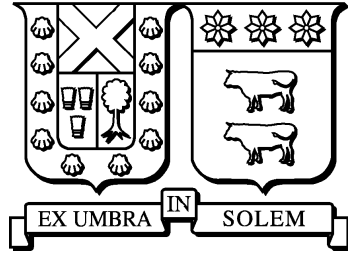


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



TRANSFORMACIÓN DIGITAL DE LOS
PROCESOS RELACIONADOS A LA GESTIÓN Y
DESARROLLO DE PROYECTOS
INFORMÁTICOS: CASO FERIA DE SOFTWARE

GABRIEL IGNACIO VALENZUELA LORCA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

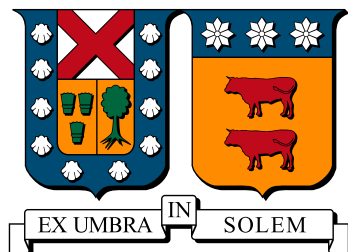
PROFESOR GUÍA: PEDRO FRANCISCO GODOY BARRERA

2023

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



**TRANSFORMACIÓN DIGITAL DE LOS
PROCESOS RELACIONADOS A LA GESTIÓN Y
DESARROLLO DE PROYECTOS
INFORMÁTICOS: CASO FERIA DE SOFTWARE**

GABRIEL IGNACIO VALENZUELA LORCA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: PEDRO FRANCISCO GODOY BARRERA

PROFESOR CORREFERENTE: CECILIA REYES COVARRUBIAS

2023

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O LA INSTITUCIÓN

Agradecimientos

Me gustaría agradecer a todas las personas que creyeron en mí durante este largo proceso. Fueron largas noches de trabajo y meses extenuantes, pero jamás lo habría logrado si no fuera por el apoyo constante de todas esas importantes personas. Mi familia, madre, hermana y hermano, que siempre creyeron en mí y mis capacidades incluso cuando parecía que ya no podría dar más; los amo inmensamente.

Agradezco a mi pareja, quien estuvo apoyándome en cada momento de trabajo, ya sea llevándome algo para comer o solo con palabras de aliento. Este trabajo es de ambos, puedo compartir el mérito contigo, pues te lo mereces totalmente.

A mis amigos, quienes me daban el apoyo de la forma más violenta posible, pero que era absolutamente necesaria. Uno a veces necesita remezones para seguir funcionando. Les dedico mi más grande agradecimiento, y espero poder recompensarlos algún día.

A mis profesores que ayudaron en esta memoria, que a pesar de los apuros y el trabajo a deshora, hicieron todo lo posible para permitir que este proceso llegase a su conclusión. De verdad que aprecio mucho todo lo que han hecho por mí durante todo este tiempo, y espero retribuirles llegado el momento.

Por último, agradezco a todas las personas que se cruzaron por mi camino en algún momento, quienes me ayudaron en alguna instancia o que compartieron conmigo en ciertas etapas de mi vida. Creo que toda persona deja un pequeño grano de arena en la mente y el corazón, y sé que hay muchos que estuvieron ahí conmigo. A ustedes, solo puedo decir gracias.

Resumen

En la Universidad Técnica Federico Santa María, en el Departamento de Informática, existe un evento llamado Feria de Software que se realiza todos los años, intercalando el emplazamiento en donde se lleva a cabo entre Campus San Joaquín en Santiago y Campus Casa Central en Valparaíso. En este evento cientos de estudiantes, cursando desde el quinto año de Ingeniería Civil Informática, exponen innovadores proyectos de *software* que se fabrican durante el plazo de un año. El desarrollo de estos proyectos es de alta complejidad, y la gestión de estos por parte de los docentes se dificulta por la cantidad de equipos a considerar y por la ineficacia de ciertas herramientas, tecnologías y procesos utilizados para realizar tal gestión de forma óptima. Un mayor control sobre los proyectos y una gestión eficiente podrían mejorar el desempeño de los alumnos, la calidad de los proyectos y ayudar a prevenir problemas durante el desarrollo, lo cual permitiría mitigar riesgos de fracaso en la entrega del producto fabricado.

Considerando lo anterior, en este trabajo se propone e implementa una herramienta de gestión de proyectos diseñada a partir de las necesidades de las asignaturas asociadas a la Feria de Software, haciendo uso de tecnologías de vanguardia, y que busque centralizar los recursos de las asignaturas, optimizar los procesos de evaluación, desarrollo y entrega; dar visibilidad a los proyectos en curso y entregar herramientas para la gestión de estos últimos.

Abstract

At Universidad Técnica Federico Santa María, at the IT department, exists an event called Feria de Software what is done every year switching the location where it takes place between San Joaquín campus at Santiago and Casa Central at Valparaíso. In this event hundred of students, cursing from the fifth year at Ingeniería Civil Informática, present innovative software projects developed over the whole year. The development of these projects is really complex, and their management related to the teachers is getting more difficult due to the quantity of teams considered and the inefficacy of some tools, technologies and processes used to make that management optimal. A better control over the projects and a efficient management would improve students performance, projects quality and help preventing problems over the development, allowing the mitigation of failure risks in the developed product delivery.

With that in consideration, this paper proposes and implements a project management tool designed by the necessities of the courses related to Feria de Software using modern technologies, looking for the centralization of the courses resources, improving the evaluation, development and delivering processes; giving visibility to in progress projects and delivering tools for their management.

Tabla de Contenidos

Agradecimientos	III
Resumen	IV
Abstract	V
Tabla de Contenidos	VI
Lista de Tablas	XI
Lista de Figuras	XIII
Glosario	XVII
Introducción	1
1. Definición del Problema	3
1.1. Contexto	3
1.2. Problema	4
1.3. Objetivos	6

1.3.1. Objetivo General	6
1.3.2. Objetivos Específicos	6
2. Marco Conceptual	7
2.1. Transformación Digital	7
2.2. Metodología de trabajo <i>SCRUM</i>	8
2.3. Herramientas de Gestión de Proyectos	10
2.4. Herramientas de Versionamiento	15
2.5. Automatización del Proceso de Producción de Software	18
2.5.1. Control de Versiones	18
2.5.2. CI: <i>Continuous Integration</i>	18
2.5.3. CD: <i>Continuous Delivery</i>	19
2.6. React	20
2.7. Flask	21
2.8. Bases de datos NoSQL	22
2.8.1. MongoDB	23
2.9. Amazon Simple Storage Service: S3	23
2.10. Amazon Amplify	24
2.11. Amazon EC2	24
2.12. Amazon Route 53	24
2.13. Principios de Nielsen	25
3. Propuesta de Solución	28

