciones de horarios y adaptarse a las restricciones y preferencias específicas de la confección de horarios docentes. El proceso iterativo permite que con cada generación, la calidad de los horarios mejore progresivamente. La línea de código `algorithms.eaSimple(population, self.toolbox, cxpb=0.5, mutpb=0.2, ngen=40, stats=None, halloffame=None, verbose=True)` gestiona este proceso, aplicando el algoritmo genético a la población durante 40 generaciones. La cantidad de generaciones se establece en 40 para proporcionar suficiente tiempo para la evolución sin extender demasiado el tiempo de procesamiento (ibíd.).

Paso 8: Resultado Final

Finalmente, se selecciona la mejor solución como el horario docente final, supervisando la aptitud de las soluciones a lo largo de las iteraciones y estableciendo un criterio de parada adecuado (López, 2010). Por ejemplo, se puede detener el algoritmo después de un número fijo de generaciones o cuando la mejora en la aptitud de los horarios sea insignificante. La mejor solución se puede identificar como el horario con la puntuación de aptitud más alta en la última generación. El código proporcionado convierte *best_ind*, que es la lista del mejor individuo (horario), en una matriz organizada por días y turnos. La matriz "horario" se transforma en "turnos", una lista de listas que contiene los detalles de cada turno. Luego, se crea un *DataFrame* de *pandas* a partir de esta matriz, que facilita la visualización y manipulación de los datos del horario. Este *DataFrame* se imprime y se retorna como resultado. La función 'Iniciar' crea una instancia de 'GeneradorHorario' con los parámetros específicos del grupo, periodo, cantidad de turnos y el lunes de la semana actual. Este generador se utiliza para crear y evaluar horarios, aplicando el algoritmo genético para encontrar la mejor solución posible. Al finalizar, la mejor solución se muestra como un *DataFrame*, reflejando el horario docente óptimo para el periodo y grupo especificados.

La aplicación de algoritmos genéticos en la generación de horarios docentes permite optimizar el proceso, explorando un espacio de soluciones y convergiendo hacia horarios que cumplen con las restricciones y preferencias establecidas. Cada etapa del algoritmo, desde la definición de la población inicial hasta la selección del resultado final, juega un papel crucial en la obtención de horarios docentes eficientes y satisfactorios.

2.2.1. Requisitos no funcionales del sistema

Los requisitos no funcionales son especificaciones que describen cómo debe comportarse un sistema en aspectos como usabilidad, seguridad, confiabilidad, portabilidad, eficiencia, mantenibilidad y *hardware/software* requerido. Estos requisitos complementan a los requisitos funcionales, que definen qué debe hacer el sistema (RNF, 2024).

A continuación, se presenta una tabla con los requisitos no funcionales establecidos para el sistema, junto con una breve descripción de cada uno:

Tabla 2.2. Requisitos no funcionales. Fuente: Elaboración propia.

No.	Requisitos	Descripción
RnF 1	Usabilidad	
RnF 1.1		El sistema debe contar con un diseño responsivo que adapte automáticamente la interfaz a diferentes resoluciones de pantalla.
RnF 2	Seguridad	
Continúa en la siguiente página		

No.	Requisitos	Descripción
RnF 2.1		La información del sistema debe estar protegida contra accesos no autorizados.
RnF 2.2		En caso de errores, el sistema no debe mostrar detalles que puedan comprometer su seguridad.
RnF 2.3		El sistema debe cerrar automáticamente la sesión del usuario después de un período de inactividad.
RnF 2.4		El sistema debe garantizar la integridad de la información mediante mecanismos de control de acceso basados en roles de usuario.
RnF 3	Portabilidad	
RnF 3.1		El sistema debe funcionar en navegadores como <i>web Chrome, Mozilla Firefox</i> y otros utilizados en la institución.
RnF 4	Mantenibilidad	
RnF 4.1		El sistema debe permitir realizar modificaciones posteriores para mejoras o nuevos requisitos.
RnF 4.2		El sistema debe estar bien documentado para facilitar su mantenimiento.
RnF 5	<i>Hardware</i>	
RnF 5.1		El servidor de base de datos debe tener una capacidad mínima de 500 GB.
RnF 5.2		El servidor de aplicaciones <i>web</i> debe tener una capacidad mínima de 500 GB.
RnF 5.3		Los servidores <i>web</i> y de base de datos deben tener al menos 16 GB de memoria RAM.
RnF 5.4		Se requiere una tarjeta de red con velocidad de 100 Mbps.
RnF 5.5		Se requiere un servidor HPC (High Performace Computing) para implementar el algoritmo genético de 16 GB de memoria RAM, con un almacenamiento de 500 GB.
RnF 6	<i>Software</i>	
RnF 6.1		Se requiere como mínimo <i>Chrome 60.0+</i> o <i>Mozilla Firefox 60.0+</i> .
RnF 6.2		Se puede utilizar en sistemas operativos como <i>GNU Linux Ubuntu Server</i> o <i>Windows 10 Home</i> siempre y cuando sea compatible con <i>Pythonv3.12</i> .

Estos requisitos no funcionales establecen las condiciones y restricciones que debe cumplir el sistema para

garantizar su correcto funcionamiento, usabilidad, seguridad y mantenibilidad.

2.3. Planificación del desarrollo

La fase de planificación en XP es crucial para establecer los cimientos del proyecto y asegurar su éxito. Durante la misma, se define el alcance general, los objetivos a cumplir y se traza un plan de acción a través de iteraciones. Además, se determinan las prioridades y las funcionalidades que se abordarán en cada ciclo de desarrollo (Beck y Andres, 2004). El objetivo principal de la planificación es crear un mapa de ruta claro y realista que guíe el avance del proyecto. Para lograr esto, se establece un proceso iterativo en el que se definen plazos y se priorizan las tareas a realizar.

La estrecha colaboración entre el equipo de desarrollo y el cliente es de vital importancia, juntos, identifican y acuerdan las funcionalidades prioritarias que se implementarán en la iteración actual, dejando para iteraciones futuras aquellas de menor prioridad. Este enfoque permite mantener una visión clara de las actividades a abordar en cada iteración, lo que a su vez ayuda a gestionar las expectativas del cliente sobre el progreso del proyecto (Rivadeneira Molina; Vilanova; Miranda y D. Cruz, 2013).

Para llevar a cabo una planificación efectiva, es fundamental iniciar con la identificación y definición de las Historia de Usuario (HU). Estas descripciones concisas de las necesidades existentes admiten enfocar el desarrollo en los requisitos reales del cliente y priorizar las funcionalidades de acuerdo con su valor.

2.4. Usuarios del sistema

Los usuarios de un sistema son las personas o entidades que interactúan con el sistema para lograr objetivos específicos. Cada usuario desempeña un rol o perfil particular, con necesidades y requisitos distintos en términos de lo que esperan que el sistema pueda hacer por ellos (Penadés y Letelier Torres, 2006).

El desarrollo de *software* no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto.

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del *software* con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir *software* y, a la vez, generando un amplio debate entre sus seguidores y quienes, por escepticismo o convencimiento, no las ven como una alternativa para las metodologías tradicionales.

A continuación, se presentan los usuarios clave de IAHorDo y sus respectivas funciones:

Tabla 2.3. Usuarios del sistema. Fuente: Elaboración propia.

Usuario	Descripción
Planificador	Los planificadores, encabezados por el profesor principal de cada año, el vicedecano de formación y el planificador de la facultad, tendrán un rol fundamental en la gestión del horario docente. Su objetivo principal será garantizar una asignación de los recursos disponibles, como marco de horario, balance de carga, carga docente, asignaturas, locales, grupos, actividades, afecaciones tanto de los profesores como de los turnos y horarios.
Decano	El decano de la facultad tendrá un rol fundamental en la gestión del horario docente específicamente para visualizar y exportar la carga docente.
Invitado	Los estudiantes y profesores podrán consultar y acceder a los horarios generados, filtrados por su año académico, grupo y semana específica. Esto les permitirá mantenerse informados sobre sus respectivas clases y actividades.

Estos usuarios clave, con sus roles y necesidades específicas, serán el punto de partida para la fase de planificación del desarrollo del sistema. Comprender sus requisitos y expectativas es crucial para crear un plan concreto que guíe el proyecto de manera ágil y eficiente (Contributors, 2024).

2.4.1. Historias de usuario

Las historias de usuario son una herramienta fundamental utilizada en XP en la especificación de requisitos de un proyecto de desarrollo de *software*. Estas descripciones representan las necesidades y expectativas del usuario final, expresadas en su propio lenguaje y perspectiva (Beck y Andres, 2004). Al elaborar las historias de usuario de manera detallada y precisa, se logra una comprensión clara de los requisitos funcionales

y no funcionales del sistema. Este enfoque orientado al usuario es esencial para el éxito del proyecto, pues establece una comunicación efectiva entre todas las personas involucradas (Rosenberg y Stephens, 2008).

Al capturar las necesidades del usuario en un formato comprensible, se evitan malentendidos y se asegura que el producto final cumpla con lo que se espera. Además, al tener una visión clara de las funcionalidades y características requeridas, se pueden establecer criterios de prioridad y asignar recursos de manera adecuada. Esto contribuye a una planificación y ejecución positiva del proyecto.

Para contribuir a una práctica efectiva en el desarrollo de *software*, se agruparán las historias de usuario bajo categorías como "Gestionar", pues permiten una organización lógica de las funcionalidades del sistema. Esto facilita la comprensión y es una estrategia organizativa que mejora la planificación, la comunicación y la eficiencia en el desarrollo del *software*.

A continuación, se identificarán algunas de las historias de usuario para comprender las necesidades y las expectativas que debe cumplir el sistema:

Tabla 2.4. Historia de usuario # 1

Historia de usuario	
Número: 1	Nombre: Autenticar usuario
Usuario: Usuario	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El sistema requiere que los usuarios validen su identidad mediante un proceso de autenticación para poder acceder al sistema. Para ello, deberán ingresar sus credenciales, incluyendo el nombre de usuario y contraseña.	
Observaciones: Solo los usuarios registrados en el sistema podrán autenticarse.	

Continúa en la próxima página

Tabla 2.4. Continuación de la página anterior

Interfaz:



IAHorDo

Iniciar Sesión

Usuario

Contraseña

Inicie Sesión

Tabla 2.5. Historia de usuario # 2

Historia de usuario	
Número: 2	Nombre: Gestionar usuario
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla 2.5. Continuación de la página anterior

Descripción: Los usuarios con roles de administración, como el administrador del sistema tendrán la capacidad de gestionar la información de los usuarios del sistema, dígase insertar, listar, modificar elementos existentes y eliminar usuarios, que pueden ser planificadores, el decano, profesores y estudiantes.

Escenario 1: Insertar usuario Si el usuario es un estudiantes se tendrán en cuenta los datos establecidos posteriormente más el año académico Si el usuario es planificador o decano se tendrán en cuenta los datos establecidos posteriormente más la facultad Si el usuario es un profesor se tendrán en cuenta los datos establecidos posteriormente más la categoría docente, la formación académica, residente. El administrador podrá crear nuevos registros de usuarios en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente:

- Nombre de usuario: campo requerido que debe ser único en el sistema.
- Contraseña: deberá cumplir con criterios de seguridad establecidos por la organización.
- Nombre: campo obligatorio que identificará al usuario dentro del sistema.
- Apellidos: campo opcional que proporcionará información adicional sobre el usuario.
- Rol: selección de un perfil de acceso predefinido que determinará las capacidades del usuario dentro del sistema.
- Este proceso garantizará la incorporación efectiva de nuevos usuarios al sistema, asegurando la integridad y seguridad de los datos ingresados.

Escenario 2: Listar usuario El administrador tendrá la capacidad de listar información detallada sobre cualquier usuario registrado en el sistema. Esta funcionalidad permitirá:

- Acceso rápido a los datos básicos del usuario (nombre, apellidos, rol).
- Listado de la estructura organizativa asignada al usuario.
- Esta función facilitará una comprensión clara de la base de datos de usuarios y su distribución dentro del sistema, resultando útil para propósitos de gestión y supervisión.

Escenario 3: Modificar usuario El administrador podrá actualizar la información de los usuarios existentes en el sistema. Este proceso implicará:

- Edición de campos como nombre, apellido y rol.
- Actualización de contraseñas, siempre respetando las políticas de seguridad establecidas.
- Ajuste de permisos o roles si es necesario.
- Corregir errores tipográficos o inconsistencias en los datos.
- Este escenario garantizará que la información de los usuarios permanezca actualizada y precisa, manteniendo así la eficacia y confiabilidad del sistema.

Escenario 4: Eliminar usuario El administrador dispondrá de la opción para eliminar completamente un registro de usuario del sistema. Este proceso implicará:

- Confirmación de la acción antes de proceder con la eliminación.
- Eliminación completa del registro del usuario en todas las tablas relacionadas.
- Actualización automática de referencias cruzadas para mantener la integridad de los datos.
- Este escenario permitirá la eliminación eficiente de usuarios que ya no sean necesarios en el sistema, manteniendo así la organización y seguridad de los datos.

Tabla 2.5. Continuación de la página anterior

Observaciones: Solo los usuarios autenticados con rol de planificador podrán realizar acciones de gestión de usuarios.

Interfaz:

Listado de usuarios

Registrar usuario

Mostrar ▼ Registro por página

Buscar

Nombre y Apellidos	Usuario	Rol	Acción
Katherine Borrero Meriño	katherinebm	admin	
Yorgelys González López	yorgelysgl	planificador	
Sailyn Salas Reyes	sailynsr	profesor	
Tania Cobas Aguilar	taniaca	estudiante	

Página 1 de 1

Tabla 2.6. Historia de usuario # 3

Historia de usuario	
Número: 3	Nombre: Gestionar asignaturas
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

Descripción: El planificador tendrá la capacidad de gestionar la información de las asignaturas dentro del sistema. Para ello se tendrá en cuenta la siguiente información de las asignaturas:

Escenario 1: Insertar asignatura El planificador podrá crear nuevos registros de asignatura en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente:

- Nombre de la asignatura
- Abreviatura
- Año

Escenario 2: Listar asignatura El planificador tendrá la capacidad de listar información detallada sobre cualquier asignatura registrada en el sistema. Esta funcionalidad permitirá:

- Acceso rápido a los datos básicos de la asignatura (nombre, abreviatura y año).
- Listado de la estructura organizativa asignada a la asignatura.

Escenario 3: Modificar asignatura El planificador podrá actualizar la información de los asignaturas existentes en el sistema. Este proceso implicará:

- Edición de campos como (nombre, abreviatura y año).
- Corregir errores tipográficos o inconsistencias en los datos.

Escenario 4: Eliminar asignatura

El planificador dispondrá de la opción para eliminar completamente un registro de asignatura del sistema. Este proceso implicará:

- Confirmación de la acción antes de proceder con la eliminación.
- Eliminación completa del registro de la asignatura en todas las tablas relacionadas.
- Actualización automática de referencias cruzadas para mantener la integridad de los datos.

Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de asignaturas.

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

Interfaz:

Asignaturas

Mostrar ▼ Entradas

Nombre	Abreviatura	Año académico	Periodo	Estado	Sesión	Acciones
Matemática Discreta I	MD I	Primer Año	Primer Semestre	Habilitada	Mañana	
Introducción a la Programación I	IP I	Primer Año	Primer Semestre	Habilitada	Mañana	
Educación Física I	EF I	Primer Año	Primer Semestre	Habilitada	Tarde	
Inglés	I	Primer Año	Primer Semestre	Habilitada	Tarde	

Página 1 de 1

Teniendo en cuenta las HU definidas, es fundamental realizar una estimación de los esfuerzos requeridos para implementar cada una de ellas, siguiendo las recomendaciones de la metodología XP. Este proceso de estimación permite al equipo de desarrollo planificar adecuadamente los recursos y el tiempo necesarios para completar cada funcionalidad del sistema.

2.4.2. Estimación por esfuerzos

La estimación de esfuerzos según la metodología XP consiste en determinar la cantidad de trabajo y recursos necesarios para implementar las historias de usuario. Esta estimación se basa en la complejidad de cada historia, las habilidades y experiencia del equipo de desarrollo, y otros factores relevantes. El objetivo de la estimación de esfuerzos es proporcionar una idea aproximada del tiempo y los recursos requeridos para completar cada historia de usuario. Esto ofrece al equipo de desarrollo planificar adecuadamente las iteraciones, asignar tareas y gestionar el tiempo (Beck y Andres, 2004).