3.1. Alternativas de Soluciones	28
3.1.1. Reorganización e integración de elementos actuales	29
3.1.2. Uso de una herramienta de gestión especializada: <i>Monday</i>	30
3.1.3. Creación de una herramienta a medida	31
3.2. Opción seleccionada	31
3.3. Requerimientos	32
3.4. Herramientas de Desarrollo	38
3.5. Modelos de Datos	40
3.5.1. Modelo conceptual de datos	40
3.5.2. Modelo relacional de datos	44
3.6. Arquitectura de la Plataforma	46
3.6.1. <i>React APP</i>	47
3.6.2. <i>Flask API Rest</i>	50
3.7. Metodología de Trabajo	54
3.8. Vistas de la Aplicación	55
3.8.1. Login	55
3.8.2. Listado de actividades	55
3.8.3. Entregables	57
3.8.4. Rúbricas	57
3.8.5. Evaluación	58
3.8.6. Tablero de historias de usuario	58
3.8.7. Dashboard de equipos	60

3.8.8.	Evaluación de <i>sprints</i>	60
3.8.9.	Equipo	61
4.	Validación de la Solución	63
4.1.	Cumplimiento de los principios de Nielsen	63
4.1.1.	Visibilidad del estado del sistema	63
4.1.2.	Concordancia entre el mundo real y el sistema	64
4.1.3.	Controles de usuario y libertad	65
4.1.4.	Consistencia y estándares	66
4.1.5.	Prevención de errores	68
4.1.6.	Reconocer es mejor que recordar	69
4.1.7.	Flexibilidad y eficiencia de uso	69
4.1.8.	Estética y diseño minimalista	70
4.1.9.	Ayudar a los usuarios a reconocer, diagnosticar y solucionar errores	71
4.1.10.	Ayuda y documentación	71
4.2.	Pruebas de Profesores	72
4.3.	Pruebas de Alumnos	73
4.3.1.	Recorrido en la aplicación	73
4.3.2.	Encuesta	74
4.3.3.	Resultados de la prueba	75
5.	Conclusiones	78
5.1.	Conclusiones generales	78

5.2. Cumplimiento de objetivos	80
5.3. Trabajo futuro	81
Referencias	83
Índice de Términos	87
A. Diagramas	87

Lista de Tablas

3.1. HU 1: <i>dashboard</i> general de proyectos.	32
3.2. HU 2: creación de actividades.	33
3.3. HU 3: registro e ingreso de usuarios.	34
3.4. HU 4: realización de entregables.	34
3.5. HU 5: creación de rúbricas.	35
3.6. HU 6: evaluación de entregables con rúbrica.	35
3.7. HU 7: listado de usuarios.	36
3.8. HU 8: creación de equipos.	36
3.9. HU 9: creación y edición de historias de usuario.	37
3.10. HU 10: tablón de historias de usuario.	37
3.11. HU 11: evaluación de <i>sprints</i>	38
3.12. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Actividad”. Fuente: elaboración propia.	51
3.13. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Entregable”. Fuente: elaboración propia.	51
3.14. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Periodo”. Fuente: elaboración propia.	52

3.15. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Rubrica”.	
Fuente: elaboración propia.	52
3.16. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Historia de Usuario”.	
Fuente: elaboración propia.	52
3.17. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Equipo”.	
Fuente: elaboración propia.	53
3.18. Listado de rutas ofrecidas por la <i>API Rest</i> para la entidad “Usuario”.	
Fuente: elaboración propia.	53
4.1. Completitud de Historias de Usuario según profesor guía. Fuente: elaboración propia.	72

Lista de Figuras

2.1. Diagrama del proceso de CI/CD de un software en desarrollo [1][2] . . .	20
3.1. Modelo conceptual de datos de la aplicación a desarrollar. Fuente: elaboración propia.	41
3.2. Modelo relacional de datos de la aplicación a desarrollar, adaptado para su compatibilidad con <i>MongoDB</i> . Fuente: elaboración propia.	45
3.3. Diagrama de arquitectura por capas de la aplicación. Fuente: elaboración propia.	46
3.4. Pantalla de <i>login</i> de la plataforma. Fuente: elaboración propia.	56
3.5. Pantalla de actividades de la plataforma. Fuente: elaboración propia. . .	56
3.6. Pantalla de entregables de los alumnos en la plataforma. Fuente: elaboración propia.	57
3.7. Pantalla de creación de rúbricas de la plataforma. Fuente: elaboración propia.	58
3.8. Pantalla de evaluación de rúbricas de la plataforma. Fuente: elaboración propia.	59
3.9. Pantalla del tablero de historias de usuario de la plataforma. Fuente: elaboración propia.	59
3.10. Pantalla de progreso de los proyectos o <i>dashboard</i> . Fuente: elaboración propia.	60

3.11. Pantalla de evaluación de <i>sprints</i> . Fuente: elaboración propia.	61
3.12. Pantalla de equipo y proyecto. Fuente: elaboración propia.	62
4.1. Evidencia de barras de carga en la aplicación. Fuente: elaboración propia. 64	
4.2. Evidencia de barras de carga en los botones de la aplicación. Fuente: elaboración propia.	64
4.3. Evidencia de destaque de elementos frente a interacción con ellos. Fuente: elaboración propia.	64
4.4. Evidencia del uso de elementos de la vida cotidiana en la aplicación. Fuente: elaboración propia.	65
4.5. Evidencia de la opción de editar el equipo y proyecto en la aplicación. Fuente: elaboración propia.	65
4.6. Evidencia de la opción de editar una rúbrica en la aplicación. Fuente: elaboración propia.	66
4.7. Evidencia del diseño del formulario de creación de actividades. Fuente: elaboración propia.	67
4.8. Evidencia del diseño del formulario de creación de historias de usuario. Fuente: elaboración propia.	67
4.9. Evidencia de la barra de navegación como elemento consistente con las páginas web. Fuente: elaboración propia.	68
4.10. Evidencia de la validación de un formulario de la aplicación. Fuente: elaboración propia.	68
4.11. Evidencia del orden de información por fecha de término, distinción de información con colores más fuertes y el uso de íconos reconocidos por otras aplicaciones. Fuente: elaboración propia.	69
4.12. Vista del tablero de historias de usuario como evidencia de estética y diseño minimalista. Fuente: elaboración propia.	71