A continuación, se presenta la estimación de esfuerzos por cada historia de usuario correspondiente a este proyecto:

Tabla 2.7. Estimación de esfuerzo por historia de usuario

Iteración	Historias de usuario		Puntos estimados (semanas)
1	1	Autenticar usuario	1
	2	Gestionar usuarios	2
	3	Gestionar asignaturas	1
	4	Gestionar balance de carga	1
	5	Gestionar grupos	1

Continúa en la próxima página

Tabla 2.7. Continuación de la página anterior

	6	Gestionar locales	1
	7	Gestionar actividades	1
	8	Gestionar afectaciones	1
	9	Gestionar carga docente	1
	10	Exportar carga docente	1
2	11	Gestionar marco horario	1
	12	Generar horario	1
	13	Exportar horario	1
	14	Visualizar horario	1
Total			15.0

Estas estimaciones proporcionan una guía inicial sobre el esfuerzo requerido para implementar cada HU. Es importante destacar que las estimaciones en XP son flexibles y pueden ajustarse a medida que se obtiene más información y conocimiento sobre el proyecto durante el desarrollo iterativo.

2.5. Plan de iteraciones

La metodología XP establece el plan de iteraciones como una parte fundamental del proceso de desarrollo. Este plan consiste en dividir el proyecto en ciclos de trabajo más pequeños y manejables, denominados iteraciones. Cada iteración tiene una duración fija, generalmente entre una y tres semanas, y se centra en la entrega de un conjunto de funcionalidades específicas y probadas. El plan de iteraciones en XP ofrece varios beneficios. Permite una mayor adaptabilidad y flexibilidad, pues se pueden hacer cambios en cada iteración según las necesidades y retroalimentación del cliente. Además, ayuda a mantener un enfoque constante en la entrega de valor al cliente, al priorizar las funcionalidades más importantes y realizar entregas frecuentes (Beck y Andres, 2004).

A continuación, se establece el plan de iteraciones que se llevará a cabo durante el ciclo del proyecto:

Tabla 2.8. Plan de duración de las iteraciones

Iteración	Historias de usuario		Duración (semanas)
1	1	Autenticar usuario	9
	2	Gestionar usuarios	
	3	Gestionar asignaturas	
	4	Gestionar balance de carga	
	5	Gestionar grupos	
	6	Gestionar locales	
	7	Gestionar actividades	
	8	Gestionar afectaciones	

Continúa en la próxima página

Tabla 2.8. Continuación de la página anterior

2	9	Gestionar carga docente	4.0
	10	Exportar carga docente	
	11	Gestionar marco horario	
	12	Generar horario	
	13	Exportar horario	
	14	Visualizar horario	
Total			13.0

2.5.1. Plan de entregas

El plan de entregas es una parte esencial en la metodología XP. Consiste en organizar las HU en grupos que serán entregados como incrementos del sistema, estableciendo el orden en el cual se realizarán esas entregas. Esta tarea recae en el cliente, quien tiene el conocimiento de sus necesidades y prioridades. El cronograma de entregas se construye tomando en cuenta las estimaciones de tiempo para el desarrollo de cada HU. Se analizan las estimaciones realizadas por el equipo de desarrollo y se determina la secuencia y fechas de entrega de cada iteración (Builes; D. L. R. Bedoya y J. W. B. Bedoya, 2019).

A continuación, se expone un plan detallado que establece las tareas y actividades a realizar durante cada iteración, incluyendo los plazos y los hitos importantes del proyecto:

Tabla 2.9. Plan de entregas. Fuente: Elaboración propia.

Entregable	Iteración 1	Iteración 2
Fecha	29-06-2024	10-12-2024

Este plan de entregas brinda una estructura organizada y transparente para la realización del proyecto, lo que facilita la comunicación, el seguimiento y la coordinación entre el equipo de desarrollo y el cliente. Luego de la creación de las historias de usuario, la elaboración del plan de iteraciones y la definición del plan de entregas, se da por concluida la fase de planificación en el desarrollo de *software*. Esta fase establece las bases para el diseño al definir los objetivos, las prioridades y la organización de las funcionalidades. Una vez que estos elementos están en su lugar, el equipo de desarrollo está listo para pasar a la fase de diseño con una comprensión sólida de lo que se debe lograr y cómo se debe lograr.

2.6. Diseño

La fase de diseño en la metodología XP es fundamental para el desarrollo exitoso de un proyecto. Durante esta etapa, se prioriza aún más la retroalimentación estrecha con el cliente, ayudando a evitar errores futuros. El objetivo es crear un diseño simple y claro, eliminando lo innecesario y enfocándose en las funcionalidades

esenciales del sistema. El enfoque incremental permite adaptarse a medida que se obtiene retroalimentación del cliente, realizando ajustes en el diseño en cada iteración. Además, se realizan pruebas de diseño para asegurar la calidad del sistema y detectar posibles problemas antes de la implementación. Se busca crear un diseño flexible y adaptable, anticipando cambios futuros y permitiendo la fácil incorporación de nuevas funcionalidades (Beck y Andres, 2004; Tobón y Carmona, 2007).

2.6.1. Representación arquitectónica

La arquitectura, en el contexto del desarrollo de *software*, se refiere a la estructura o diseño fundamental de un sistema o aplicación. La arquitectura de *software* proporciona una visión global y abstracta del sistema, definiendo cómo se divide en partes, cómo se comunican entre sí y cómo se gestionan los datos y las funcionalidades. Es como el plano o la estructura básica sobre la cual se construye el *software* (Navarro; Moreno; Aranda; Parra; Rueda y Pantano, 2017).

Para el desarrollo del IAHorDo se empleará la arquitectura MVT que es una variación del popular patrón Modelo Vista Controlador (MVC) utilizado en el desarrollo *web* (Ranjan, 2021). Esta arquitectura permite separar de manera clara y organizada las diferentes responsabilidades del desarrollo *web* en modelo, vista y *template*.

- **Modelo *Model*:** el modelo representa los datos y la lógica de negocio de la aplicación. *django*, los modelos son clases *Python* que interactúan con la base de datos. Cada modelo se asocia con una tabla en la base de datos y define métodos para realizar consultas y manipulaciones de datos.
- **Vista *View*:** la vista es la capa encargada de la presentación de los datos al usuario. En *django*, las vistas son funciones *Python* que gestionan las solicitudes del usuario y coordinan la interacción entre los modelos y las plantillas. Las vistas definen la lógica necesaria para obtener los datos del modelo y pasarlos a la plantilla correspondiente.
- **Plantilla *Template*:** la plantilla es la capa responsable de la presentación visual de los datos. En *django*, las plantillas son archivos HTML con código de plantillas que permiten mostrar los datos obtenidos del modelo de manera formateada. Las plantillas pueden contener etiquetas y lógica de control para generar el HTML dinámicamente.

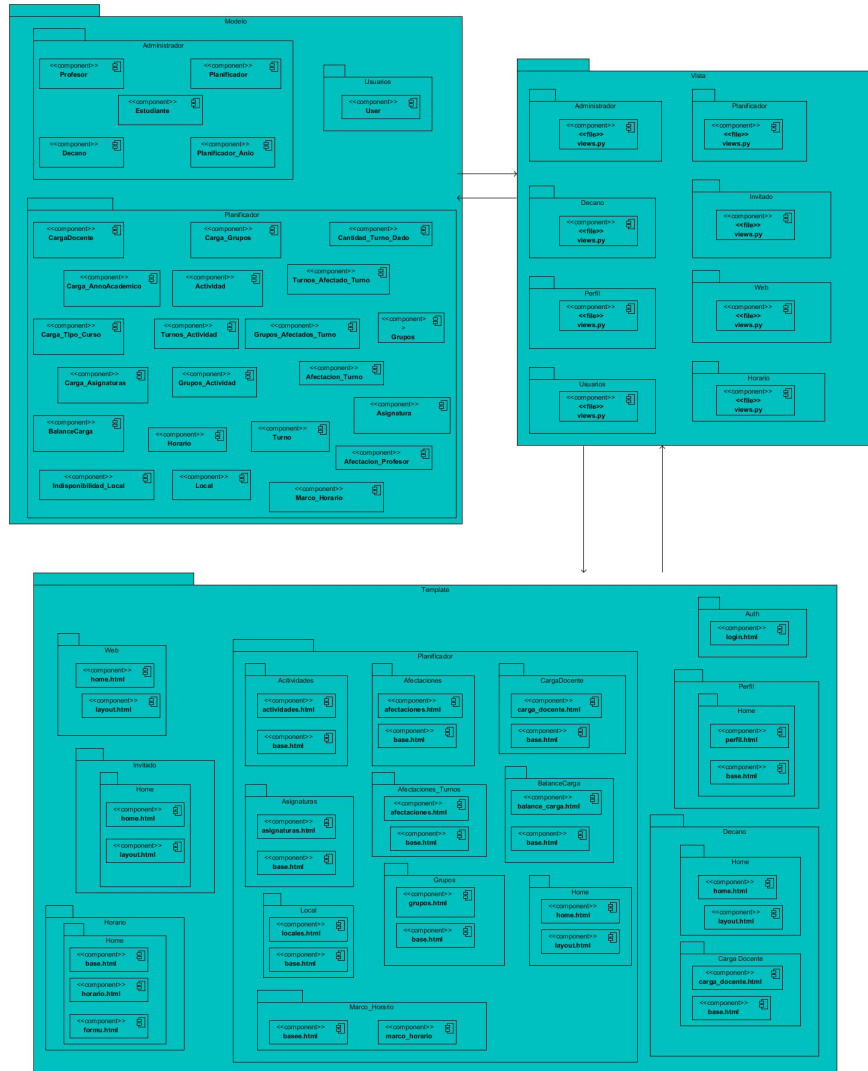


Figura 2.3. Representación del MVT

Esta arquitectura MVT proporciona una separación clara de responsabilidades, lo que facilita el desarrollo, el mantenimiento y la escalabilidad del sistema. Al dividir el código en modelos, vistas y plantillas, se promueve la reutilización de código, la modularidad y la testabilidad del sistema.

2.6.2. Modelos de datos

Un modelo de datos es la descripción formal de una base de datos. Permite definir las estructuras de datos que la componen, incluyendo su tipo, descripción y relaciones. También establece las restricciones de integridad que deben cumplirse. En este sentido, un modelo de datos facilita la representación de los elementos de la realidad que interviene en un problema determinado, así como las relaciones existentes entre ellos.

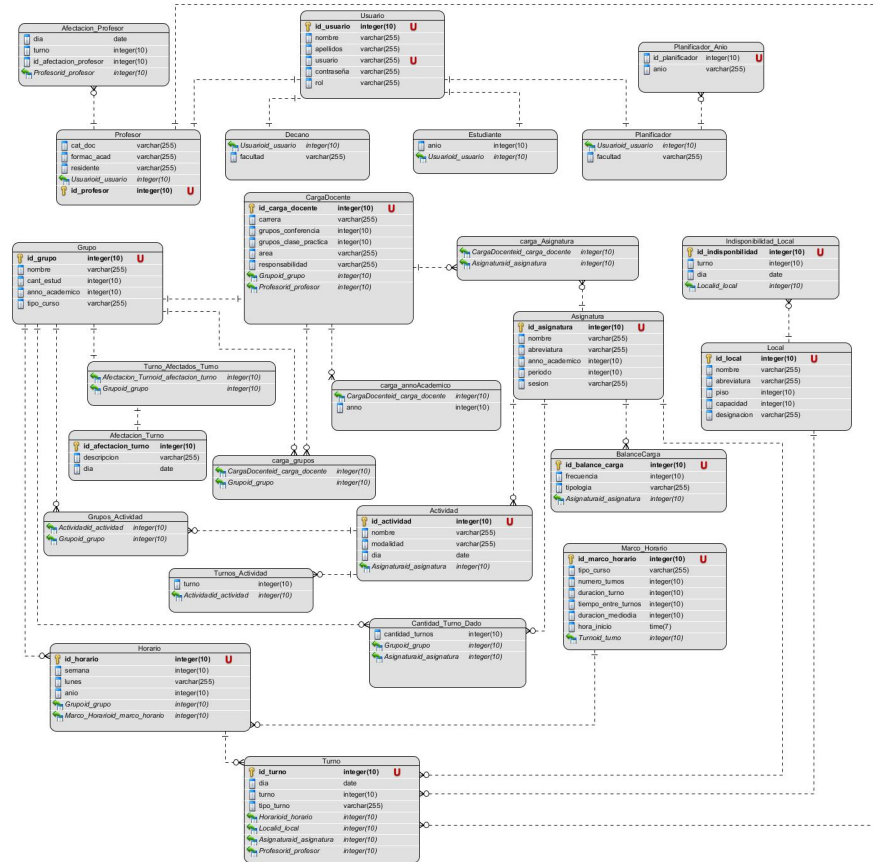


Figura 2.4. Modelo de Datos

2.7. Patrones de diseño

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos relacionados con el diseño de interacción o interfaces. En este contexto se han aplicado patrones General Responsibility Assignment Software Patterns (GRASP, por sus siglas en inglés) y Gang of Four (GOF, por sus siglas en inglés). Estos patrones están diseñados para ser extensibles y modificables según sea necesario, lo que facilita la evolución y el mantenimiento del *software* a lo largo del tiempo (Bradley, 2021; C. G. Lasater y Lasater, 2006).

2.7.1. Patrones GRASP

Los patrones GRASP proporcionan directrices esenciales sobre cómo asignar responsabilidades, así como sobre la creación y colaboración de objetos. En el sistema, se observa que cada clase desempeña una función específica y colabora con otras para completar tareas concretas, especialmente en la capa de acceso a datos. A continuación, se describen los patrones aplicados en este sistema:

- **Patrón experto:** este patrón se implementa mediante una clase que tiene conocimiento específico para establecer conexiones con la base de datos, junto con otra clase encargada de gestionar los datos. Esta última posee los recursos necesarios para realizar consultas, incluyendo operaciones de selección, inserción, modificación y eliminación.
- **Patrón de bajo acoplamiento:** este principio se centra en minimizar las dependencias entre objetos, lo cual es crucial para un diseño orientado a objetos eficaz. Por ejemplo, al mostrar las asignaturas de un año académico en el balance de carga, se debe recorrer la lista de asignaturas, lo que puede aumentar el acoplamiento entre las clases Asignaturas y BalanceCarga. Un alto acoplamiento implica que los cambios en una clase pueden afectar a otras, complicando el mantenimiento del código. Por el contrario, un bajo acoplamiento permite que las clases operen de manera más independiente, facilitando así su modificación y reutilización. Esto se logra asignando responsabilidades de forma que las clases interactúen solo cuando es necesario.
- **Patrón de alta cohesión:** este patrón busca que las clases tengan responsabilidades bien definidas y relacionadas entre sí, evitando una complejidad innecesaria.

2.7.2. Patrones GOF

Los patrones de diseño GOF se dividen en tres categorías: creacionales, estructurales y comportamiento, y son esenciales para el desarrollo de *software* orientado a objetos, pues establecen las interacciones entre clases y objetos. En el contexto del sistema que se está desarrollando con *django*, un *framework* de desarrollo *web* en *Python*, estos patrones son aplicados para optimizar la estructura del código. Dos patrones destacados en este sistema son el patrón de plantilla y el patrón iterador, que son particularmente útiles en aplicaciones *web*.

El patrón de plantilla, o *Template Method*, es un patrón estructural que establece la estructura de un algoritmo dentro de una superclase, permitiendo que las subclasses modifiquen pasos específicos sin alterar la estructura general. En *django*, este patrón se utiliza para definir la base de una vista o controlador, lo que permite a las subclasses implementar lógicas particulares sin cambiar la estructura general del sistema. Este enfoque es especialmente valioso en aplicaciones *web* donde varias vistas o controladores requieren una estructura común, pero con lógicas de procesamiento diferenciadas. Además, este patrón se manifiesta en las vistas basadas en clases de *django*, donde se define el esqueleto de un algoritmo y se delegan ciertos pasos a las subclasses. Esto facilita la reutilización del código y proporciona puntos de personalización, resultando en un código más limpio y fácil de mantener. Un ejemplo concreto es el uso de vistas genéricas en *django*, como la herencia de *ListView* para listar asignaturas o profesores.

Por otro lado, el patrón iterador es un diseño que permite recorrer los elementos de una colección sin revelar su representación interna. En el marco de *Django*, este patrón resulta útil para gestionar colecciones de objetos modelo, como usuarios o publicaciones. Facilita a los desarrolladores recorrer estos objetos sin

necesidad de acceder directamente a la base de datos o a la estructura interna de la colección. Esto no sólo mejora la eficiencia del código, sino también su legibilidad, especialmente cuando se manejan grandes volúmenes de datos. Un ejemplo práctico del uso del patrón iterador se observa al mostrar listas de asignaturas, profesores y locales dentro de las plantillas del sistema.

2.7.3. Diagrama de Despliegue

Un diagrama de despliegue es una representación gráfica que ilustra la disposición física de los componentes de un sistema, mostrando cómo se distribuyen los artefactos de *software* en nodos de *hardware*. Este tipo de diagrama es parte del UML y es utilizado para modelar la arquitectura en tiempo de ejecución de un sistema, facilitando la visualización de la infraestructura necesaria para su operación. Los elementos clave que componen un diagrama de despliegue son:

1. Nodos: los nodos representan los recursos computacionales, como servidores, dispositivos o entornos de ejecución. Se suelen representar como cubos o prismas y pueden contener componentes o artefactos. Los nodos pueden ser físicos *hardware* o lógicos *software* y son esenciales para entender cómo se ejecuta el sistema en su totalidad.
2. Relaciones: las relaciones indican cómo se comunican los nodos. Se representan mediante líneas que conectan nodos y pueden incluir información sobre el tipo de comunicación (como protocolos) y la multiplicidad (cuántas instancias están involucradas) en cada extremo.
3. Componentes: los componentes son unidades independientes del sistema que encapsulan una funcionalidad específica. Se representan como rectángulos con dos protuberancias en el lado izquierdo y pueden ser clasificados según su tipo. Los componentes ayudan a desglosar el sistema en partes más manejables.
4. Estereotipos: los estereotipos proporcionan información adicional sobre los nodos, especificando su función o características especiales. Por ejemplo, un nodo puede ser estereotipado como «*executionEnvironment*», «servidor», «cliente», o «base de datos», lo cual ayuda a clarificar su rol dentro del sistema.

Es importante incluir las conexiones entre los nodos para mostrar cómo se comunican. Por ejemplo, el PC cliente se conectará al servidor *web* a través de *HTTP/HTTPS*, y el servidor *web* se comunicará con el servidor de base de datos a través del protocolo *SQL* o similar. Este enfoque asegura que todos los componentes necesarios y sus requisitos no funcionales estén claramente representados en el diagrama, facilitando la comprensión del sistema completo.

Servidor *web*: este nodo es el encargado de mostrar el código fuente de la aplicación y proporciona a los usuarios las interfaces necesarias durante todos los procesos del negocio. Se comunica con el servidor de base de datos a través del protocolo *TCP/IP*, utilizando el puerto 8000 para las solicitudes *HTTP/HTTPS*. Debe contar con un disco duro de al menos 80 GB, una tarjeta RAM de 4 GB y una tarjeta de red con

velocidad aproximada a los 100 Mbps.

Dispositivo del cliente: este nodo representa la estación de trabajo del cliente, que necesita un navegador *web* para conectarse al servidor. Utiliza el protocolo *HTTPS* para garantizar una comunicación segura, estableciendo conexión a través del puerto 8000. El cliente debe contar con un navegador moderno como *Chrome* o *Mozilla Firefox*, sistemas operativos como *Linux* o *Windows*, y debe tener una tarjeta de red con velocidad cercana a los 100 Mbps.

Servidor de base de datos: Este nodo es responsable del almacenamiento de los datos registrados en la aplicación. Se comunica con el servidor *web* mediante el protocolo *TCP/IP* a través del puerto 5432, que es el utilizado por *PostgreSQL*. Para su correcto funcionamiento, debe tener una capacidad mínima de 80 GB, al menos 4 GB de RAM. Es esencial que este servidor cuente con *PostgreSQL* como gestor de base de datos, permitiendo así el acceso y manipulación eficiente de la información por parte tanto del usuario como de la aplicación.

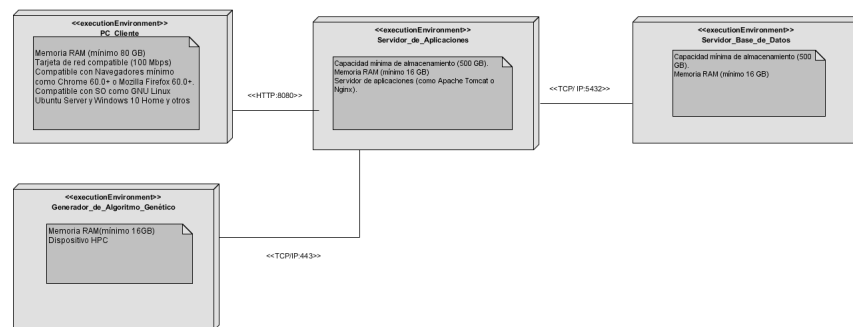


Figura 2.5. Diagrama de despliegue

2.7.4. Las tarjetas CRC

Las tarjetas Class-Responsibility-Collaboration (CRC, por sus siglas en inglés) son una técnica valiosa en el desarrollo de *software* orientado a objetos, introducida por Kent Beck y Ward Cunningham. Este método se utiliza para facilitar el análisis y diseño de las interacciones entre clases dentro de un sistema, ofreciendo una representación visual clara que ayuda a los equipos a comprender mejor la estructura del *software*.

Cada tarjeta CRC se divide en tres secciones clave: el nombre de la clase, sus responsabilidades y las colaboraciones necesarias para cumplir con dichas responsabilidades. Esta organización permite a los desarrolladores identificar rápidamente qué hace cada clase, así como con quién debe interactuar para realizar sus funciones. La simplicidad de este formato fomenta un enfoque colaborativo en el diseño, donde todos los miembros del equipo pueden contribuir a la discusión y refinamiento del modelo.

El uso de tarjetas CRC ofrece múltiples ventajas en el proceso de diseño:

- Visualización de interacciones: al proporcionar una representación visual de las clases y sus relaciones, las tarjetas CRC facilitan la identificación de problemas potenciales, como dependencias excesivas entre clases o ambigüedades en las responsabilidades asignadas.
- Fomento de la colaboración: estas tarjetas actúan como un puente de comunicación entre los distintos miembros del equipo, incluidos desarrolladores, analistas y diseñadores. Al tener un recurso visual común, se promueve un entendimiento compartido del diseño del sistema, lo que puede resultar en una alineación más efectiva respecto a las funcionalidades y objetivos del *software*.
- Detección temprana de problemas: la claridad que ofrecen las tarjetas CRC permite detectar conflictos de responsabilidades o problemas de diseño en etapas tempranas del desarrollo. Esto no sólo ahorra tiempo y recursos a largo plazo, sino que también mejora la calidad del producto final.
- Iteración rápida: la naturaleza flexible de las tarjetas permite realizar cambios fácilmente. Los equipos pueden iterar sobre el diseño sin complicaciones, explorando diferentes alternativas rápidamente y adaptándose a requisitos cambiantes.

Tabla 2.10. Tarjeta CRC # 1

Tarjeta CRC	
Clase: Usuario	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	

Tabla 2.11. Tarjeta CRC # 2

Tarjeta CRC	
Clase: Profesor	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	User

Tabla 2.12. Tarjeta CRC # 3

Tarjeta CRC	
Clase: Asignatura	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	

Las tarjetas CRC no sólo ayudan a organizar y clarificar las interacciones entre clases, sino que también fomentan un ambiente colaborativo que mejora la comunicación y la eficiencia del equipo. Al facilitar un entendimiento común del sistema, contribuyen significativamente a un diseño más coherente, esencial para el éxito en proyectos complejos.

Conclusiones parciales

La elaboración de este capítulo ha sido esencial para definir los objetivos estratégicos del proyecto y establecer una guía clara. A continuación, se presentan las conclusiones:

- Se identificaron 14 requisitos funcionales, 14 no funcionales, y se definieron 14 historias de usuario. Esto facilita la gestión del desarrollo en iteraciones manejables, permitiendo entregas concretas y medibles.
- Se utilizó el método de tarjetas CRC para definir de manera precisa la colaboración entre los componentes del sistema, asegurando interacciones efectivas.
- La elección del patrón *Modelo-Vista-Template* permitió una clara separación entre la interfaz de usuario y la lógica de negocio, facilitando futuras modificaciones y mantenimientos.
- La incorporación de los patrones GRASP y GOF organizó las responsabilidades de manera lógica y minimizó el acoplamiento entre componentes, mejorando la escalabilidad y robustez del sistema.
- Se generó un modelo de datos que estructuró lógicamente la información, garantizando una representación clara de las relaciones entre los elementos del sistema.
- Un análisis exhaustivo de las restricciones y variables relevantes aseguró la viabilidad y efectividad del sistema, ayudando a anticipar y mitigar posibles riesgos.
- La creación de un diagrama de despliegue fue crucial para visualizar la estructura del sistema y las interacciones entre sus componentes, facilitando la comprensión del diseño global.

Validación de IAHorDo para la planificación y generación automática del horario en la FTI.

La validación de un sistema de gestión del horario docente es un paso decisivo hacia la realización de una visión que transformará la organización de las instituciones educativas. En este capítulo, se abordará la transición de las ideas previamente delineadas hacia la validación de una herramienta práctica, diseñada para satisfacer las necesidades específicas de la facultad. Este enfoque metódico asegura la calidad del sistema y fomenta su adopción en el entorno educativo. La validación del sistema se llevará a cabo mediante pruebas en diferentes niveles y escenarios, incluyendo pruebas unitarias y pruebas de aceptación. Estas pruebas aseguran que cada módulo del sistema funcione de manera aislada y en conjunto, cumpliendo con los criterios de aceptación del cliente. Además, se aplicarán técnicas de desarrollo guiado por pruebas (TDD) y revisiones de código para mejorar continuamente la calidad del *software*. Durante este proceso, se tendrán en cuenta las actividades de ingeniería respondiendo a cada historia de usuario que se estableció. Finalmente, estas actividades asegurarán que el sistema esté alineado con los requerimientos definidos, optimizando la gestión del horario docente y elevando los estándares en la administración educativa.

3.1. Tareas de Ingeniería

Las tareas de ingeniería son actividades específicas y bien definidas que los desarrolladores y equipos técnicos ejecutan a lo largo del proceso de desarrollo de *software*, con el propósito de crear, modificar o mantener un sistema informático (Humble y Farley, 2010). Estas tareas están orientadas a cumplir objetivos concretos, tales como la implementación de funcionalidades, la resolución de incidencias, la optimización del rendimiento y la garantía de la calidad del *software*, fueron realizadas 34 tareas de ingeniería, para la primera iteración un total de 29 tareas y en la segunda iteración 5 tareas. A continuación, se presenta un ejemplo representativo de una historia de usuario junto con las correspondientes tareas de ingeniería que se llevaron a cabo en cada iteración. Para un examen más completo, el resto de las HU y sus respectivas tareas están disponibles en los apéndices de esta investigación.

Tareas de ingeniería para la I Iteración

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	Número de Historia de usuario: 3
Nombre de la tarea: Registrar asignatura	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 15 de agosto de 2024	Fecha de fin: 17 de agosto de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar una nueva asignatura en el sistema	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	Número de Historia de usuario: 3
Nombre de la tarea: Listar asignaturas	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 20 de agosto de 2024	Fecha de fin: 23 de agosto de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar las asignaturas registradas en el sistema	

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	Número de Historia de usuario: 3
Nombre de la tarea: Modificar asignatura	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 24 de agosto de 2024	Fecha de fin: 26 de agosto de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar las asignaturas registradas en el sistema	

Tabla 3.4. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	Número de Historia de usuario: 3

Continúa en la próxima página

Tabla 3.4. Continuación de la página anterior

Nombre de la tarea: Eliminar asignatura	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 27 de septiembre de 2024	Fecha de fin: 29 de septiembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita eliminar las asignaturas registradas en el sistema	

Tareas de ingeniería para la II Iteración

Tabla 3.5. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	Número de Historia de usuario: 13
Nombre de la tarea: Registrar marco horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 1 de octubre de 2024	Fecha de fin: 7 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar el marco horario en el sistema	

Tabla 3.6. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	Número de Historia de usuario: 13
Nombre de la tarea: Eliminar marco horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 7 de octubre de 2024	Fecha de fin: 8 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita eliminar el marco horario en el sistema	

Tabla 3.7. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	Número de Historia de usuario: 12
Nombre de la tarea: Generar horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 9 de octubre de 2024	Fecha de fin: 15 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla 3.7. Continuación de la página anterior

Descripción: Se implementa una funcionalidad que permita generar el horario mediante la gestión previa de una variedad de datos críticos como marco horario, balance de carga, carga docente, asignaturas, profesores, grupos, locales, actividades y afectaciones en el sistema

Tabla 3.8. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	Número de Historia de usuario: 12
Nombre de la tarea: Buscar horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 16 de octubre de 2024	Fecha de fin: 17 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita buscar el horario una vez generado introduciendo los criterios semana y grupo en el sistema	

Tabla 3.9. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	Número de Historia de usuario: 13
Nombre de la tarea: Exportar horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 18 de noviembre de 2024	Fecha de fin: 23 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita exportar el horario en formato xlsx una vez generado en el sistema	

Tabla 3.10. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	Número de Historia de usuario: 14
Nombre de la tarea: Visualizar horario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 1 de diciembre de 2024	Fecha de fin: 10 de diciembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita visualizar el horario una vez generado en el sistema	

Con las tareas de ingeniería claramente delineadas para el proceso de desarrollo de *software*, es esencial explicar los estándares de codificación seguidos. Esto es importante debido a que los estándares de codificación

establecen reglas y pautas que los desarrolladores deben seguir al escribir código.

3.1.1. Estándares de codificación

Los estándares de codificación en *Python* son fundamentales para garantizar la legibilidad, mantenibilidad y calidad del código. Estos estándares ayudan a los desarrolladores a seguir un conjunto de convenciones que facilitan la colaboración en proyectos, especialmente en entornos donde múltiples programadores trabajan juntos (Wilkinson, 2020). A continuación, se describen algunos de los principales estándares y buenas prácticas.

Uno de los documentos más importantes en el desarrollo de *Python* es Python Enhancement Proposal 8 (PEP 8, por sus siglas en inglés), que establece las guías de estilo para la escritura de código (Python, 2024). Este documento cubre aspectos como:

- Indentación: se recomienda usar 4 espacios por nivel de indentación y evitar mezclar espacios y tabulaciones.
- Longitud de línea: las líneas no deben exceder los 79 caracteres para facilitar la lectura en pantallas pequeñas.
- Nomenclatura:
 1. Las variables deben seguir la convención *snake_case*, donde todas las letras son minúsculas y las palabras están separadas por guiones bajos.
 2. Los nombres de las clases deben usar *CapWords* también conocido como *CamelCase*, donde cada palabra comienza con una letra mayúscula.

```
class CargaDocente(models.Model):
    profesor = models.OneToOneField(admin_models.Profesor, on_delete=models.CASCADE, null=True)
    carrera = models.TextField(null=True)
    grupos_conferencia = models.CharField(max_length=255, null=True, blank=True)
    grupos_clase_practica = models.CharField(max_length=255)
    responsabilidad = models.TextField(null=True)
    area = models.TextField(null=True, blank=True)
```

Figura 3.1. Nombre de clases bajo la convención CapWord

```
class Marco_Horario(View):
    @staticmethod
    def Error(request, Error=None, Success=None):
        if request.user.is_authenticated:
            if request.user.rol == "Planificador":
                pl = admin_models.Planificador.objects.get(userid=request.user)
                marco = models.Marco_Horario.objects.all()
                return render(request, "planificador/marco_horario/marco_horario.html", {
                    "opciones_turnos": list(range(0, 11)), "marco_horario": marco,
                    'Error': Error, 'Success': Success,
                    'anios': admin_models.Planificador_Anio.objects.filter(planificador_id=pl),
                })
            else:
                return home(request=request)
        else:
            return redirect("login")
```

Figura 3.2. Indentación del código basado en PEP8

- Espacios en blanco: se deben utilizar espacios alrededor de operadores y después de comas.
- *Docstrings*: se recomienda documentar funciones, clases y módulos utilizando cadenas de documentación *docstrings* con comillas triples. Esto ayuda a otros desarrolladores a entender el propósito y uso del código.

Además de seguir PEP 8, hay otras buenas prácticas que pueden mejorar la calidad del código como:

- Uso de herramientas: herramientas como *Flake8* y *Pylint* pueden ayudar a detectar errores y violaciones a las normativas de PEP 8.
- Manejo de excepciones: es importante utilizar excepciones para gestionar errores en lugar de depender de códigos de error, lo que mejora la robustez del código.
- Importaciones: las importaciones deben colocarse al principio del archivo, justo después de cualquier comentario o *docstring*. Deben estar antes de las variables y constantes globales siguiendo un orden específico.

```
1 from django.shortcuts import render, redirect
2 from django.views import View
3 from django.http import HttpResponse
4 from usuarios.views.perfilView import home
5 from . import models
6 from . import utils
7 from administrador import models as admin_models
8 import re
9
```

Figura 3.3. Importaciones de librerías bajo PEP8

1. Importaciones de la biblioteca estándar de *Python*.
2. Importaciones de bibliotecas de terceros o paquetes externos.
3. Importaciones locales de la aplicación o biblioteca.