A.1. Diagrama del Flujo de Feria [3]. En esta se marca el flujo de los procesos, los entregables resultantes del proceso y los entregables opcionales (en línea punteada).	87
A.2. Diagrama ampliado del proceso de CI/CD de un software en desarrollo [1][2].	88
A.3. Modelo conceptual de datos de la aplicación a desarrollar, ampliado. Fuente: elaboración propia.	89
A.4. Modelo relacional de datos de la aplicación a desarrollar adaptado a <i>MongoDB</i> , ampliado. Fuente: elaboración propia.	90
A.5. Pantalla de <i>login</i> de la plataforma expandida. Fuente: elaboración propia.	91
A.6. Pantalla de actividades de la plataforma expandida. Fuente: elaboración propia.	92
A.7. Pantalla de entregables de los alumnos en la plataforma expandida. Fuente: elaboración propia.	93
A.8. Pantalla de creación de rúbricas de la plataforma expandida. Fuente: elaboración propia.	94
A.9. Pantalla de evaluación de rúbricas de la plataforma expandida. Fuente: elaboración propia.	95
A.10. Pantalla del tablero de historias de usuario de la plataforma expandida. Fuente: elaboración propia.	96
A.11. Pantalla de progreso de los proyectos o <i>dashboard</i> expandida. Fuente: elaboración propia.	97
A.12. Pantalla de evaluación de <i>sprints</i> expandida. Fuente: elaboración propia.	98
A.13. Pantalla de equipo y proyecto expandida. Fuente: elaboración propia.	99
A.14. Gráfico del año de rendición de la Feria de Software. Fuente: Google Forms	100

A.15. Gráfico de los resultados obtenidos en la pregunta 1 de la Sección 4.3.2.	
Fuente: Google Forms	100
A.16. Gráfico de los resultados obtenidos en la pregunta 2 de la Sección 4.3.2.	
Fuente: Google Forms	101
A.17. Gráfico de los resultados obtenidos en la pregunta 3 de la Sección 4.3.2.	
Fuente: Google Forms	101
A.18. Gráfico de los resultados obtenidos en la pregunta 4 de la Sección 4.3.2.	
Fuente: Google Forms	102
A.19. Gráfico de los resultados obtenidos en la pregunta 5 de la Sección 4.3.2.	
Fuente: Google Forms	102
A.20. Gráfico de los resultados obtenidos en la pregunta 6 de la Sección 4.3.2.	
Fuente: Google Forms	103
A.21. Gráfico de los resultados obtenidos en la pregunta 7 de la Sección 4.3.2.	
Fuente: Google Forms	103
A.22. Gráfico de los resultados obtenidos en la pregunta 8 de la Sección 4.3.2.	
Fuente: Google Forms	104
A.23. Gráfico de los resultados obtenidos en la pregunta 9 de la Sección 4.3.2.	
Fuente: Google Forms	104
A.24. Gráfico de los resultados obtenidos en la pregunta 10 de la Sección 4.3.2.	
Fuente: Google Forms	105
A.25. Gráfico de los resultados obtenidos en la pregunta 11 de la Sección 4.3.2.	
Fuente: Google Forms	105

Glosario

A

API

(Application Programming Interface) pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades. 21, 34

B

Backlog

Dentro de la gestión ágil de proyectos, lista priorizada de funcionalidades que debe contener un producto. A veces se denomina lista de tareas pendientes. 29

D

Dashboard

herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los indicadores clave de desempeño, métricas y datos fundamentales para hacer un seguimiento del estado de un proceso específico. 30–32

F

Framework

estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. 20

J

Javascript

lenguaje de programación que permite implementar funciones complejas en páginas web. 20

P

Python

lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. 21

S

Scrum

Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. 14

Sprint

Período breve de tiempo fijo en el que un equipo de *scrum* trabaja para completar una cantidad de trabajo establecida. 29, 32, 33, 37, 38, 42, 60

SQL

(Structured Query Language) lenguaje específico de dominio, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. 22

T

Token

referencia (un identificador) que regresa a los datos sensibles a través de un sistema de sustitución de elementos de datos sensibles por equivalentes no sensibles. 34, 50

Introducción

A lo largo de la historia el ser humano ha logrado una gran cantidad de avances tecnológicos que le han permitido mejorar su calidad de vida y alcanzar horizontes que en la antigüedad eran inimaginables. Su capacidad de adaptarse a los cambios y la búsqueda constante de mejoras le han permitido conseguir tales descubrimientos, por lo que la evolución del hombre deja de ser algo netamente biológico, sino que también algo social y cultural. Mientras que en la antigüedad el reflejo de tecnología se apreciaba en la construcción de un hermoso castillo, en la actualidad un software de gran calidad y diseño queda grabado en la memoria de las personas que hacen uso de él; siendo que ambos productos son de dimensiones completamente diferentes, ambos comparten más características de las que uno podría imaginar: el uso de herramientas aptas, los materiales necesarios, diseños previos para guiar la construcción, trabajadores capacitados para el trabajo o la correcta gestión de todo lo antes nombrado.

Con el paso del tiempo, nuevas problemáticas han surgido y con ello, nuevas soluciones han tenido que ser estudiadas y desarrolladas. En el desarrollo de software, los proyectos de alta complejidad suelen estar ligados a un gran equipo de trabajo, distintas tecnologías y nuevas metodologías, volviendo necesario que una correcta gestión de todo lo anterior se vuelva indispensable para conseguir el éxito. Dado esto, no es difícil imaginar que la correcta gestión de numerosos proyectos es una tarea abrumadora para quien sea, y que buscar alternativas para facilitar estos procesos se vuelve algo necesario no solo para dar comodidad a los usuarios, sino que también para asegurar que la gestión se realice de manera eficaz y eficiente.

Es por lo anterior que la presente memoria busca crear una herramienta que permita gestionar el desarrollo de numerosos proyectos de una forma más sencilla y eficiente, enfocada tanto en los encargados de la gestión como en los equipos de trabajo. Todo

esto se enmarca en el contexto de las asignaturas ligadas a la Feria de Software, evento desarrollado por el Departamento de Informática de la Universidad Técnica Federico Santa María, donde profesores y ayudantes cumplen el rol de gestionar el correcto progreso de los proyectos de forma conjunta, mientras que los alumnos se encargan de la gestión de su propio trabajo y del desarrollo de este mismo. Esta herramienta será validada con estudiantes de la carrera de Ingeniería Civil Informática y docentes que estén participando activamente en las asignaturas mencionadas previamente .

La estructura del documento consta de 4 capítulos, los cuales son los siguientes:

El primero es “Definición del Problema”, donde se detalla la necesidad que tienen los alumnos y docentes de la Feria de Software por poseer una herramienta que mejore la trazabilidad de los proyectos y la unificación de los procesos ligados a la asignatura, facilitando la gestión tanto para profesores como estudiantes y, por ende, del ramo en sí mismo. Todo lo anterior provocado por las dificultades persistentes al realizar seguimiento del trabajo de los equipos a lo largo del transcurso de la asignatura y su constante evolución.

El segundo capítulo es “Marco Conceptual”, en donde se presenta el estado del arte sobre transformación digital, herramientas de gestión, versionamiento y automatización del proceso de producción de software, y las distintas tecnologías para su implementación.

El tercer capítulo es “Propuesta de Solución”, el cual entrega distintas alternativas detalladas del diseño e implementación de la herramienta de gestión para el ramo de Feria de Software.

En el cuarto capítulo, “Validación de la Solución”, se entregan los resultados obtenidos por los alumnos y docentes de la Feria de Software asociados a su experiencia haciendo uso de la herramienta en el desarrollo de la asignatura.

Finalmente, se presentan las conclusiones obtenidas del trabajo.

Capítulo 1

Definición del Problema

En este capítulo se definirá el problema que dio origen a esta memoria, planteándose además los objetivos de este trabajo y su alcance.

1.1. Contexto

La Feria de Software es un evento realizado por el Departamento de Informática de la Universidad Técnica Federico Santa María en donde estudiantes, tanto del campus San Joaquín (Santiago) como del campus Casa Central (Valparaíso), desarrollan un proyecto informático haciendo uso de la metodología de trabajo *scrum* (el cual se explica en la **Sección 2.2**) en el plazo de dos semestres, para finalmente presentarlo al público en la instancia ya mencionada, lugar al cual acuden tanto sus familias como amigos, conocidos y personas de la industria quienes buscan potenciar estos trabajos o a los estudiantes. En el primer semestre ("Gestión de Proyectos de Informática") los estudiantes se encargan de sentar las bases para el desarrollo de su proyecto, definiendo la idea a trabajar y la documentación necesaria para su correcto avance distribuida en varios entregables informativos como los riesgos, planificación, historias de usuario, entre otros. Además, es en este semestre en donde los alumnos deben construir un "Producto Mínimo Viable" (o PMV) con el fin de mitigar los riesgos relevantes del proyecto, el cual consta de un primer desarrollo que abarca los requerimientos más importantes y que demuestra la viabilidad del proyecto propuesto. El segundo semestre ("Taller Desarrollo de Proyecto

de Informática") se enfoca casi exclusivamente en el desarrollo e implementación de su software, mejorándolo semana a semana y concluyendo con un producto completo y listo para ser presentado.

La Feria es la instancia que otorga la carrera de Ingeniería Civil Informática para que los alumnos vivan la experiencia de gestionar y desarrollar un proyecto de *software* completo, con un cliente real, haciendo uso de metodologías vigentes en la industria y herramientas de vanguardia, de tal modo que puedan experimentar de primera mano el ejercicio de una de las áreas de desarrollo de la profesión enmarcada en un contexto académico. Para esto, el trabajo colaborativo se vuelve esencial durante las asignaturas, ya que todo lo que deba realizarse se distribuye en grupos de 5 a 6 personas donde cada miembro tiene un rol importante a desempeñar, ya sea en la generación de documentos, planificación del proyecto o desarrollo del mismo.

1.2. Problema

La Feria de Software, al ser una asignatura universitaria, también debe ser constantemente supervisada y evaluada por sus profesores, por lo que estos últimos deben generar directrices sobre lo que los alumnos deben enviar como entrega y las rúbricas con las que estas se evaluarán. Tal tarea tendría que ser de relativa facilidad en una asignatura que se realiza año a año, pero considerando el enfoque que esta tiene de acercar a los alumnos a la industria aplicando metodologías de trabajo vigentes y tecnologías de vanguardia, las bases de la Feria suelen estar cambiando constantemente para adaptarse a la actualidad. Es por lo anterior que la tarea de crear los cimientos de los entregables, incluyendo sus rúbricas de evaluación correspondientes, puede volverse una labor que requiera mucho esfuerzo por parte de los docentes, considerando además que estos últimos deben trabajar de forma conjunta y coordinarse plenamente para conseguir este objetivo, lo que añade una dificultad extra en comparación al trabajo en solitario.

Realizando un análisis más profundo sobre la planificación de la asignatura [3] se logró obtener el desglose de los procesos que conforman a esta y los distintos entregables de cada uno, tal como se puede apreciar en el **Apéndice A.1**. Con esto en consideración, resulta sencillo notar que tan solo el primer semestre se tienen en evaluación cerca de 8 entregables, donde cada uno de estos tiene información de gran relevancia para los

alumnos en el transcurso del curso. Complementando lo anterior, una gran cantidad de estos entregables posee una gran extensión y se desarrollan de forma iterativa, trabajando su contenido constantemente y provocando un exceso de documentación redundante; en consecuencia, los alumnos suelen tener problemas cuando, en etapas posteriores de la asignatura, necesitan de algún elemento que han desarrollado previamente, viéndose obligados a buscar entre todas las versiones de sus archivos nuevamente para encontrar aquellos elementos deseados (por ejemplo, la búsqueda de la última versión de sus historias de usuario y sus estimaciones en la planificación). Además, como mucha de esta información está siendo constantemente trabajada y corregida, no es extraño que ciertas definiciones se repitan en distintos entregables para reflejar la evolución que estas van teniendo, dificultando aún más la tarea de realizar búsquedas de una posterior versión final.

Dado que cada entregable es evaluado por los docentes y ayudantes de la asignatura, estos poseen una rúbrica respectiva para la asignación de puntos, derivando finalmente en una calificación final. Con esto en consideración, la asignatura debe hacer uso de una gran cantidad de planillas y documentos de cálculo para entregar los resultados a los alumnos (entre el puntaje de cada elemento, ponderación, comentarios y calificación final), donde la cantidad de estos últimos es de, por lo menos, 100 estudiantes entre ambos campus. Lo mencionado reitera la problemática presentada en el párrafo anterior, ya que la cantidad de planillas y documentos dificulta la tarea de buscar alguna calificación o comentario en particular tanto para alumnos como para profesores, viéndose agravada por la dificultad de restauración, en caso de algún error, por los problemas de trazabilidad ya presentados.

Complementando lo anterior, la asignatura no posee alguna plataforma que unifique sus tareas, ya que esta hace uso de distintas herramientas para cubrir las necesidades pertinentes; entre ellas podemos mencionar a *Moodle* para la entrega de información, *Google Drive* para el manejo de planillas de notas, *YouNoodle* para la recepción de ciertos entregables, entre otros. Con esto en mente, la trazabilidad de los documentos se complica aún más, añadiendo el exceso de información que el manejo de distintas plataformas puede provocar en el usuario.

Por último, el resultado al englobar todo lo presentado hasta el momento conlleva a una gran dificultad, tanto para docentes como para ayudantes, sobre llevar una trazabilidad constante de los proyectos en desarrollo; esto dada la imposibilidad de ver los progresos actualizados de cada equipo (lo cual solo se hace a través de reuniones),

aumentando la probabilidad de incurrir en fallas durante el desarrollo de los proyectos, cambios arbitrarios sin aprobación docente, inconsistencias entre la planificación y el desarrollo entregado o, por consecuencia de lo anterior, la retirada de la asignatura por incumplimiento en etapas demasiado avanzadas como para hacer algo al respecto.