- Nombres de módulos: los nombres de los módulos deben ser cortos y en minúsculas. Se pueden usar guiones bajos para mejorar la legibilidad si es necesario.
- Nombres de clases: casi sin excepción, los nombres de las clases deben seguir la convención *CapWord*, donde cada palabra comienza con una letra mayúscula. Las clases para uso interno pueden tener un guión bajo como prefijo.
- Nombres de funciones: las funciones deben estar en minúsculas, con las palabras separadas por un guión bajo. Los guiones bajos se aplican tanto como sea necesario para mejorar la legibilidad. Después del nombre de la función, se agrega la palabra clave "def" seguida de los parámetros entre paréntesis.

3.2. Fase de Pruebas

La fase de pruebas es una etapa crucial en el proceso de desarrollo de *software*, destinada a garantizar la calidad y el correcto funcionamiento del producto. Su objetivo principal es verificar y validar que el sistema cumpla con las especificaciones y funcionalidades definidas previamente. Las pruebas de *software* son un componente esencial del ciclo de desarrollo, aplicándose en diferentes niveles y escenarios para alcanzar diversos objetivos (Beck y Andres, 2004).

A continuación, se describirán los niveles de prueba que se llevarán a cabo en el sistema de planificación y generación del horario docente de la Facultad de Tecnologías Interactivas.

Tabla 3.11. Niveles de Pruebas. Fuente: Elaboración propia.

Nivel de prueba	Objetivo	Etapa de ejecución
Pruebas Unitarias:	Evaluar componentes individuales del <i>software</i> , como funciones o métodos.	Desarrollo
Pruebas de Aceptación:	Validar si el <i>software</i> satisface las necesidades del usuario final.	Despliegue

3.2.1. Pruebas Unitarias

Las pruebas unitarias se enfocan en verificar que las unidades individuales de código, tales como funciones o métodos, funcionen correctamente en aislamiento. El objetivo principal es asegurar que cada componente del *software* realice su tarea según lo previsto, incluyen reducir riesgos mediante herramientas integradas que permiten verificar que los comportamientos funcionales y no funcionales del componente coincidan con lo diseñado y especificado. Además, buscan generar confianza en la calidad del componente, detectar defectos y prevenir que estos se propaguen a niveles de pruebas superiores. En esta fase, se probarán pequeñas porciones de código independientemente, sin considerar sus interacciones con otras partes del sistema.

Django proporciona un marco para realizar pruebas unitarias, facilitando a los desarrolladores la creación de pruebas específicas para sus modelos, vistas y otros componentes de la aplicación (Tracey, 2010). Utilizando la clase *TestCase* de *django*, los desarrolladores pueden estructurar sus pruebas para evaluar el comportamiento esperado de cada unidad de código.

Por ejemplo, se llevarán a cabo pruebas unitarias en las clases, *Estudiantes* y *Asignaturas*. Estas pruebas permitirán:

- Probar que un planificador autenticado puede modificar correctamente una asignatura existente.
- Verificar que los campos de la asignatura se actualizan correctamente en la base de datos.
- Probar que se maneja adecuadamente la situación cuando se intenta modificar una asignatura con un ID que no existe.
- Verificar que el mensaje La asignatura no existe se muestra correctamente en la respuesta.
- Probar que se maneja adecuadamente la situación cuando se intenta modificar una asignatura con un nombre ya existente.
- Verifica que el mensaje El campo 'nombre' ya esta en uso se muestra correctamente en la respuesta.
- Probar que se manejen adecuadamente los errores de validación en el formulario de modificación de asignaturas.
- Probar que un usuario autenticado que no es planificador es redirigido a la página principal *home*.
- Probar que un usuario no autenticado es redirigido a la página de inicio de sesión *login*.

Implementación de la clase *TestCase* para realizar pruebas unitarias en el modelo *Asignatura*

```
def test_modificar_asignatura_exitoso(self):
    self.client.login(username='planificador', password='planificadormapass')
    post_data = {
        'assign_id': self.asignatura.id,
        'nombre': 'Física',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 200)
    asignatura_modificada = planificador_models.Asignatura.objects.get(id=self.asignatura.id)
    self.assertEqual(asignatura_modificada.nombre, 'Física')
    self.assertEqual(asignatura_modificada.abreviatura, 'FIS')
    self.assertEqual(asignatura_modificada.anno_academico, 1)
    self.assertEqual(asignatura_modificada.periodo, 1)
    self.assertEqual(asignatura_modificada.sesion, 'Tarde')
```

Figura 3.4. Prueba la modificación de una asignatura existente

```
def test_modificar_asignatura_no_existe(self):
    self.client.login(username='planificador', password='planificadpass')
    post_data = {
        'assign_id': 9999, # Un ID que no existe
        'nombre': 'Física',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 200)
```

Figura 3.5. Prueba la modificación de una asignatura que no existe.

```
def test_modificar_asignatura_nombre_ya_existente(self):
    otra_asignatura = planificador_models.Asignatura.objects.create(
        nombre='Física', abreviatura='FIS', anno_academico=1, periodo=1, sesion='Tarde'
    )
    self.client.login(username='planificador', password='planificadpass')
    post_data = {
        'assign_id': self.asignatura.id,
        'nombre': 'Física',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 200)
```

Figura 3.6. Prueba la modificación con un nombre de asignatura ya existente.

```
def test_modificar_asignatura_datos_invalidos(self):
    self.client.login(username='planificador', password='planificadpass')
    post_data = {
        'assign_id': self.asignatura.id,
        'nombre': '',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 200)
```

Figura 3.7. Prueba la modificación con datos inválidos.

```
def test_modificar_asignatura_usuario_no_planificador(self):
    usuario_no_planificador = User.objects.create_user(
        username='usuario', password='usuariopass', rol='Estudiante'
    )
    self.client.login(username='usuario', password='usuariopass')
    post_data = {
        'assign_id': self.asignatura.id,
        'nombre': 'Física',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 302) # Redirige a home
```

Figura 3.8. Prueba la modificación por un usuario que no es planificador.

```
def test_modificar_asignatura_no_autenticado(self):
    post_data = {
        'assign_id': self.asignatura.id,
        'nombre': 'Física',
        'abreviatura': 'FIS',
        'anio': '1',
        'periodo': '1',
        'sesion': 'Tarde'
    }
    response = self.client.post(self.modificar_asignatura_url, data=post_data)
    self.assertEqual(response.status_code, 302) # Redirige a login
```

Figura 3.9. Prueba la modificación por un usuario no autenticado.

- Probar que un administrador autenticado pueda eliminar correctamente un estudiante existente.
- Verificar que el Estudiante y su User asociado han sido eliminados de la base de datos.
- Probar que se maneja adecuadamente la situación cuando se intenta eliminar un estudiante con un ID que no existe.
- Verificar que el mensaje El estudiante no existe se muestra correctamente en la respuesta.
- Probar que un usuario autenticado que no es administrador es redirigido a la página principal *home*.
- Probar que un usuario no autenticado es redirigido a la página de inicio de sesión *login*.

Implementación de la clase TestCase para realizar pruebas unitarias en el modelo Estudiante

```
def test_eliminar_estudiante_exitoso(self):
    self.client.login(username='admin', password='adminpass')
    post_data = {
        'estudiante_id': self.estudiante.id
    }
    response = self.client.post(self.eliminar_estudiante_url, data=post_data)
    self.assertEqual(response.status_code, 200)
    self.assertFalse(admin_models.Estudiante.objects.filter(id=self.estudiante.id).exists())
    self.assertFalse(User.objects.filter(id=self.student_user.id).exists())
```

Figura 3.10. Prueba la eliminación exitosa de un estudiante:

```
def test_eliminar_estudiante_no_existe(self):
    self.client.login(username='admin', password='adminpass')
    post_data = {
        'estudiante_id': 9999 # Un ID que no existe
    }
    response = self.client.post(self.eliminar_estudiante_url, data=post_data)
    self.assertEqual(response.status_code, 200)
    self.assertIn('El estudiante no existe', response.content.decode())
```

Figura 3.11. Prueba la eliminación de un estudiante que no existe:

```
def test_eliminar_estudiante_usuario_no_admin(self):
    self.client.login(username='student', password='studentpass')
    post_data = {
        'estudiante_id': self.estudiante.id
    }
    response = self.client.post(self.eliminar_estudiante_url, data=post_data)
    self.assertEqual(response.status_code, 302) # Redirige a home
```

Figura 3.12. Prueba la eliminación por un usuario que no es administrador:

```
def test_eliminar_estudiante_no_autenticado(self):
    post_data = {
        'estudiante_id': self.estudiante.id
    }
    response = self.client.post(self.eliminar_estudiante_url, data=post_data)
    self.assertEqual(response.status_code, 302) # Redirige a login
```

Figura 3.13. Prueba la eliminación por un usuario no autenticado:

3.2.2. Resultados de las pruebas unitarias

El gráfico ilustra los resultados satisfactorios de las pruebas ejecutadas, mostrando que en todas las iteraciones se obtuvo el resultado esperado, que consiste en un identificador único de la acción a ejecutar por el agente, entre todas las acciones posibles.

```
(venv) C:\Users\ismae\Desktop\TD\VersionFinalIAHorDo>py manage.py test
Found 33 test(s).
Creating test database for alias 'default'...
importando librerias
definiendo clase
importando librerias
definiendo clase
System check identified no issues (0 silenced).
.....Estudiante matching query does not exist.
.....
-----
Ran 33 tests in 31.353s

OK
Destroying test database for alias 'default'...
```

Figura 3.14. Resultados de las pruebas:

Se realizaron 33 pruebas unitarias abarcando diferentes aspectos y funcionalidades del modelo Asignatura y Estudiantes. Durante las primeras iteraciones se detectaron varios errores, como la posibilidad de introducir números en el campo nombre, la falta de obligatoriedad del campo abreviatura y la aceptación de datos inválidos en el proceso de modificación de asignaturas. Por ejemplo, al modificar una asignatura, se probó con un nombre ya existente y con datos vacíos en el campo nombre, asegurando que el sistema maneje adecuadamente estos casos. Adicionalmente, se verificó el proceso de eliminación de estudiantes, incluyendo casos como intentar eliminar un estudiante que no existe y probar la eliminación por usuarios no administradores y no autenticados. En las iteraciones finales, los resultados de todas las pruebas fueron exitosos, validando la correcta funcionalidad y estabilidad del código. Estas pruebas mostraron una cobertura completa de los requisitos establecidos, identificando posibles errores desde el inicio y garantizando un *software* confiable y de calidad. El éxito en las pruebas unitarias ahorra tiempo y recursos al prevenir problemas futuros, proporcionando una base sólida para iteraciones y mejoras continuas.

3.2.3. Pruebas de Aceptación

Las pruebas de aceptación se centran en verificar que el sistema en su totalidad cumple con los requisitos funcionales y no funcionales establecidos por el cliente o las partes interesadas. El objetivo principal es garantizar que el *software* sea útil y satisfaga las necesidades del usuario final. Este tipo de prueba puede considerarse como un enfoque de caja negra, donde cada prueba representa una salida esperada del sistema (Crispin y Gregory, 2009).

Es responsabilidad del cliente validar la corrección del sistema y tomar decisiones basadas en los resultados obtenidos durante estas pruebas. Al realizar pruebas de aceptación, se busca confirmar que todas las funcionalidades implementadas cumplen con las expectativas y requisitos definidos previamente, asegurando así la satisfacción del usuario final con el producto entregado.

Pruebas de aceptación para la Iteración I

En la primera iteración, se desarrollaron un total de 16 casos de prueba de aceptación. A continuación, se presenta la descripción detallada de uno de estos casos:

Tabla 3.12. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario: 3
Nombre: Registrar asignatura	
Descripción: Confirma que los usuarios con roles de planificador pueden registrar la información de las asignaturas en el sistema.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Pasos de ejecución: Para registrar una asignatura: 1. El planificador debe acceder a la sección de 'Asignaturas'. 2. El planificador debe presionar el ícono de registrar asignatura. 3. El sistema muestra un formulario para introducir los datos requeridos (nombre asignatura, abreviatura, año académico, sesión y periodo). 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Registrar'.	
Resultados esperados: • Si los datos introducidos son válidos, se registra la nueva asignatura en el sistema y se incluye en la lista de asignaturas. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.	

Tabla 3.13. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario: 3
Nombre: Listar asignaturas	
Descripción: Confirma que los usuarios con rol de planificador puedan listar las asignaturas en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una asignatura registrada.	

Continúa en la próxima página

Tabla 3.13. Continuación de la página anterior

Pasos de ejecución: Para listar asignatura: 1. El planificador debe acceder a la sección de 'Asignaturas'. 2. El sistema muestra una tabla de las asignaturas donde se recoge la siguiente información (nombre asignatura, abreviatura, año académico, sesión y periodo) .
Resultados esperados: • Los datos de las asignaturas registradas, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay asignaturas registradas.

Tabla 3.14. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario: 3
Nombre: Modificar asignatura	
Descripción: Confirma que los usuarios con rol de planificador puedan modificar asignatura en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una asignatura asociada a un profesor registrado en el sistema.	
Pasos de ejecución: Para modificar asignatura: 1. El planificador debe acceder a la sección de 'Asignaturas'. 2. El planificador debe presionar el ícono de modificar asignatura en la lista de cargas docentes. 3. El sistema muestra un formulario con la información actual de la asignatura. 4. El planificador modifica el campo deseado, los datos son (nombre de asignatura, abreviatura, piso, capacidad, designación). 5. El planificador presiona el botón Actualizar.	
Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay asignaturas registradas.	

Tabla 3.15. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario: 3
Nombre: Eliminar asignatura	
Descripción: Confirma que los usuarios con rol de planificador puedan eliminar asignatura en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una asignatura registrada en el sistema.	
Pasos de ejecución: Para eliminar asignatura: 1. El planificador debe acceder a la sección de 'Asignaturas'. 2. El planificador debe presionar el ícono de eliminar asignatura en la lista de asignaturas. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar la 'asignatura'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.	
Resultados esperados: • Los datos registrados de la asignatura seleccionada son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.	

Pruebas de aceptación para la Iteración II

En la segunda iteración, se desarrollaron un total de 4 casos de prueba de aceptación. A continuación, se presenta la descripción detallada de estos casos:

Tabla 3.16. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU11_P11	Historia de usuario: 11
Nombre: Registrar marco horario	
Descripción: Confirma que los usuarios con roles de planificador pueden registrar el marco horario	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema.	

Continúa en la próxima página

Tabla 3.16. Continuación de la página anterior

Pasos de ejecución: Para registrar marco horario: 1. El planificador debe acceder a la sección de 'Marco horario'. 2. El planificador debe presionar el ícono de registrar marco horario. 3. El sistema muestra un formulario donde se insertan los siguientes datos requeridos (tipo de curso, número de turnos, hora de inicio, duración, tiempo entre turnos, duración mediodía) 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Aceptar'.
Resultados esperados: • Si los datos introducidos son válidos, se registra el nuevo marco de horario en el sistema y se incluye en la lista de marcos de horarios. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.

Tabla 3.17. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU11_P11	Historia de usuario: 11
Nombre: Eliminar el marco de horario	
Descripción: Confirma que los usuarios con rol de planificador puedan eliminar el marco horario en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un marco de horario registrado en el sistema.	
Pasos de ejecución: Para eliminar el marco horario: 1. El planificador debe acceder a la sección de 'Marco horario'. 2. El planificador debe presionar el ícono de eliminar el marco de horario en la lista de el marco horarios. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar la 'el marco horario'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.	

Continúa en la próxima página

Tabla 3.17. Continuación de la página anterior

Resultados esperados: • Los datos registrados del marco horario seleccionado son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.
--

Tabla 3.18. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU12_P12	Historia de usuario: 12
Nombre: Generar Horario	
Descripción: Confirma que los usuarios con roles de planificador pueden gestionar la información necesaria para la generación automática del horario docente	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Los datos deben estar previamente introducidos.	
Pasos de ejecución: Para generar horario con una nueva asignatura: 1. El planificador debe acceder a la sección de 'Horario'. 2. El planificador debe introducir en el formulario la información requerida como número de la semana, periodo y año académico. 3. El planificador debe presionar el botón generar. 4. Una vez generado el horario el planificador debe presionar el botón Aceptar.	
Resultados esperados: El horario debe generarse luego de haber introducido los datos correspondientes	

Tabla 3.19. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU13_P13	Historia de usuario: 13
Nombre: Exportar horario	
Descripción: Confirma que los usuarios con rol de planificador puedan exportar horario en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un horario registrado en el sistema.	

Continúa en la próxima página

Tabla 3.19. Continuación de la página anterior

Pasos de ejecución: Para exportar horario: 1. El planificador debe acceder a la sección de 'Horario'. 2. El planificador debe presionar el ícono de exportar horario. 3. El sistema realiza la exportación de los datos en un archivo formato xlsx.
Resultados esperados: • Los datos registrados del horario son exportados.