1.3. Objetivos

1.3.1. Objetivo General

Modernizar los procesos de gestión asociados a las asignaturas de Feria de Software creando una plataforma web, apoyándose en la transformación digital, que automatice el proceso de gestión, seguimiento y evaluación de cada equipo participante de estas asignaturas, de tal manera que su desarrollo sea más eficaz y eficiente.

1.3.2. Objetivos Específicos

1. Capturar y estudiar los requerimientos esenciales, en conjunto con los docentes, para cubrir las problemáticas presentadas.
2. Crear una plataforma web con los requerimientos detectados, haciendo uso de los estándares vigentes en la producción de herramientas de gestión, que permita mejorar el desarrollo, seguimiento y evaluación de los proyectos candidatos a participar en la Feria de Software.
3. Validar la efectividad de la herramienta implementada a través de su uso por parte de los docentes y mediante su aplicación en ciertos equipos de las asignaturas.

Capítulo 2

Marco Conceptual

En el presente capítulo se explicarán y definirán los principales conceptos necesarios para comprender la propuesta que se busca concretar en esta memoria, centrándose específicamente en los conceptos de transformación digital, herramientas de gestión, versionamiento y automatización del proceso de producción de software, junto a las distintas tecnologías para su implementación.

2.1. Transformación Digital

Mucho se ha hablado en las últimas décadas acerca de la transformación digital y la necesidad de las empresas por enmarcarse en este concepto; sin embargo, la concepción de este proceso no es clara para todos quienes lo usan. Según la empresa Red Hat [4], la transformación digital es la adopción de tecnologías (TI) para la mejora de los procesos de una organización con distintos objetivos en mente, los cuales pueden ser mayor rapidez, menores pérdidas, mejores respaldos, entre otros. Sin embargo, y a pesar de que el concepto de transformación digital pareciera ser relativamente sencillo de comprender, uno de los más grandes pecados cometidos en su aplicación es la creencia de que solo consta de la modernización tecnológica de la organización; y es que como su nombre sugiere, “transformación” es un concepto más grande como para solo englobar la incorporación de tecnologías, sino que además debe abarcar la evolución de la cultura organizacional que la respalda. Es decir, para que una transformación digital

sea efectiva, se deben incorporar las tecnologías que mejoren los distintos procesos de la organización, además de capacitar al personal que hará uso de él y concientizar a las distintas partes para que comiencen a asimilar que la modernización no es solo la adquisición de herramientas modernas, sino que es cambiar la mentalidad y asumir que este proceso es constante; la transformación digital no tiene un final claro, sino que se avanza buscando constantemente formas de seguir mejorando.

Según una encuesta realizada e impulsada por Red Hat [5], el 86 % de las organizaciones líderes consultadas concuerdan en que la transformación digital exige la combinación correcta de procesos, tecnología y cultura; sin embargo, un 63 % considera que el cambio cultural es el reto más grande. Por otro lado, el 73 % de los encuestados afirma que la tecnología es de suma importancia para adaptarse rápido a la demanda de los clientes, pero solo el 31 % considera que sus inversiones han sido fructíferas con respecto a lo esperado, dado que es dificultoso incorporar estos cambios correctamente.

En el proceso de modernización, las tecnologías que se han vuelto más necesarias son diversas dependiendo del objetivo que se tenga en mente al realizar la transformación o del propósito que cumple la misma organización (ya sea consultora, cadena hotelera, retail u otros). Según lo anterior, las tecnologías que se han visto más requeridas son el cambio de infraestructura de la información por una versión en la nube, el desarrollo especializado de aplicaciones web de alto rendimiento o la incorporación de herramientas digitales para agilizar distintos procesos existentes. En el marco del desarrollo de software (y por ende del presente documento), se indagará sobre el proceso de gestión de proyectos y la automatización en la producción de software.

2.2. Metodología de trabajo *SCRUM*

A la hora de desarrollar un proyecto se tiene que tener en consideración la metodología de trabajo con la cual este será implementado, ya que es una parte esencial sobre su desarrollo. Esta metodología debe ajustarse a las necesidades del equipo y el proyecto, como por ejemplo los plazos, el presupuesto, el tamaño del equipo, entre otros factores; lo anterior debido a que la metodología establece las directrices que el equipo tendrá que seguir para llegar a la culminación de su proyecto, y estas deben trabajar en concordancia con la organización misma del proyecto.

Las metodologías ágiles, según [6], son un tipo de metodología que hacen uso de un modelo iterativo de trabajo en donde se realizan entregas cíclicas y en cada entrega se realizan todas las fases del ciclo: desde toma de requerimientos, diseño, verificación y entrega. Su gran diferencia con las metodologías tradicionales es la entrega valor y la retroalimentación constante durante todo el proyecto. De esta forma, las planificaciones se realizan en periodos cortos y definidos de tiempo llamados *sprints*, luego de los cuales los equipos reflexionan sobre el trabajo realizado y buscan mejoras para las etapas venideras.

La metodología *SCRUM*, según [7], es una metodología ágil que permite a los equipos gestionarse por cuenta propia, aprender a partir de la experiencia y adaptarse al cambio. Esta cuenta con los siguientes artefactos:

- **Product Backlog:** lista dinámica de características, requisitos, mejoras y arreglos que se deben completar para que el proyecto tenga éxito.
- **Sprint Backlog:** lista de los elementos que el equipo de desarrollo debe completar durante el ciclo *sprint* en curso. Antes de cada *sprint*, el equipo escoge en qué elementos trabajará a partir del *Product Backlog*. Un *sprint backlog* es flexible y puede evolucionar durante un *sprint*.
- **Incremento:** producto final y utilizable que surge a partir de un Sprint.

Los artefactos mencionados son obtenidos mediante distintas ceremonias que se practican en esta metodología, entre las cuales se pueden encontrar:

- **Sprint Planning:** este evento se realiza antes de cada *sprint*, y es donde el equipo estima el trabajo que se debe completar durante este. Los miembros definen los objetivos del *sprint*, que a su vez deben ser específicos, medibles y viables. Tras esta reunión cada miembro del equipo debe comprender cómo se puede entregar cada incremento planificado.
- **Sprint:** periodo durante el cual el equipo trabaja conjuntamente para la obtención de un incremento. La duración más común es de dos semanas, pero que suele variar según las necesidades del proyecto y el equipo. Entre más complejo sea el trabajo y más factores desconocidos haya, el periodo del *sprint* debe ser más corto.

- Daily Meeting: sesión breve en la que los miembros del equipo se reportan y planifican el día. Informan sobre el trabajo concluido y expresan cualquier desafío para alcanzar los objetivos.
- Sprint Review: evento llevado a cabo al finalizar un *sprint*, donde el equipo se reúne como una sesión informal para revisar el trabajo concluido y exponerlo a los interesados. Es posible que el *product owner* también ajuste el *product backlog* en función de lo expuesto.
- Sprint Retrospective: reunión donde el equipo de trabajo documenta y habla sobre qué funcionó y qué no durante el *sprint*. Las ideas generadas se utilizan para mejorar las próximas iteraciones.

Los eventos mencionados anteriormente son realizados por el equipo de trabajo, donde destacan los siguientes roles:

- Product Owner: es quien se concentra en garantizar que el equipo proporcione el mayor valor al negocio. Comprende y prioriza las necesidades cambiantes de los usuarios finales y de los clientes.
- Scrum Master: se encargan de velar por la metodología *scrum* dentro de los equipos de trabajo. Son responsables por la efectividad del equipo y su formación, incluyendo la formación de los *product owner* y al negocio para mejorar los procesos de *scrum* y optimizar la entrega.
- Development Team: es el equipo que llevará a cabo el desarrollo del proyecto en cada *sprint*, cuyos miembros cuentan con habilidades diferentes y se entrenan entre sí de modo que ninguna persona se convierta en un cuello de botella para la entrega del trabajo.

2.3. Herramientas de Gestión de Proyectos

Para la correcta gestión de un proyecto, el uso de una herramienta no es esencial (pero sí de ayuda) ya que las variables que se deben tener en consideración son manejables para un equipo. Sin embargo, cuando una organización posee una gran cantidad de

proyectos, las variables (como equipos de trabajo, miembros, plazos, avances u otros) se incrementan en gran medida y su gestión se vuelve un trabajo que, incluso, puede escapar de las manos para una oficina manejada por personas, por lo que el uso de una herramienta pasa a ser algo de carácter esencial; dado lo anterior es que surgen las herramientas digitales de gestión para apoyar al funcionamiento de los procesos de gestión existentes, las cuales suelen estar alojadas en la nube y forman un entorno de trabajo colaborativo. Según Abellán [8], el uso de estrategias y técnicas en la gestión de proyectos provoca que cerca de un 80 % de los proyectos alcancen las metas propuestas dentro de los límites establecidos, es decir, que culminan como proyectos exitosos. En consecuencia, el uso de una herramienta de gestión aumenta la probabilidad de éxito en el desarrollo de un proyecto ya que estas proveen distintas directivas y soluciones para su correcta administración.

Entre los distintos beneficios que pueden proporcionar las herramientas de gestión, se pueden listar los siguientes:

- Organizar los tiempos de un proyecto.
- Proporcionar información para estimar de forma correcta tiempos y costes.
- Ayudar a gestionar y minimizar los riesgos del proyecto.
- Mejorar la rentabilidad.
- Desarrollar las habilidades del equipo.
- Mejorar la productividad.
- Conocer de un vistazo y el tiempo real el grado de ejecución de una tarea.

A la hora de elegir una herramienta para una organización se debe tener en consideración las necesidades de esta última, ya que no hay una herramienta definitiva que cumpla con todos los requerimientos de la industria. Por el contrario, existe una gran gamma de estas dependiendo las necesidades de cada usuario, por lo que se debe realizar un trabajo previo para buscar aquella que se adapte mejor a lo requerido y que permita explotar las ventajas de la organización. Complementando lo anterior, existen distintas características esenciales que una herramienta debe presentar para que pueda sacar el máximo provecho dentro de los proyectos, en donde se pueden mencionar las siguientes:

1. Accesibilidad: el usuario debe ser capaz de ingresar al sitio a través de cualquier dispositivo y cualquier navegador.
2. Colaboración: el sistema debe ser capaz de albergar varios usuarios para que estos puedan relacionarse a través del sitio y fomentar el trabajo en equipo. Para esto se suelen agregar herramientas extras como chat, administración de permisos u otros.
3. Usabilidad: la interfaz usuaria debe facilitar la navegación entre los distintos proyectos y sus características, de tal forma que su uso no dificulte aún más la gestión.
4. Organización: el sistema debe permitir establecer plazos, asignar recursos, presentar tiempos de ejecución y establecer dependencias entre tareas.
5. Visualización: la interfaz debe permitir que el usuario pueda apreciar la información de los distintos proyectos en un panel para realizar un correcto seguimiento de estos con el objetivo de planificar y tomar decisiones. Para esto se emplean distintos métodos como Kanban o Gantt.
6. Tiempo Real: la información dispuesta debe estar actualizada en cada momento para evaluar la correcta evaluación de los proyectos.
7. Comunicable: el sistema debe contar con medios para compartir la información con otros sistemas (como mediante una API) para que sean utilizados en otros entornos con distintos objetivos dentro de la organización.
8. Seguridad: el sistema debe cumplir con los requisitos de privacidad necesarios para que este no pueda ser adulterado.
9. Metodología de trabajo: el sistema debe ser capaz de soportar la metodología de trabajo seleccionada para el proyecto en que será usada, entre las cuales se barajan *scrum*, tradicional, híbrida, entre otras.

La cantidad de herramientas de gestión de proyectos es tan grande en la actualidad, cada una ofreciendo alguna característica que las diferencie de sus competidoras, que realizar una reseña de cada una se vuelve una tarea imposible. Es por esta razón que se indagó sobre las más recomendadas y populares (según [9], [10], [11], [12], [13] y [14]) para analizar cuales son las distintas ventajas y desventajas de cada una considerando los

criterios ya mencionados anteriormente, incluyendo además elementos como precio, cupos, número de proyectos, entre otros.

1. Trello

■ Ventajas:

- Visualización mediante Kanban.
- Fácil de usar gracias a tarjetas con *Drag and Drop*.
- Tableros personalizables.
- Posee versión gratuita con acceso a la gran mayoría de funcionalidades.
- Se puede conectar con otras aplicaciones.
- Gran enfoque en la colaboración del equipo.
- Posibilidad de crear tareas recurrentes.

■ Desventajas:

- No proporciona herramientas de planificación temporal como calendarios o Gantt.
- Al ser tan visual, en ocasiones resulta difícil encontrar tareas específicas.
- No posee un tablero que muestre el progreso de los proyectos.
- La versión gratuita permite el uso de hasta 10 tableros.
- Los planes de pago comienzan de forma mensual por usuario.