Tabla 3.20. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU14_P14	Historia de usuario: 14
Nombre: Visualizar horario	
Descripción: Confirma que los usuarios con roles de planificador pueden visualizar el horario en el sistema	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Los datos deben estar previamente introducidos.	
Pasos de ejecución: Para visualizar el horario 1. El usuario debe iniciar sesión en el sistema en correspondencia con el rol de usuario. 2. El usuario debe acceder a la sección 'Horario'. 3. El usuario debe completar el formulario con los datos necesarios para que se muestre el horario dado un año académico, una semana o un grupo. 4. El sistema muestra el horario automáticamente después de hacer clic en el botón buscar.	
Resultados esperados: Se debe visualizar el horario docente generado correspondiente a los datos introducidos.	

Análisis de las pruebas de aceptación

El análisis de las pruebas de aceptación realizadas en cada iteración permitió obtener porcentajes de resultados satisfactorios y no satisfactorios. A continuación, se presentan los hallazgos correspondientes a cada iteración mediante un gráfico de barras.

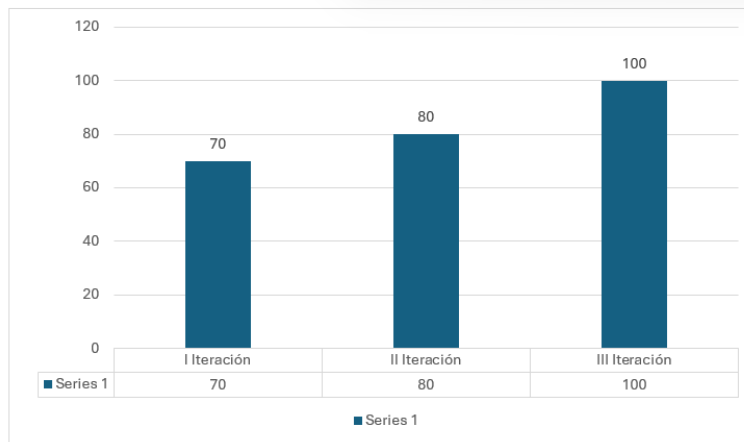


Figura 3.15. Resultados de las pruebas de aceptación

Para la primera iteración, se logró un 70 % de resultados satisfactorios y un 30 % de resultados no satisfactorios. Se identificó que el sistema no editaba correctamente, lo que llevó a la corrección del error, transformando finalmente la prueba en satisfactoria. En la segunda iteración, se alcanzó un 80 % de pruebas satisfactorias y un 20 % de resultados no satisfactorios. En este caso, se detectó que el caso de prueba relacionado con la visualización del horario generado no mostraba correctamente el horario. Se realizaron las modificaciones necesarias, lo que resultó en la satisfacción del caso de prueba. Para la tercera iteración, se obtuvo un 100 % de resultados satisfactorios, sin ningún resultado no satisfactorio, lo que demuestra el cumplimiento adecuado de los requisitos establecidos para estas iteraciones.

Conclusiones parciales

En este capítulo, se han consolidado los avances en el desarrollo del "Sistema para la planificación y generación automática de horarios docentes en la Facultad de Tecnologías Interactivas", llegando a las siguientes conclusiones:

- La aplicación de estándares de codificación ha resultado en un código fuente estructurado, claro y de fácil comprensión. Esto ha contribuido a obtener un sistema funcional y completamente operativo, garantizando la consistencia, calidad y mantenibilidad del código.
- Las tareas de ingeniería facilitadas han permitido la implementación efectiva de las funcionalidades solicitadas. Este enfoque estructurado ha ofrecido una organización bien definida y orientada a lograr resultados concretos.
- La realización de pruebas unitarias de forma automatizada ha permitido validar la funcionalidad interna de los módulos del sistema. Estas pruebas han sido fundamentales para garantizar que el sistema sea resistente a amenazas y vulnerabilidades.

- Las pruebas de aceptación han confirmado la alineación del sistema con los requerimientos específicos de la FTI. Esto ha asegurado que se satisfacen las necesidades y expectativas del usuario final, validando que el sistema cumple con los criterios establecidos.

Al concluir la investigación, se puede afirmar que los objetivos planteados han sido cumplidos, permitiendo arribar a las siguientes conclusiones:

1. La definición de los fundamentos teóricos ha establecido un marco conceptual sólido para los sistemas de planificación y generación automática de horarios académicos, facilitando una comprensión integral del problema y sustentando la propuesta.
2. Se realizó una descripción detallada del proceso actual de planificación y confección de horarios en la Facultad de Tecnologías Interactivas, identificando sus limitaciones y oportunidades de mejora. Esto proporcionó una base clara para el desarrollo del sistema.
3. La evaluación de soluciones homólogas permitió identificar buenas prácticas que son aplicables al contexto de la facultad, destacando la necesidad de una solución personalizada para abordar las especificidades locales.
4. La selección de algoritmos de optimización adecuados se basó en una investigación exhaustiva, lo que permitió identificar técnicas que se ajustan a las necesidades del sistema, garantizando su eficiencia y efectividad.
5. Se implementó un sistema informático que automatiza la planificación de horarios docentes, mejorando significativamente la gestión del tiempo y los recursos en la FTL.
6. La realización de pruebas de software siguiendo la metodología seleccionada validó la funcionalidad y la calidad del sistema. Las pruebas unitarias y de aceptación confirmaron que el sistema cumple con los requisitos establecidos, asegurando su resistencia a amenazas y vulnerabilidades.

Recomendaciones

Se recomienda adaptar el sistema para gestionar no sólo el balance de carga de una semana sino todo el periodo, también integrar funcionalidades específicas para otras facultades de la Universidad de las Ciencias Informáticas, abordando sus particularidades y necesidades como la inclusión de otras carreras. Además de generar el horario docente de todo el periodo.

Referencias bibliográficas

- ABRAMSON, D., 1991. Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science*. Vol. 37, n.º 1, págs. 98-113. Url: <https://pubsonline.informs.org/doi/10.1287/mnsc.37.1.98> (vid. págs. 38, 39, 50).
- ABREU GUERRA, Liana, 2012. *Sistema de gestión para la planificación docente en la Facultad Regional Mártires de Artemisa*. Artemisa. Disponible desde DOI: TrabajodeDiploma. Tesis doctoral. Universidad de las Ciencias Informáticas (vid. pág. 16).
- APPADDITIO, 2014. *Additio App*. Plataforma integral de gestión escolar de nueva generación. Url: <https://additioapp.com> (vid. pág. 19).
- AREA MOREIRA, Manuel; GUTIÉRREZ MARTÍN, Alfonso y VIDAL FERNÁNDEZ, Fernando, 2012. *Alfabetización digital y competencias informacionales*. Barcelona, Madrid: Ariel ; Fundación Telefónica (vid. pág. 1).
- BECK, Kent y ANDRES, Cynthia, 2004. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional (vid. págs. 28, 53, 54, 60, 61, 63, 77).
- BLANCO, Castañet y ALEJANDRO, Richard, 2022. *Módulo de generación de horarios docentes para el Sistema de Trabajo Educativo en la Facultad I*. La Habana. Disponible desde DOI: TrabajodeDiploma. Tesis doctoral. Universidad de las Ciencias Informáticas (vid. pág. 16).
- BRADLEY, Mr Sean, 2021. *Design Patterns in Python: Common GOF*. ISBN 9798702082608 (vid. pág. 65).
- BUILES, Jovani Alberto Jiménez; BEDOYA, Diego León Ramírez y BEDOYA, John Willian Branch, 2019. Metodología de desarrollo de software para plataformas educativas robóticas usando ROS-XP. *Revista Politécnica* [online]. Vol. 15, n.º 30, págs. 55-69 [visitado 2024-11-18]. ISSN 2256-5353. Disponible desde DOI: 10.33571/rpolitec.v15n30a6 (vid. pág. 62).
- CARCÍA, Mora y MIGUEL, Antonio, 2009. *Resolución del problema militar de búsqueda de camino óptimo multiobjetivo mediante el uso de algoritmos de optimización basados en colonias de hormigas*. Granada. Disponible desde DOI: Tesisdoctoral. Tesis doctoral. Universidad de Granada (vid. pág. 13).
- CASTRO CÁCERES, Ricardo Andrés y RODRÍGUEZ ROJAS, Felipe Francisco, 2017. *Diseño universal para el aprendizaje y co-enseñanza: estrategias pedagógicas para una educación inclusiva*. Primera edición. Providencia, Santiago de Chile, Santiago: RIL editores ; Ediciones Universidad Santo Tomás (vid. pág. 1).

- CHÁVEZ BÓSQUEZ, Óscar; POZOS PARRA, Pilar y GÓMEZ RAMOS, José Luis, 2015. Búsqueda tabú con criterio de aspiración probabilístico aplicada a la generación de horarios escolares. *Revista de Matemática: Teoría y Aplicaciones* [online]. Vol. 22, n.º 1, págs. 153-177 [visitado 2024-11-17]. ISSN 2215-3373, 2215-3373. Disponible desde: <https://dialnet.unirioja.es/servlet/articulo?codigo=5232938> (vid. pág. 13).
- CONTRIBUITORS, 2024. *Tipos De Usuarios Del Sistema*. Url: <https://www.uco.es/grupos/eatco/cev/abril98/paranin/manual/usuarios.html> (vid. págs. 24, 54).
- CRISPIN, Lisa y GREGORY, Janet, 2009. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley. ISBN 9780321534460. Google-Books-ID: 3UdsAQAAQBAJ (vid. pág. 82).
- CRUZ, Andres, 2023. *Primeros pasos con Bootstrap 5: Tu guía para iniciar en el desarrollo en Bootstrap de una manera práctica*. Andres Cruz (vid. pág. 26).
- DJANGO, 2024. *Django Project*. Django documentation | Django documentation. Url: <https://docs.djangoproject.com/en/5.1/> (vid. pág. 26).
- EDUCACIÓN, Ministerio Superior de, 2022. *Resolución 47/2022. Reglamento del proceso docente y de dirección del trabajo para la carreras universitarias* [online] [visitado 2024-11-17]. Disponible desde: <https://siteal.iiep.unesco.org/bdnp/4355/resolucion-472022-reglamento-proceso-docente-direccion-trabajo-carreras-universitarias> (vid. pág. 1).
- EDUCARIA, 2022. *Educaria*. Untis programa líder mundial en la realización automática de horarios para colegios y universidades. Url: <https://www.educaria.com/horarios/> (vid. pág. 18).
- FRIMAN, Katia Maria Cobas, 2012. Título: Algoritmos genéticos aplicados a problemas de planificación (vid. pág. 12).
- GOLDBERG, David Elliott y GOLDBERG, David E., 2012. *Genetic algorithms in search, optimization, and machine learning*. 30. print. Boston: Addison-Wesley (vid. págs. 12, 42).
- HTML5, 2019. *OpenWebinars.net*. Qué es HTML5: definición y funcionamiento | OpenWebinars. Url: <https://openwebinars.net/blog/que-es-html5/> (vid. pág. 25).
- HUMBLE, Jez y FARLEY, David, 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education. ISBN 9780321670229 (vid. pág. 71).
- JAVASCRIPTS, 2024. *¿Qué es JavaScript? - Aprende desarrollo web* | MDN. Url: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript (vid. pág. 25).
- JAVIER González Millán, José y TERESA Rodríguez Díaz, Miryam, 2019. *Manual práctico de planeación estratégica*. Ediciones Díaz de Santos (vid. pág. 8).
- LAMARCA, Montserrat Estañol, 2016. *Artifact-centric Business Process Models in UML : Specification and Reasoning*. Universitat Politècnica de Catalunya (vid. pág. 21).

- LASATER, Christopher G. y LASATER, 2006. *Design Patterns*. Jones & Bartlett Publishers. ISBN 9781449612887. Google-Books-ID: vmZ5lvTyEYEC (vid. pág. 65).
- LÓPEZ, Jose Carlos, 2010. *Adictos al trabajo*. Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA. Url: <https://adictosaltrabajo.com/2010/10/07/jgap/> (vid. pág. 51).
- MCFARLAND, David Sawyer, 2012. *CSS3: the missing manual*. 3rd ed. Sebastopol, CA: O'Reilly (vid. pág. 26).
- MICHALEWICZ, Zbigniew, 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Science & Business Media (vid. pág. 12).
- MICHALEWICZ, Zbigniew, 1999. *Genetic algorithms + data structures: = evolution programs ; with 36 tables*. 3., rev. and extended ed., 1. corrected printing. Berlin Heidelberg: Springer (vid. págs. 44, 45).
- MUNIER, Nolberto J., 1986. *Aplicaciones de la programación lineal: ejemplos de esta técnica para la resolución de diversos problemas industriales y de planeamiento*. Astrea (vid. pág. 13).
- NAVARRO, Mirta Elizabeth; MORENO, Marcelo P.; ARANDA, Juan; PARRA, Lorena; RUEDA, Jose R. y PANTANO, Juan Cruz, 2017. Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. En: *Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles*. Url: <http://sedici.unlp.edu.ar/handle/10915/62077> (vid. pág. 63).
- NELSON, Asdrual Henry; FUENTES, Fernando José Artigas y CANDELARIA, Manuel Rafael Hernández, 2021. LA PLANIFICACIÓN DOCENTE UTILIZANDO ALGORITMOS GENÉTICOS : PLANIFICACIÓN DOCENTE. *Didáctica y Educación ISSN 2224-2643*. Vol. 12, n.º 4, págs. 1-11. Url: <https://revistas.ult.edu.cu/index.php/didascalía/article/view/1074> (vid. págs. 11, 12, 15).
- ORTEGA, Cristina, 2023. *QuestionPro*. Programación lineal: Qué es, usos y pasos para realizarla. Url: <https://www.questionpro.com/blog/es/programacion-lineal/> (vid. pág. 13).
- PENADÉS, Mª Carmen y LETELIER TORRES, Patricio Orlando, 2006. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica administrativa*. Vol. 5, n.º 26, págs. 1. Url: <https://dialnet.unirioja.es/servlet/articulo?codigo=1983605> (vid. págs. 28, 53).
- PENALARA, Software, 2022. *Blog de organización y optimización de horarios*. Horarios óptimos para la enseñanza. Url: <https://www.penalara.com/blog/index.php/2022/04/13/horarios-optimos-para-la-ensenanza/> (vid. pág. 17).
- PERALTA, Manuel, 2000. *Sistema de Información*. El Cid Editor apuntes. Url: <http://www.monografias.com/trabajos7/sisinf/sisinf.shtml>. (vid. pág. 7).
- PEREDA, Carlos, 2000. *El concepto de heurística en las ciencias y las humanidades*. Siglo XXI (vid. pág. 12).

- POSTGRESQL, 2024. *POSTGRESQL.ORG.ES*. Url: <https://e-mc2.net/es/projects/postgresql.org/> (vid. pág. 27).
- PRESSMAN, Roger S., 2006. *Ingeniería del software: un enfoque práctico*. McGraw-Hill (vid. págs. 21, 22).
- PRESSMAN, Roger S., 2010. *INGENIERIA DE SOFTWARE*. McGraw-Hill Interamericana de España S.L. ISBN 9786071503145. Google-Books-ID: deuwcQAACAAJ (vid. págs. 28, 29).
- PYTHON, 2024. *Python.org*. Welcome to Python.org. Url: <https://www.python.org/> (vid. págs. 25, 75).
- RANJAN, Awanish, 2021. *Building Websites with Django: Build and Deploy Professional Websites with Python Programming and the Django Framework (English Edition)*. BPB Publications. ISBN 9789389328288. Google-Books-ID: SWEeEAAAQBAJ (vid. pág. 63).
- REYES, M. C. Hugo Román; HUGO, Román Reyes M. C.; GALAVIZ, M. C. Juan Carlos Manzanarez; CARLOS, Manzanarez Galaviz M. C. Juan; ARMENTA, Juan José Tapia y JOSÉ, Tapia Armenta Dr Juan, 2012. *Optimización Con Algoritmos Genéticos: Aplicando Cómputo Multi-Núcleo*. Editorial Academica Espanola (vid. pág. 37).
- RIVADENEIRA MOLINA, Silvia; VILANOVA, Gabriela; MIRANDA, María y CRUZ, Diana, 2013. El modelado de requerimientos en las metodologías ágiles. En: *El modelado de requerimientos en las metodologías ágiles* [online] [visitado 2024-11-18]. Disponible desde: <http://sedici.unlp.edu.ar/handle/10915/27196> (vid. pág. 53).
- RNF, 2024. *Gluo*. Requisitos no funcionales: ¿Por qué son importantes? Url: <https://gluo.mx/blog/requisitos-no-funcionales-por-que-son-importantes> (vid. pág. 51).
- ROBBINS, Stephen Paul, 2002. *Fundamentos de administración*. Distrito Federal: Pearson Educación (vid. pág. 8).
- ROSENBERG, Don y STEPHENS, Matt, 2008. *Extreme Programming Refactored: The Case Against XP*. Apress (vid. pág. 55).
- SUÁREZ, Víctor F.; GUERRERO, Álvaro y CASTRILLÓN, Omar D., 2013. Programación de Horarios Escolares basados en Ritmos Cognitivos usando un Algoritmo Genético de Clasificación No-dominada, NSGA-II. *Información tecnológica*. Vol. 24, n.º 1, págs. 103-114. Url: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642013000100012&lng=en&nrm=iso&tlng=en (vid. págs. 12, 24, 38).
- TOBÓN, Luis Miguel Echeverry y CARMONA, Luz Elena Delgado, 2007. *Caso práctico de la metodología ágil XP al desarrollo de software*. Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería de Sistemas y Computación (vid. pág. 63).
- TRACEY, Karen M., 2010. *Django 1.1 Testing and Debugging*. Packt Publishing Ltd. ISBN 9781847197573. Google-Books-ID: mWG1AAAAQBAJ (vid. pág. 78).