2. Asana

■ Ventajas:

- Capacidad de crear plantillas para automatizar tareas cotidianas.
- La opción de establecer controles de seguridad y designar administradores.
- Más de 100 integraciones.
- Campos de proyectos personalizados, compartir documentos y filtrar tareas.
- Gráficos de progreso de proyectos.
- Posee chat integrado.
- Gratis para proyectos ilimitados para equipos de hasta 15 personas.

- Desventajas:
 - Totalmente en inglés, lo que dificulta su uso a algunos usuarios.
 - Al iniciar con la herramienta, toma bastante tiempo comenzar a organizar las tareas.
 - El costo para más miembros es por cada usuario a agregar.

3. Jira

- Ventajas:
 - Capacidad de crear tableros *Kanban* y tablas de *SCRUM* Scrum personalizadas.
 - Reportes en tiempo real.
 - Gran cantidad de integraciones.
 - Gran enfoque y aporte frente a metodologías ágiles.
 - Permite su uso gratuito ilimitado para equipos de hasta 10 personas.
- Desventajas:
 - Toma gran tiempo aprender a utilizarla.
 - Su enfoque ágil dificulta adaptarse a otros métodos.
 - El precio se cobra por usuario al mes.
 - Enfoque principal de un solo proyecto.

4. Monday

- Ventajas:
 - Larga lista de integraciones.
 - Fácil de personalizar un tablero o *workflow*.
 - Personalización de colores y diseños.
 - Posee automatización de procesos.
 - Cuenta con todas las características esenciales y mucho más.
- Desventajas:
 - Poca claridad en los planes que ofrecen.
 - Es completamente de pago, con su plan más básico para 2 personas.

5. Wrike

■ Ventajas:

- Presenta variadas maneras de mostrar la información.
- Variadas opciones de personalización.
- Integración con cientos de plataformas.
- Gratis hasta 5 miembros.

■ Desventajas:

- Los planes son por usuarios, y requieren un mínimo de 5 usuarios más.
- La versión gratuita presenta limitaciones esenciales del sistema.

6. Teamwork

■ Ventajas:

- Cuenta con un chat integrado para los miembros.
- Seguimiento del tiempo de los miembros y sus actividades.
- Los paneles, subtareas e hitos ayudan a rastrear el progreso de los proyectos.
- La versión gratuita permite el acceso de hasta 2 proyectos y 5 miembros.

■ Desventajas:

- Los planes se cobran por cada usuario.
- Su versión gratuita es inviable para varios proyectos.

2.4. Herramientas de Versionamiento

A la hora de trabajar con documentos, el correcto manejo de sus versiones resulta vital para tener un mejor desempeño. La posibilidad de listar los cambios que un archivo ha tenido y poder volver a versiones previas resulta de gran utilidad, además de permitir la sincronización de estos entre distintos miembros de la organización en lugar de enviar un archivo por correo una y otra vez con distinto nombre. Esto ha permitido mejorar el desempeño de distintas áreas dentro de las organizaciones, ayudando a mejorar el control de los documentos, permitiendo ver su evolución y previniendo errores inesperados.

Según Dropbox [15], herramienta especializada en versionamiento, el control de versiones es el proceso de administrar diferentes versiones de los archivos a lo largo del tiempo, lo cual se hace normalmente usando un sistema de control de versiones. Existen distintas formas de implementar una herramienta de control de versiones, entre las que se encuentra el rastreo de los cambios de un archivo para tener almacenadas las distintas versiones o el manejo de una arquitectura de carpetas donde se guarden las distintas versiones de un mismo archivo con distinto etiquetado. Para esto es importante tener clara una convención para el manejo de los nombres de los documentos, de tal forma que su diferenciación sea de fácil comprensión tanto para la herramienta como para los usuarios (ya sea haciendo uso de fechas o de códigos cronológicos). Además de esto, es importante tener un correcto etiquetado de las carpetas que organizan estos archivos, creando así una “arquitectura de la información” capaz de mantener distintas versiones de archivos completamente ordenados.

Un software de control de versiones debe realizar las tareas mencionadas previamente de forma automática, induciendo los nombres de los distintos archivos y organizándolos correctamente con unos pocos clicks, de tal forma que se pueda acceder fácilmente a la versión previa de un archivo y teniendo copias de seguridad de los distintos elementos. Gracias a esto es que los equipos pueden colaborar en un mismo documento, permitiendo su evolución y correcta administración para concluir en el resultado esperado.

Gracias a una herramienta de control de versiones se pueden tomar decisiones sobre qué debe estar en el documento final y qué no, ya que se pueden administrar los cambios provenientes de los distintos miembros de un equipo. La herramienta debería permitir la solicitud de aprobaciones para incorporar cambios en un archivo delicado, de tal forma que se tenga claridad constante de qué se está incorporando y previniendo que alguno de los miembros pueda corromper un documento por accidente (o a propósito). Como consecuencia, la administración de los documentos se vuelve un proceso mucho más transparente, permitiendo además analizar el comportamiento de los miembros de un equipo dentro del trabajo realizado.

Al igual que con las herramientas de gestión de proyectos, existe un sinnúmero de plataformas que permiten el versionamiento de documentos. Entre las más utilizadas yacen Dropbox, OneDrive y Google Drive (según [16] y [17]), cuyas principales diferencias se enumeran a continuación:

1. Dropbox

■ Ventajas:

- Cuenta con grandes protocolos de seguridad.
- Cuenta con acceso a distintos tipos de documentos de forma gratuita como hojas de calculo o presentaciones.

■ Desventajas:

- Ofrece solo 2GB de almacenamiento para los archivos deseados en su plan gratuito.
- Su plan más económico es de 1TB como mínimo, siendo de alto costo para organizaciones más pequeñas.

2. OneDrive

■ Ventajas:

- Cuenta con grandes protocolos de seguridad.
- Viene pre-instalado en las distribuciones actuales de Windows, lo que le permite tener mayor dominio sobre el sistema.

■ Desventajas:

- Ofrece solo 5GB de almacenamiento para los archivos deseados en su plan gratuito.
- Las tarifas permiten obtener solo 50GB como mínimo en su plan más económico.
- Cuenta con acceso a distintos tipos de documentos de Office 365 solo en sus planes de pago.

3. Google Drive

■ Ventajas:

- Ofrece hasta 15GB de almacenamiento para los archivos deseados en su plan gratuito.
- Las tarifas permiten obtener 100GB como mínimo en su plan más económico.
- Cuenta con acceso a distintos tipos de documentos de forma gratuita como hojas de calculo o presentaciones.

- Desventajas:
 - Cuenta con protocolos de seguridad inferiores a la competencia, o al menos es lo que Google informa.