- URBANO, Claudio Ariel, 2006. *Técnicas Para Investigar 2*. Crádoba, Argentina: Brujas (vid. pág. 4).
- VISUALPARADIGM, 2024. *What's New in Visual Paradigm?* Url: <https://www.visual-paradigm.com/whats-new/> (vid. pág. 23).
- VISUALSTUDIOCODE, 2024. *Why Visual Studio Code?* Url: <https://code.visualstudio.com/docs/editor/whyvscode> (vid. pág. 24).
- WILKINSON, Christopher, 2020. *Codificación con Python: Una guía introductoria para que los principiantes aprendan y comiencen a codificar con Python*(Libro En Español/Self Publishing Spanish Book Version). Independently Published. ISBN 9781654180058. Google-Books-ID: HWovzQEACAAJ (vid. pág. 75).
- ZABALZA, Miguel Ángel, 2023. *La enseñanza universitaria: El escenario y sus protagonistas*. Narcea Ediciones (vid. pág. 7).

Apéndice

Entrevistas

Anexo 1 [anex:anexo1]. Entrevista realizada al personal involucrado en el proceso de la planificación y confección del horario docente de la Facultad de Tecnologías Interactivas para recopilar información cualitativa sobre sus experiencias, necesidades y expectativas respecto al horario docente.

1. ¿Podría describir el proceso de planificación del horario docente de la Facultad de Tecnologías Interactivas?
2. ¿Quiénes son los principales actores que participan en este proceso y cuál es el rol de cada uno?
3. ¿Qué nivel de complejidad presenta la planificación y confección del horario docente, y cuáles son los factores que contribuyen a esta complejidad?
4. ¿Qué tan flexible es el horario docente actual y cómo se manejan las solicitudes de cambios por parte de los docentes?
5. ¿Cómo se adaptan a situaciones imprevistas, como la ausencia de un docente o cambios en la matrícula de los estudiantes?

Anexo 2. Entrevista realizada al personal involucrado en el proceso de la planificación y confección del horario docente de la FTI, con el objetivo de realizar la captura de los requerimientos funcionales.

1. ¿Qué tipo de información es esencial recopilar sobre las asignaturas, docentes y grupos para cada año académico?
2. ¿Cuáles son los principales inconvenientes que pueden surgir durante el proceso de planificación del horario docente?
3. ¿Qué datos específicos son necesarios para desarrollar el modelo de datos del sistema que se propone implementar?

Anexo 3. Entrevista realizada a los profesores principales de año de la FTI.

Profesores Principales de Año:

1. ¿Considera que la implementación de un sistema para la planificación y generación automática del horario docente en la Facultad de Tecnologías Interactivas sería útil y necesaria?
Sí ____ No ____ Tal vez ____

2. ¿Qué grado de importancia le asigna a la creación de un sistema para la generación de horarios docentes en la Facultad de Tecnologías Interactivas?
3. ¿Cree que el sistema de planificación y generación automática del horario docente desarrollado para la FTI podría ser aplicable a otras facultades de la universidad?

Historias de Usuario

Tabla B.1. Historia de usuario # 4

Historia de usuario	
Número: 4	Nombre: Gestionar balance de carga
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador gestiona el balance de carga mediante operaciones de inserción, modificación, eliminación y visualización de datos. Escenario 1: Insertar balance de carga El planificador al adicionar nuevos elementos, como: • Asignatura • Frecuencia de clases • Tipología • Se asegura de que el balance esté actualizado y refleje adecuadamente las modalidades de enseñanza. Escenario 2: Visualizar Balance de carga El planificador tendrán la capacidad de consultar información detallada sobre cualquier profesor registrado en el sistema. Esta funcionalidad permitirá: • La funcionalidad de búsqueda permite acceder a la información existente, facilitando el listado de asignaturas, lo cual es crucial para la toma de decisiones. Escenario 3: Modificar Balance de carga El planificador podrá actualizar la información de los profesores existentes en el sistema. Este proceso implicará que: • Se podrá ajustar datos como las frecuencias para adaptarse a las necesidades del currículo.	

Continúa en la próxima página

Tabla B.1. Continuación de la página anterior

Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de balance de carga.

Interfaz:

Asignatura	Curso	Frecuencia	Semana	Tipología	Horas Clases	Acciones
Matemática Discreta I	2023-2024	3	1	Presencial	27h TP	[Iconos de gestión]
Introducción a la Programación I	2023-2024	4	1	Presencial	64h TP	[Iconos de gestión]
Educación Física I	2023-2024	2	1	Presencial	32h TP	[Iconos de gestión]
Inglés	2023-2024	3	1	Presencial	28h TP	[Iconos de gestión]

Tabla B.2. Historia de usuario # 5

Historia de usuario	
Número: 5	Nombre: Gestionar grupos
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla B.2. Continuación de la página anterior

Descripción: El planificador podrá gestionar los grupos mediante el listado, adición, modificación y eliminación de datos relevantes.

Escenario 1: Insertar grupo El planificador podrá crear nuevos registros de grupos en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente:

- Nombre
- Matrícula
- Año académico
- Tipo de curso

Escenario 2: Listar grupo El planificador tendrá la capacidad de listar información detallada sobre cualquier grupo registrado en el sistema. Esta funcionalidad permitirá:

- Acceso rápido a los datos básicos del grupo (nombre, matrícula, año académico y tipo de curso).
- Listado de la estructura organizativa asignada al grupo.

Escenario 3: Modificar grupo El planificador podrá actualizar la información de los grupos existentes en el sistema. Este proceso implicará:

- Edición de campos como (nombre, matrícula, año académico y tipo de curso)
- Corregir errores tipográficos o inconsistencias en los datos.

Escenario 4: Eliminar grupo El planificador dispondrá de la opción para eliminar completamente un registro de grupo del sistema. Este proceso implicará:

- Confirmación de la acción antes de proceder con la eliminación.
- Eliminación completa del registro del grupo en todas las tablas relacionadas.
- Actualización automática de referencias cruzadas para mantener la integridad de los datos.

Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de grupos.

Interfaz:

The screenshot shows a web interface titled "Grupos". At the top left is a button "Registrar grupo". Below it is a search bar with a "Mostrar" dropdown, a text input for "Entradas", and a "Buscar" button. The main content is a table with the following data:

Nombre	Matrícula	Año académico	Acciones
IDF4101	32	Primer Año	[Iconos de edición y eliminación]
IDF4102	40	Primer Año	[Iconos de edición y eliminación]
IDF4103	35	Primer Año	[Iconos de edición y eliminación]
IDF4104	39	Primer Año	[Iconos de edición y eliminación]

At the bottom right, there is a pagination bar showing "Página 1 de 1".

Tabla B.3. Historia de usuario # 6

Historia de usuario	
Número: 6	Nombre: Gestionar locales
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador podrá listar, adicionar, modificar y eliminar datos de los locales. Escenario 1: Insertar local El planificador podrá crear nuevos registros de locales en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente: • Nombre • Abreviatura • Piso • Matrícula • Designación • Disponibilidad Escenario 2: Listar local El planificador tendrán la capacidad de listar información detallada sobre cualquier local registrado en el sistema. Esta funcionalidad permitirá: • Acceso rápido a los datos básicos del local (nombre, abreviatura, piso, matrícula, designación y disponibilidad). • Listado de la estructura organizativa asignada al local. Escenario 3: Modificar local El planificador podrá actualizar la información de los locales existentes en el sistema. Este proceso implicará: • Edición de campos como (nombre, abreviatura, piso, matrícula, designación y disponibilidad) • Corregir errores tipográficos o inconsistencias en los datos. Escenario 4: Eliminar local El planificador dispondrá de la opción para eliminar completamente un registro de local del sistema. Este proceso implicará: • Confirmación de la acción antes de proceder con la eliminación. • Eliminación completa del registro del local en todas las tablas relacionadas. • Actualización automática de referencias cruzadas para mantener la integridad de los datos.	
Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de locales.	

Continúa en la próxima página

Tabla B.3. Continuación de la página anterior

Interfaz:

Locales

Registrar local

Mostrar Entradas Buscar

Nombre	Facultad	Piso	Capacidad	Designación	Disponibilidad	Acciones
Salón 301	Facultad de Tecnologías Interactivas	Tercer Piso	40	Salón de Conferencia	Disponible	  
Aula 303	Facultad de Tecnologías Interactivas	Tercer Piso	35	Aula teórica	Dispoible	  
Salón 302	Facultad de Tecnologías Interactivas	Tercer Piso	40	Salón de Conferencia	Disponible	  
Laboratorio 301	Facultad de Tecnologías Interactivas	Tercer Piso	39	Laboratorio	No disponible	  

Página de 1

Tabla B.4. Historia de usuario # 7

Historia de usuario	
Número: 7	Nombre: Gestionar actividades
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla B.4. Continuación de la página anterior

Descripción: El planificador tendrá la capacidad de gestionar información sobre las actividades del sistema. Esto implica insertar nuevas actividades, modificar información existente y listar las actividades que se impartirán por año académico.

Escenario 1: Insertar actividad El planificador podrá crear nuevos registros de actividades en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente:

- Nombre
- Tipo de actividad
- Año académico
- Turno
- Día

Escenario 2: Listar actividad El planificador tendrán la capacidad de listar información detallada sobre cualquier actividad registrado en el sistema. Esta funcionalidad permitirá:

- Acceso rápido a los datos básicos de la actividad (nombre de la actividad, tipo de actividad, año académico, turno y día).
- Listado de la estructura organizativa asignada a la actividad.

Escenario 3: Modificar actividad El planificador podrá actualizar la información de las actividades existentes en el sistema. Este proceso implicará:

- Edición de campos como (nombre de la actividad, tipo de actividad, año académico, turno y día)
- Corregir errores tipográficos o inconsistencias en los datos.

Escenario 4: Eliminar actividad El planificador dispondrá de la opción para eliminar completamente un registro de actividad del sistema. Este proceso implicará:

- Confirmación de la acción antes de proceder con la eliminación.
- Eliminación completa del registro de la actividad en todas las tablas relacionadas.
- Actualización automática de referencias cruzadas para mantener la integridad de los datos.

Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de actividades.

Continúa en la próxima página

Tabla B.4. Continuación de la página anterior

Interfaz:

Actividades

Registrar actividad

Mostrar Entradas

Buscar

Nombre	Tipo de Actividad	Año académico	Periodo	Facultad	Acciones
Reunión de FEU	presencial	Primer Año	Primer Semestre	Facultad 4	  
Bastión	presencial	Primer Año	Primer Semestre	Facultad 4	  
Asamblea de Ubicación Laboral	presencial	Primer Año	Primer Semestre	Facultad 4	  
Asamblea de Integralidad	presencial	Primer Año	Primer Semestre	Facultad 4	  

Página de 1

Tabla B.5. Historia de usuario # 8

Historia de usuario	
Número: 8	Nombre: Gestionar afectaciones
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla B.5. Continuación de la página anterior

Descripción: El planificador podrá listar, adicionar, modificar y eliminar datos sobre afectaciones que impacten la programación académica.

Escenario 1: Insertar afectación El planificador podrá crear nuevos registros de afectaciones en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente:

- Día
- Turno

Escenario 2: Listar afectación El planificador tendrán la capacidad de listar información detallada sobre cualquier afectación registrado en el sistema. Esta funcionalidad permitirá:

- Acceso rápido a los datos básicos de la afectación (día y turno).
- Listado de la estructura organizativa asignada a la afectación.

Escenario 3: Modificar Afectación El planificador podrá actualizar la información de las afectaciones existentes en el sistema. Este proceso implicará:

- Edición de campos como (día y turno)
- Corregir errores tipográficos o inconsistencias en los datos.

Escenario 4: Eliminar afectación El planificador dispondrá de la opción para eliminar completamente un registro de afectación del sistema. Este proceso implicará:

- Confirmación de la acción antes de proceder con la eliminación.
- Eliminación completa del registro de la afectación en todas las tablas relacionadas.
- Actualización automática de referencias cruzadas para mantener la integridad de los datos.

Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de afectaciones.

Interfaz:

Profesor afectado	Año académico	Semana	Día	Turno	Acciones
Rey Caturia Pon	Primer Año	1	2	Tercera	[Iconos de acciones]
Daniela Cobas Mora	Primer Año	1	14	Primera	[Iconos de acciones]
Salet González López	Primer Año	1	20	Segunda	[Iconos de acciones]
Juan Hurt Cebano	Primer Año	1	30	Primera	[Iconos de acciones]

Tabla B.6. Historia de usuario # 9

Historia de usuario

Continúa en la próxima página

Tabla B.6. Continuación de la página anterior

Número: 9	Nombre: Gestionar carga docente
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador podrá gestionar la carga docente a través del listado, la adición, la modificación y la eliminación de datos relevantes sobre los docentes. Escenario 1: Insertar carga docente El planificador podrá crear nuevos registros de carga docente en el sistema. Este proceso implicará la inserción de datos en los campos preestablecidos, específicamente: • Profesor • Área • Responsabilidad • Carrera • Tipo de curso • GrupoConf • GruposCP Escenario 2: Listar carga docente El planificador tendrá la capacidad de consultar información detallada sobre cualquier carga docente registrado en el sistema. Esta funcionalidad permitirá: • Acceso rápido a los datos básicos de la carga docente (profesor, área, responsabilidad, carrera, tipo de curso, grupos a los que imparte Conferencia, grupos a los que imparte Clase práctica y asignaturas impartidas). • Visualización de la estructura organizativa asignada a la carga docente. Escenario 3: Modificar carga docente El planificador podrá actualizar la información de la carga docente existente en el sistema. Este proceso implicará: • Edición de campos como (profesor, área, responsabilidad, carrera, tipo de curso, grupos a los que imparte Conferencia, grupos a los que imparte Clase práctica y asignaturas impartidas). • Corregir errores tipográficos o inconsistencias en los datos.	
Observaciones: Solo los usuarios autenticados con el rol de planificador podrán realizar acciones de gestión de carga docente.	

Continúa en la próxima página

Tabla B.6. Continuación de la página anterior

Interfaz:

The screenshot shows a web interface titled "Carga Docente". At the top, there is a button labeled "Registrar carga docente". Below this, there is a search bar with a dropdown menu labeled "Mostrar" and a button labeled "Buscar". The main content is a table with the following columns: "Profesor", "Asignatura", "Grupo", and "Acciones". The table contains four rows of data:

Profesor	Asignatura	Grupo	Acciones
Rey Caturia Pon	Primer Año	IDF4101	[Iconos de acciones]
Daniela Cobas Mora	Primer Año	IDF4102	[Iconos de acciones]
Salet González López	Primer Año	IDF4103	[Iconos de acciones]
Juan Hurt Cebano	Primer Año	IDF4104	[Iconos de acciones]

At the bottom right of the interface, there is a pagination control showing "Página 1 de 1".

Tabla B.7. Historia de usuario # 10

Historia de usuario	
Número: 10	Nombre: Exportar carga docente
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 0.4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador podrá exportar datos relacionados con la carga docente. Esto incluye información sobre el profesor como nombre y apellidos, categoría docente, área, responsabilidad, formación académica, carrera, tipo de curso, año académico a los que imparte clase, facultad, grupos a los que imparte Conferencia, grupos a los que imparte Clase práctica y asignaturas impartidas. La exportación facilitará el análisis y la presentación de datos en diferentes formatos para su uso administrativo o académico.	
Observaciones: Solo los usuarios autenticados con el rol de planificador podrá realizar la acción de gestión exportar carga docente.	

Continúa en la próxima página

Tabla B.7. Continuación de la página anterior

Interfaz:

Carga Docente

Registrar carga docente

Mostrar

Entradas

Buscar

Profesor	Asignatura	Grupo	Acciones
Rey Caturia Pon	Primer Año	IDF4101	
Daniela Cobas Mora	Primer Año	IDF4102	
Salet González López	Primer Año	IDF4103	
Juan Hurt Cebano	Primer Año	IDF4104	

Exportar

Página

1

de 1

Tabla B.8. Historia de usuario # 11

Historia de usuario	
Número: 11	Nombre: Generar horario
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 4	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador podrá generar el horario docente para su año. Esta capacidad es fundamental para organizar las clases y asegurar que se cumplan las necesidades educativas de los estudiantes.	
Observaciones: Solo los usuarios autenticados con rol de planificador podrán generar el horario docente.	

Continúa en la próxima página

Tabla B.8. Continuación de la página anterior

Interfaz:

Tabla B.9. Historia de usuario # 12

Historia de usuario	
Número: 12	Nombre: Exportar horario
Usuario: Planificador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El planificador podrá exportar el horario docente generado en formatos PDF o XML que incluirán información sobre grupos, profesores, asignaturas y turnos. Esta funcionalidad permitirá compartir fácilmente el horario con otros interesados o utilizarlo en diferentes plataformas.	
Observaciones: Solo los usuarios autenticados con el rol de planificador podrán exportar el horario docente.	

Continúa en la próxima página

Tabla B.9. Continuación de la página anterior

Interfaz:

Visualizar Horario

Horario del grupo ID4102

	L	M	M	J	V
1	IP I (C)	MD I (C)	IP I (CP)	A (C)	ICI (C)
2	A (CP)	SN (C)	F (C)	SN (CP)	F (CP)
3	-	R (FEU)	A (CP)	F (CP)	SN (C)
4	I	-	I	-	-
5	I	EF I	I	EF I	-
6	-	-	-	-	-

Año académico
 Ingrese el año académico

Profesor
 Ingrese el nombre del profesor

Semana
 Ingrese el número de semana

Grupo
 Ingrese el grupo

Tabla B.10. Historia de usuario # 13

Historia de usuario	
Número: 13	Nombre: Visualizar horario
Usuario: Planificador, Estudiantes, Profesores y Decano	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Katherine Borrero Meriño	
Descripción: El decano, secretario docente, planificador, profesor y estudiantes podrán visualizar el horario docente una vez generado. Esto incluye horarios filtrados por grupos y año. La visualización accesible del horario es importante para todos los involucrados en la planificación académica.	
Observaciones: Solo los usuarios autenticados podrán visualizar el horario docente.	

Continúa en la próxima página

Tabla B.10. Continuación de la página anterior

Interfaz:

Visualizar Horario

Horario del grupo ID4102

	L	M	M	J	V
1	IP I (C)	MD I (C)	IP I (CP)	A (C)	ICI (C)
2	A (CP)	SN (C)	F (C)	SN (CP)	F (CP)
3	-	R (FEU)	A (CP)	F (CP)	SN (C)
4	I	-	I	-	-
5	I	EF I	I	EF I	-
6	-	-	-	-	-

Año académico

 Ingrese el año académico

Profesor

 Ingrese el nombre del profesor

Semana

 Ingrese el número de semana

Grupo

 Ingrese el grupo

Tarjetas CRC

Tabla C.1. Tarjeta CRC # 4

Tarjeta CRC	
Clase: cargaAsignaturas	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Asignatura • CargaDocente

Tabla C.2. Tarjeta CRC # 5

Tarjeta CRC	
Clase: cargaannoAcademico	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Asignatura • CargaDocente

Tabla C.3. Tarjeta CRC # 6

Tarjeta CRC	
Clase: cargatipocurso	
Responsabilidad	Colaboración

Continúa en la próxima página

Tabla C.3. Continuación de la página anterior

• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• CargaDocente
---	--

Tabla C.4. Tarjeta CRC # 7

Tarjeta CRC	
Clase: cargagrupos	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• CargaDocente

Tabla C.5. Tarjeta CRC # 8

Tarjeta CRC	
Clase: Grupo	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	

Tabla C.6. Tarjeta CRC # 9

Tarjeta CRC	
Clase: Local	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	

Tabla C.7. Tarjeta CRC # 10

Tarjeta CRC	
Clase: AfectacionProfesor	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Turno • Pofesor

Tabla C.8. Tarjeta CRC # 11

Tarjeta CRC	
Clase: Actividad	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Asignatura

Tabla C.9. Tarjeta CRC # 12

Tarjeta CRC	
Clase: CargaDocente	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Profesor

Tabla C.10. Tarjeta CRC # 13

Tarjeta CRC	
Clase: AfectacionTurno	
Responsabilidad	Colaboración

Continúa en la próxima página

Tabla C.10. Continuación de la página anterior

• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	
---	--

Tabla C.11. Tarjeta CRC # 14

Tarjeta CRC	
Clase: IndisponibilidadLocal	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Local

Tabla C.12. Tarjeta CRC # 15

Tarjeta CRC	
Clase: MarcoHorario	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	

Tabla C.13. Tarjeta CRC # 16

Tarjeta CRC	
Clase: Turno	
Responsabilidad	Colaboración
• <i>get</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Asignatura

Tabla C.14. Tarjeta CRC # 17

Tarjeta CRC	
Clase: BalanceCarga	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Asignatura

Tabla C.15. Tarjeta CRC # 18

Tarjeta CRC	
Clase: Estudiantes	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Usuario

Tabla C.16. Tarjeta CRC # 19

Tarjeta CRC	
Clase: Planificador	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Usuario

Tabla C.17. Tarjeta CRC # 20

Tarjeta CRC	
Clase: PlanificadorAnio	
Responsabilidad	Colaboración

Continúa en la próxima página

Tabla C.17. Continuación de la página anterior

• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Usuario
---	---

Tabla C.18. Tarjeta CRC # 21

Tarjeta CRC	
Clase: Decano	
Responsabilidad	Colaboración
• <i>get()</i> • <i>post()</i> • <i>filter()</i> • <i>delete()</i>	• Usuario

Tareas de Ingeniería

Tabla D.1. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	Número de Historia de usuario: 1
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 29 de junio de 2024	Fecha de fin: 7 de julio de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita al usuario autenticarse en el sistema	

Tabla D.2. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	Número de Historia de usuario: 2
Nombre de la tarea: Registrar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha de inicio: 10 de julio de 2024	Fecha de fin: 23 de julio de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar un usuario en el sistema	

Tabla D.3. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	Número de Historia de usuario: 2
Nombre de la tarea: Listar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 2

Continúa en la próxima página

Tabla D.3. Continuación de la página anterior

Fecha de inicio: 24 de julio de 2024	Fecha de fin: 30 de julio de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar a los usuarios una vez registrados en el sistema	

Tabla D.4. Tarea de ingeniería # 14

Tarea	
Número de tarea: 14	Número de Historia de usuario: 2
Nombre de la tarea: Modificar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha de inicio: 1 de agosto de 2024	Fecha de fin: 9 de agosto de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar a los usuarios una vez registrados en el sistema	

Tabla D.5. Tarea de ingeniería # 15

Tarea	
Número de tarea: 15	Número de Historia de usuario: 2
Nombre de la tarea: Eliminar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha de inicio: 10 de agosto de 2024	Fecha de fin: 14 de agosto de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar a los usuarios una vez registrados en el sistema	

Tabla D.6. Tarea de ingeniería # 16

Tarea	
Número de tarea: 16	Número de Historia de usuario: 4
Nombre de la tarea: Registrar balance de carga	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 30 de septiembre de 2024	Fecha de fin: 1 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar el balance de carga en el sistema	

Tabla D.7. Tarea de ingeniería # 17

Tarea	
Número de tarea: 17	Número de Historia de usuario: 5
Nombre de la tarea: Registrar grupos	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 6 de octubre de 2024	Fecha de fin: 11 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar grupos de estudiantes en el sistema	

Tabla D.8. Tarea de ingeniería # 18

Tarea	
Número de tarea: 18	Número de Historia de usuario: 5
Nombre de la tarea: Listar grupos	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 12 de octubre de 2024	Fecha de fin: 13 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar los grupos de estudiantes una vez registrados en el sistema	

Tabla D.9. Tarea de ingeniería # 19

Tarea	
Número de tarea: 19	Número de Historia de usuario: 5
Nombre de la tarea: Modificar grupos	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 14 de octubre de 2024	Fecha de fin: 15 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar los grupos de estudiantes una vez registrados en el sistema	

Tabla D.10. Tarea de ingeniería # 20

Tarea	
Número de tarea: 20	Número de Historia de usuario: 5
Nombre de la tarea: Eliminar grupos	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 17 de octubre de 2024	Fecha de fin: 19 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla D.10. Continuación de la página anterior

Descripción: Se implementa una funcionalidad que permita eliminar los grupos registrados en el sistema

Tabla D.11. Tarea de ingeniería # 21

Tarea	
Número de tarea: 21	Número de Historia de usuario: 6
Nombre de la tarea: Registrar locales	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 20 de octubre de 2024	Fecha de fin: 21 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar locales en el sistema	

Tabla D.12. Tarea de ingeniería # 22

Tarea	
Número de tarea: 22	Número de Historia de usuario: 6
Nombre de la tarea: Listar locales	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 22 de octubre de 2024	Fecha de fin: 23 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar los locales registrados en el sistema	

Tabla D.13. Tarea de ingeniería # 23

Tarea	
Número de tarea: 23	Número de Historia de usuario: 6
Nombre de la tarea: Modificar locales	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 24 de octubre de 2024	Fecha de fin: 26 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar los locales registrados en el sistema	

Tabla D.14. Tarea de ingeniería # 24

Tarea
Continúa en la próxima página

Tabla D.14. Continuación de la página anterior

Número de tarea: 24	Número de Historia de usuario: 6
Nombre de la tarea: Eliminar locales	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 27 de octubre de 2024	Fecha de fin: 29 de octubre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita eliminar los locales registrados en el sistema	

Tabla D.15. Tarea de ingeniería # 25

Tarea	
Número de tarea: 25	Número de Historia de usuario: 7
Nombre de la tarea: Registrar actividades	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 30 de octubre de 2024	Fecha de fin: 1 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar las actividades en el sistema	

Tabla D.16. Tarea de ingeniería # 26

Tarea	
Número de tarea: 26	Número de Historia de usuario: 7
Nombre de la tarea: Listar actividades	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 2 de noviembre de 2024	Fecha de fin: 3 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar las actividades registradas en el sistema	

Tabla D.17. Tarea de ingeniería # 27

Tarea	
Número de tarea: 27	Número de Historia de usuario: 7
Nombre de la tarea: Modificar actividades	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 4 de noviembre de 2024	Fecha de fin: 5 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla D.17. Continuación de la página anterior

Descripción: Se implementa una funcionalidad que permita modificar las actividades registradas en el sistema

Tabla D.18. Tarea de ingeniería # 28

Tarea	
Número de tarea: 28	Número de Historia de usuario: 7
Nombre de la tarea: Eliminar actividades	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 6 de noviembre de 2024	Fecha de fin: 8 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita eliminar los actividades registradas en el sistema	

Tabla D.19. Tarea de ingeniería # 29

Tarea	
Número de tarea: 29	Número de Historia de usuario: 8
Nombre de la tarea: Registrar afectaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 9 de noviembre de 2024	Fecha de fin: 10 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar las afectaciones en el sistema	

Tabla D.20. Tarea de ingeniería # 30

Tarea	
Número de tarea: 30	Número de Historia de usuario: 8
Nombre de la tarea: Modificar afectaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 11 de noviembre de 2024	Fecha de fin: 10 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar las afectaciones registradas en el sistema	

Tabla D.21. Tarea de ingeniería # 31

Tarea	
Número de tarea: 31	Número de Historia de usuario: 9
Nombre de la tarea: Registrar carga docente	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 11 de noviembre de 2024	Fecha de fin: 12 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita registrar la carga docente asociada a un profesor en el sistema	

Tabla D.22. Tarea de ingeniería # 32

Tarea	
Número de tarea: 32	Número de Historia de usuario: 9
Nombre de la tarea: Listar carga docente	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 13 de noviembre de 2024	Fecha de fin: 14 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita listar la carga docente en el sistema	

Tabla D.23. Tarea de ingeniería # 33

Tarea	
Número de tarea: 33	Número de Historia de usuario: 9
Nombre de la tarea: Modificar carga docente	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 15 de noviembre de 2024	Fecha de fin: 16 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita modificar la carga docente en el sistema	

Tabla D.24. Tarea de ingeniería # 34

Tarea	
Número de tarea: 34	Número de Historia de usuario: 9
Nombre de la tarea: Eliminar carga docente	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 17 de noviembre de 2024	Fecha de fin: 18 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	

Continúa en la próxima página

Tabla D.24. Continuación de la página anterior

Descripción: Se implementa una funcionalidad que permita eliminar la carga docente en el sistema

Tabla D.25. Tarea de ingeniería # 35

Tarea	
Número de tarea: 35	Número de Historia de usuario: 10
Nombre de la tarea: Exportar carga docente	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 20 de noviembre de 2024	Fecha de fin: 24 de noviembre de 2024
Programador responsable: Katherine Borrero Meriño	
Descripción: Se implementa una funcionalidad que permita exportar la carga docente en formato xlsx en el sistema	

Casos de pruebas de aceptación

Tabla E.1. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario: 1
Nombre: Autenticar usuario	
Descripción: Confirma que los usuarios puedan autenticarse en el sistema ingresando sus credenciales como nombre de usuario y su contraseña.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Pasos de ejecución: Para autenticarse: 1. El usuario debe acceder a la página de inicio sesión del sistema 2. El usuario debe introducir las credenciales: nombre de usuario y contraseña. 3. El usuario debe presionar el botón 'Iniciar Sesión'. 4. El sistema realiza las validaciones de las credenciales y autentica al usuario en caso de ser correctas.	
Resultados esperados: • El usuario se autentifica correctamente en el sistema accediendo a su página principal según el rol que posea. • El sistema muestra un modal especificando un mensaje de error que las credenciales son incorrectas.	

Tabla E.2. Prueba de aceptación # 11

Caso de prueba de aceptación

Continúa en la próxima página

Tabla E.2. Continuación de la página anterior

Código: HU2_P2	Historia de usuario: 2
Nombre: Registrar usuario	
Descripción: Confirma que los usuarios con rol de administrador puedan registrar usuarios en el sistema en correspondencia con su rol	
Condiciones de ejecución: El administrador debe estar autenticado en el sistema.	
Pasos de ejecución: Para registrar usuario: 1. El administrador debe acceder a sección de 'Usuarios'. 2. El administrador debe presionar el ícono de registrar usuario. 3. El sistema muestra un formulario en correspondencia con el rol de usuario con los campos fijos a introducir como (nombre, apellidos, nombre de usuario, contraseña) en caso de ser estudiante se le agrega el año académico, si es planificador se le agrega el año y la facultad, si es decano solamente la facultad y si es profesor se introducen además, categoría docente, formación académica y residente. 4. El administrador rellena todos los campos obligatorios con información válida. 5. El administrador presiona el botón 'Registrar'.	
Resultados esperados: • Si los datos introducidos son válidos, se registra el nuevo usuario en el sistema y se incluye en la lista de usuarios. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.	

Tabla E.3. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU2_P2	Historia de usuario: 2
Nombre: Listar usuario	
Descripción: Confirma que los usuarios con rol de administrador puedan listar usuarios en el sistema en correspondencia con su rol	
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema. • Debe existir al menos un usuario registrado.	

Continúa en la próxima página

Tabla E.3. Continuación de la página anterior

Pasos de ejecución: Para listar usuario: 1. El administrador debe acceder a sección de 'Usuarios'. 2. El sistema muestra una tabla en correspondencia con el rol de usuario donde se recoge la siguiente información (nombre, apellidos y nombre de usuario) en caso de ser estudiante se muestra el año académico, si es planificador se muestra el año y la facultad, si es decano solamente la facultad y si es profesor se muestra además, categoría docente, formación académica y residente.
Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados en dependencia del rol se muestra un mensaje indicando que no hay usuarios registrados.

Tabla E.4. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU2_P2	Historia de usuario: 2
Nombre: Modificar usuario	
Descripción: Confirma que los usuarios con rol de administrador puedan modificar usuarios en el sistema en correspondencia con su rol	
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema. • Debe existir al menos un usuario registrado en el sistema.	
Pasos de ejecución: Para modificar usuario: 1. El administrador debe acceder a sección de 'Usuarios'. 2. El administrador debe presionar el ícono de modificar usuario en la lista de usuarios. 3. El sistema muestra un formulario con la información actual del usuario. 4. El administrador modifica el campo deseado en correspondencia con el rol, los siguientes datos son comunes para todos usuarios (nombre, apellidos, nombre de usuario y contraseña), si es estudiante se puede modificar el año académico, si es planificador se puede modificar el dato año académico y facultad, si es decano incluye el dato facultad y si es profesor están los datos formación académica, categoría docente y residente. 5. El administrador presiona el botón Actualizar.	

Continúa en la próxima página

Tabla E.4. Continuación de la página anterior

Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados en dependencia del rol se muestra un mensaje indicando que no hay usuarios registrados.

Tabla E.5. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU2_P2	Historia de usuario: 2
Nombre: Eliminar usuario	
Descripción: Confirma que los usuarios con rol de administrador puedan eliminar usuarios en el sistema	
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema. • Debe existir al menos un usuario registrado en el sistema.	
Pasos de ejecución: Para eliminar usuario: 1. El administrador debe acceder a sección de 'Usuarios'. 2. El administrador debe presionar el ícono de eliminar usuario en la lista de usuarios. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar al 'usuario'? 4. El administrador confirma la acción realizada presionando el botón Aceptar.	
Resultados esperados: • Los datos registrados del usuario seleccionado son removidos de la lista, si el administrador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el administrador cancela la eliminación.	

Tabla E.6. Prueba de aceptación # 15

Caso de prueba de aceptación	
Código: HU4_P4	Historia de usuario: 4
Nombre: Registrar balance de carga	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar el balance de carga correspondiente a las asignaturas del año académico en el sistema	

Continúa en la próxima página

Tabla E.6. Continuación de la página anterior

Condiciones de ejecución:
El planificador debe estar autenticado en el sistema.
Pasos de ejecución:
Para registrar balance de carga: - El planificador debe acceder a sección de 'Balance de carga'. - El planificador debe presionar el ícono de registrar balance de carga correspondiente a la asignatura seleccionada en la lista de asignaturas. - El sistema muestra un formulario con los datos a introducir 'Opción' que puede ser C, CP, S, T, TE, L, PP, PF, TC. - El planificador presiona el botón Registrar.
Resultados esperados:
Si los datos introducidos son válidos, se registra el nuevo balance de carga en el sistema.

Tabla E.7. Prueba de aceptación # 16

Caso de prueba de aceptación	
Código: HU5_P5	Historia de usuario: 5
Nombre: Registrar grupo	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar grupos en el sistema	
Condiciones de ejecución:	
El planificador debe estar autenticado en el sistema.	
Pasos de ejecución:	
Para registrar grupo: - El planificador debe acceder a sección de 'Grupos'. - El planificador debe presionar el ícono de registrar grupo. - El sistema muestra un formulario para introducir los datos requeridos (nombre de grupo, cantidad de estudiantes, año docente, seleccione el tipo de curso). - El planificador rellena todos los campos obligatorios con información válida. - El planificador presiona el botón 'Registrar'.	

Continúa en la próxima página

Tabla E.7. Continuación de la página anterior

Resultados esperados: • Si los datos introducidos son válidos, se registra el nuevo grupo en el sistema y se incluye en la lista de grupos. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.
--

Tabla E.8. Prueba de aceptación # 17

Caso de prueba de aceptación	
Código: HU5_P5	Historia de usuario: 5
Nombre: Listar grupos	
Descripción: Confirma que los usuarios con rol de planificador puedan listar grupos en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un grupo registrado.	
Pasos de ejecución: Para listar grupos: 1. El planificador debe acceder a sección de 'Grupos'. 2. El sistema muestra una tabla de los grupos donde se recoge la siguiente información (nombre de grupo, cantidad de estudiantes, año docente, seleccione el tipo de curso).	
Resultados esperados: • Los datos de los grupos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay grupos registrados.	

Tabla E.9. Prueba de aceptación # 18

Caso de prueba de aceptación	
Código: HU5_P5	Historia de usuario: 5
Nombre: Modificar grupo	
Descripción: Confirma que los usuarios con rol de planificador puedan modificar grupos en el sistema	

Continúa en la próxima página

Tabla E.9. Continuación de la página anterior

Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un grupo registrado en el sistema.
Pasos de ejecución: Para modificar grupo: 1. El planificador debe acceder a sección de 'Grupos'. 2. El planificador debe presionar el ícono de modificar grupo en la lista de grupos. 3. El sistema muestra un formulario con la información actual del grupo. 4. El planificador modifica el campo deseado, los datos son (nombre de grupo, cantidad de estudiantes, año docente, seleccione el tipo de curso). 5. El planificador presiona el botón Actualizar.
Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay grupos registrados.

Tabla E.10. Prueba de aceptación # 19

Caso de prueba de aceptación	
Código: HU5_P5	Historia de usuario: 5
Nombre: Eliminar grupo	
Descripción: Confirma que los usuarios con rol de planificador puedan eliminar grupos en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un grupo registrado en el sistema.	
Pasos de ejecución: Para eliminar grupo: 1. El planificador debe acceder a sección de 'Grupos'. 2. El planificador debe presionar el ícono de eliminar grupo en la lista de grupos. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar el 'grupo'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.	

Continúa en la próxima página

Tabla E.10. Continuación de la página anterior

Resultados esperados: • Los datos registrados del grupo seleccionado son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.
--

Tabla E.11. Prueba de aceptación # 20

Caso de prueba de aceptación	
Código: HU6_P6	Historia de usuario: 6
Nombre: Registrar local	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar locales en el sistema	
Condiciones de ejecución: El planificador debe estar autenticado en el sistema.	
Pasos de ejecución: Para registrar local: 1. El planificador debe acceder a sección de 'Locales'. 2. El planificador debe presionar el ícono de registrar local. 3. El sistema muestra un formulario para introducir los datos requeridos (nombre de local, cantidad de estudiantes, año docente, seleccione el tipo de curso). 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Registrar'.	
Resultados esperados: • Si los datos introducidos son válidos, se registra el nuevo local en el sistema y se incluye en la lista de locales. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.	

Tabla E.12. Prueba de aceptación # 21

Caso de prueba de aceptación	
Código: HU6_P6	Historia de usuario: 6
Nombre: Listar locales	
Descripción: Confirma que los usuarios con rol de planificador puedan listar locales en el sistema	

Continúa en la próxima página

Tabla E.12. Continuación de la página anterior

Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un local registrado.
Pasos de ejecución: Para listar locales: 1. El planificador debe acceder a sección de 'Locales'. 2. El sistema muestra una tabla de los locales donde se recoge la siguiente información (nombre de local, abreviatura, piso, capacidad, designación).
Resultados esperados: • Los datos de los locales registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay locales registrados.

Tabla E.13. Prueba de aceptación # 22

Caso de prueba de aceptación	
Código: HU6_P6	Historia de usuario: 6
Nombre: Modificar local	
Descripción: Confirma que los usuarios con rol de planificador puedan modificar locales en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un local registrado en el sistema.	
Pasos de ejecución: Para modificar local: 1. El planificador debe acceder a sección de 'Locales'. 2. El planificador debe presionar el ícono de modificar local en la lista de locales. 3. El sistema muestra un formulario con la información actual del local. 4. El planificador modifica el campo deseado, los datos son (nombre de local, abreviatura, piso, capacidad, designación). 5. El planificador presiona el botón Actualizar.	

Continúa en la próxima página

Tabla E.13. Continuación de la página anterior

Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay locales registrados.

Tabla E.14. Prueba de aceptación # 23

Caso de prueba de aceptación	
Código: HU6_P6	Historia de usuario: 6
Nombre: Eliminar local	
Descripción: Confirma que los usuarios con rol de planificador puedan eliminar locales en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos un local registrado en el sistema.	
Pasos de ejecución: Para eliminar local: 1. El planificador debe acceder a sección de 'Locales'. 2. El planificador debe presionar el ícono de eliminar local en la lista de locales. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar el 'local'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.	
Resultados esperados: • Los datos registrados del local seleccionado son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.	

Tabla E.15. Prueba de aceptación # 24

Caso de prueba de aceptación	
Código: HU7_P7	Historia de usuario: 7
Nombre: Registrar actividad	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar actividades en el sistema	

Continúa en la próxima página

Tabla E.15. Continuación de la página anterior

Condiciones de ejecución: El planificador debe estar autenticado en el sistema.
Pasos de ejecución: Para registrar actividad: 1. El planificador debe acceder a la sección de 'Actividades'. 2. El planificador debe presionar el ícono de registrar actividad. 3. El sistema muestra un formulario para introducir los datos requeridos (nombre de actividad, cantidad de estudiantes, año docente, seleccione el tipo de curso). 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Registrar'.
Resultados esperados: • Si los datos introducidos son válidos, se registra la nueva actividad en el sistema y se incluye en la lista de actividades. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.

Tabla E.16. Prueba de aceptación # 25

Caso de prueba de aceptación	
Código: HU7_P7	Historia de usuario: 7
Nombre: Listar actividades	
Descripción: Confirma que los usuarios con rol de planificador puedan listar actividades en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una actividad registrada.	
Pasos de ejecución: Para listar actividades: 1. El planificador debe acceder a la sección de 'Actividades'. 2. El sistema muestra una tabla de las actividades donde se recoge la siguiente información (nombre de actividad, abreviatura, piso, capacidad, designación).	

Continúa en la próxima página

Tabla E.16. Continuación de la página anterior

Resultados esperados: • Los datos de las actividades registradas, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay actividades registradas.
--

Tabla E.17. Prueba de aceptación # 26

Caso de prueba de aceptación	
Código: HU7_P7	Historia de usuario: 7
Nombre: Modificar actividad	
Descripción: Confirma que los usuarios con rol de planificador puedan modificar actividades en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una actividad registrado en el sistema.	
Pasos de ejecución: Para modificar actividad: 1. El planificador debe acceder a la sección de 'Actividades'. 2. El planificador debe presionar el ícono de modificar actividad en la lista de actividades. 3. El sistema muestra un formulario con la información actual de la actividad. 4. El planificador modifica el campo deseado, los datos son (nombre de actividad, abreviatura, piso, capacidad, designación). 5. El planificador presiona el botón Actualizar.	
Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay actividades registradas.	

Tabla E.18. Prueba de aceptación # 27

Caso de prueba de aceptación	
Código: HU7_P7	Historia de usuario: 7
Nombre: Eliminar actividad	

Continúa en la próxima página

Tabla E.18. Continuación de la página anterior

Descripción: Confirma que los usuarios con rol de planificador puedan eliminar actividades en el sistema
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una actividad registrada en el sistema.
Pasos de ejecución: Para eliminar actividad: 1. El planificador debe acceder a la sección de 'Actividades'. 2. El planificador debe presionar el ícono de eliminar actividad en la lista de actividades. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar la 'actividad'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.
Resultados esperados: • Los datos registrados de la actividad seleccionada son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.

Tabla E.19. Prueba de aceptación # 28

Caso de prueba de aceptación	
Código: HU8_P8	Historia de usuario: 8
Nombre: Registrar afectación	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar afectaciones en el sistema	
Condiciones de ejecución: El planificador debe estar autenticado en el sistema.	
Pasos de ejecución: Para registrar afectación: 1. El planificador debe acceder a la sección de 'Afectaciones'. 2. El planificador debe presionar el ícono de registrar afectación. 3. El sistema muestra un formulario para introducir los datos requeridos (nombre de afectación, cantidad de estudiantes, año docente, seleccione el tipo de curso). 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Registrar'.	

Continúa en la próxima página

Tabla E.19. Continuación de la página anterior

Resultados esperados: • Si los datos introducidos son válidos, se registra la nueva afectación en el sistema y se incluye en la lista de afectaciones. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.

Tabla E.20. Prueba de aceptación # 29

Caso de prueba de aceptación	
Código: HU9_P9	Historia de usuario: 9
Nombre: Registrar carga docente	
Descripción: Confirma que los usuarios con rol de planificador puedan registrar la carga docente en el sistema	
Condiciones de ejecución: El planificador debe estar autenticado en el sistema.	
Pasos de ejecución: Para registrar carga docente: 1. El planificador debe acceder a la sección de 'Carga docente'. 2. El planificador debe presionar el ícono de registrar carga docente. 3. El sistema muestra un formulario para introducir los datos requeridos (nombre del profesor, categoría docente, formación académica, residente, carrera, asignaturas, años, grupos, grupos de conferencia, grupos de clase práctica, área y responsabilidad). 4. El planificador rellena todos los campos obligatorios con información válida. 5. El planificador presiona el botón 'Registrar'.	
Resultados esperados: • Si los datos introducidos son válidos, se registra la nueva carga docente en el sistema y se incluye en la lista de la carga docente. • Si alguno de los campos requeridos son inválidos el sistema muestra un modal con un mensaje de error indicando donde está el problema.	

Tabla E.21. Prueba de aceptación # 30

Caso de prueba de aceptación	
Código: HU9_P9	Historia de usuario: 9
Nombre: Listar carga docente	

Continúa en la próxima página

Tabla E.21. Continuación de la página anterior

Descripción: Confirma que los usuarios con rol de planificador puedan listar la carga docente de los profesores en el sistema
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una carga docente registrada asociada a un profesor.
Pasos de ejecución: Para listar carga docente: 1. El planificador debe acceder a la sección de 'Carga docente'. 2. El sistema muestra una tabla de las cargas docentes de los profesores donde se recoge la siguiente información (nombre del profesor, categoría docente, formación académica, residente, carrera, asignaturas, años, grupos, grupos de conferencia, grupos de clase práctica, área y responsabilidad).
Resultados esperados: • Los datos de la carga docente registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay carga docente registrada.

Tabla E.22. Prueba de aceptación # 31

Caso de prueba de aceptación	
Código: HU9_P9	Historia de usuario: 9
Nombre: Modificar carga docente	
Descripción: Confirma que los usuarios con rol de planificador puedan modificar carga docente en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una carga docente asociada a un profesor registrado en el sistema.	

Continúa en la próxima página

Tabla E.22. Continuación de la página anterior

Pasos de ejecución: Para modificar carga docente: 1. El planificador debe acceder a la sección de 'Carga docente'. 2. El planificador debe presionar el ícono de modificar carga docente en la lista de cargas docentes. 3. El sistema muestra un formulario con la información actual de la carga docente. 4. El planificador modifica el campo deseado, los datos son (nombre de carga docente, abreviatura, piso, capacidad, designación). 5. El planificador presiona el botón Actualizar.
Resultados esperados: • Los datos registrados, se muestran en una lista con la información correspondiente. • Si no existen datos registrados se muestra un mensaje indicando que no hay carga docente registrada.

Tabla E.23. Prueba de aceptación # 32

Caso de prueba de aceptación	
Código: HU9_P9	Historia de usuario: 9
Nombre: Eliminar carga docente	
Descripción: Confirma que los usuarios con rol de planificador puedan eliminar carga docente en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una carga docente registrada en el sistema.	
Pasos de ejecución: Para eliminar carga docente: 1. El planificador debe acceder a la sección de 'Carga docente'. 2. El planificador debe presionar el ícono de eliminar carga docente en la lista de cargas docentes. 3. El sistema muestra un modal solicitando la confirmación de la acción realizada con el mensaje: ¿Desea eliminar la 'carga docente'? 4. El planificador confirma la acción realizada presionando el botón Aceptar.	

Continúa en la próxima página

Tabla E.23. Continuación de la página anterior

Resultados esperados: • Los datos registrados de la carga docente seleccionada son removidos de la lista, si el planificador confirma la eliminación. • Los datos se mantienen en su estado anterior, si el planificador cancela la eliminación.
--

Tabla E.24. Prueba de aceptación # 33

Caso de prueba de aceptación	
Código: HU9_P9	Historia de usuario: 9
Nombre: Exportar carga docente	
Descripción: Confirma que los usuarios con rol de planificador puedan exportar carga docente en el sistema	
Condiciones de ejecución: • El planificador debe estar autenticado en el sistema. • Debe existir al menos una carga docente registrada en el sistema.	
Pasos de ejecución: Para exportar carga docente: 1. El planificador debe acceder a la sección de 'Carga docente'. 2. El planificador debe presionar el ícono de exportar carga docente. 3. El sistema realiza la exportación de los datos en un archivo formato xlsx.	
Resultados esperados: • Los datos registrados de la carga docente son exportados.	

Tabla E.25. Prueba de aceptación # 34

Caso de prueba de aceptación	
Código: HU12_P12	Historia de usuario: 12
Nombre: Visualizar horario	
Descripción: Confirma que los usuarios con roles de planificador pueden buscar el horario en el sistema	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Los datos deben estar previamente introducidos.	

Continúa en la próxima página

Tabla E.25. Continuación de la página anterior

Pasos de ejecución:

Para buscar el horario

1. El usuario debe iniciar sesión en el sistema en correspondencia con el rol de usuario.
2. El usuario debe acceder a la sección 'Horario'.
3. El usuario debe introducir los criterios de búsqueda (año académico, una semana o un grupo).
4. El usuario debe hacer clic en el botón Buscar.

Resultados esperados: Se debe buscar el horario docente generado correspondiente a los datos introducidos.