2.5. Automatización del Proceso de Producción de Software

2.5.1. Control de Versiones

A la hora de desarrollar software de forma colaborativa siempre ha existido la problemática de la integración del código de cada miembro tratando de no morir en el intento. A lo largo del tiempo se han utilizado distintas técnicas para aplacar estos inconvenientes, pero quien realmente logró convertir esta disyuntiva en un beneficio para las organizaciones fue Git [18], la cual es una herramienta de código abierto que dejó atrás a sus competidores para crear un sistema de control de versiones sumamente eficaz y eficiente al implementarse como un sistema de control distribuido capaz de manipular sus distintos hitos de tal forma que el problema del versionamiento pasó a ser una ventaja con el uso de *branches* y áreas de trabajo. Gracias a las herramientas de versionamiento de código, se revolucionó la forma de programar al convertir el proceso de integración en un requisito dentro de los equipos de trabajo; sin embargo, la llegada de estas herramientas abrieron una nueva clase de potenciales características que podrían potenciar aún más el desarrollo de software, dentro de las cuales se pueden nombrar la Integración Continua o la Entrega Continua.

2.5.2. CI: *Continuous Integration*

La integración continua según Amazon [1] es una práctica que permite automatizar distintos procesos clave a la hora de integrar el código de los colaboradores, tales como pruebas, revisión de código, compilación, entre otros. Esta práctica nace por el aislamiento de los miembros de un equipo, quienes al integrar sus códigos notaban que la enorme cantidad de cambios que se procesaban chocaban entre ellos y generaban

distintas fallas dentro de los sistemas. Dado esto, comenzaron a generarse prácticas de integración periódica para que los cambios procesados fuesen los menos posibles, permitiendo que se pudiesen realizar análisis sobre ellos para generar distintas métricas, permitiendo así que se pudiesen realizar pruebas de funcionamiento en el sistema antes y después de que se incorporaran los cambios, realizando chequeos del código y dejando el sistema de desarrollo en estado de funcionamiento. Gracias a esto se pueden identificar errores aislados en fases tempranas de la programación, de tal modo que su corrección resulta una tarea de menor dificultad. Esto, además, provoca que los programadores adquieran las costumbres de estar probando su código constantemente, tratando de evitar la mayor cantidad de errores posibles. Finalmente, todo este proceso permite que el sistema en desarrollo sea más sencillo de disponer a la hora de entregar actualizaciones a los clientes.

2.5.3. CD: *Continuous Delivery*

La entrega continua según Amazon [2] es una ampliación de las prácticas de integración continua, las cuales tienen su enfoque en la preparación del código para los ambientes de producción. Para esto, una vez que el proceso de integración continua ha sido realizado, se lleva el sistema a un ambiente de pruebas y se aplican tests exhaustivos al sistema desarrollado (como desempeño, integración, interfaz, seguridad, u otros), provocando así que siempre se tenga un sistema listo para su paso a producción. A todo lo anterior se le suma una tercera fase, la implementación continua (*Continuous Deployment*), la cual envía automáticamente los cambios realizados a los ambientes de producción una vez que se han pasado todas las pruebas realizadas en las fases anteriores. Como resultado, la entrega continua permite automatizar completamente el proceso de publicación de software, agilizando el proceso en gran medida, mientras que se mejora la productividad de los equipos, se encuentran errores con mayor rapidez y los clientes reciben actualizaciones de manera más eficaz.

Los procesos presentados anteriormente se pueden ver en la **Figura 2.1** para una mejor comprensión del proceso mencionado. A su vez, esta puede verse de mejor forma en el **Apéndice A.2**

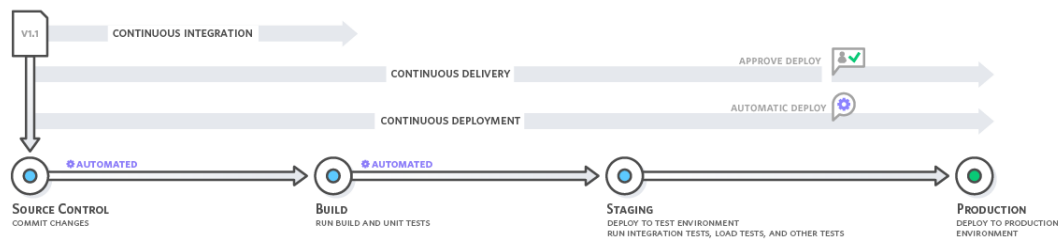


Figura 2.1: Diagrama del proceso de CI/CD de un software en desarrollo [1][2]

2.6. React

Según [19], *React* es una librería para la construcción de interfaces usuarias basada en el uso de componentes y programación reactiva desarrollada por *Facebook* y que en la actualidad es de código abierto, mientras es mantenido tanto por su empresa como por la comunidad. Esta herramienta fue creada para el desarrollo de interfaces usuarias de manera sencilla haciendo uso de componentes, los cuales son piezas individuales de código funcional que encapsulan su propia funcionalidad y que exponen elementos de la interfaz. Estos se pueden mezclar y reutilizar para crear interfaces y vistas, los que finalmente permiten obtener una aplicación web completa.

Otra de sus principales características (y la razón del nombre de la librería) es su reactividad, donde la interfaz “reacciona” a los cambios de un grupo definido de datos (llamado “estado”), permitiendo que la aplicación actualice la vista dependiendo de como interactúa la información, reduciendo así la complejidad de desarrollar una interfaz dinámica con *Javascript* puro donde cada interacción debía ser programada manualmente.

Su curva de aprendizaje es bastante sencilla, ya que basa su uso en el lenguaje de programación *Javascript*, permitiendo que el manejo de este lenguaje permita dominar la mayoría de la librería. Es en este apartado donde cabe destacar, según [20], que al ser una librería y no un *framework* de *Javascript*, permite que su curva de aprendizaje no sea tan compleja como la de uno de sus más grandes competidores: *Angular*. Esto se debe a que, para aprender *Angular*, no basta solo con saber *Javascript*, sino que también se tiene que tener expertiz con el lenguaje *Typescript*, sumado a tener que incorporar el ecosistema de funcionamiento de *Angular*.

Una de las grandes ventajas que posee esta librería por sobre otras como *Angular* y *Vue*, según [20], es la comunidad que esta posee, ya que al ser una de las herramientas más utilizadas por desarrolladores, la comunidad se ha vuelto muy activa con respecto a la solución de problemas y resolución de dudas de aquellos que se encuentran aprendiendo. Además, el hecho de que sea desarrollada y mantenida por *Facebook* inspira confianza en los equipos de desarrollo, ya que no es una herramienta que vaya a perder soporte a largo plazo.

2.7. Flask

Según [21], *Flask* es un *microframework* de desarrollo web escrito en el lenguaje de programación *Python* con enfoque al patrón de diseño MVC (modelo vista controlador). Al ser un *microframework*, este incluye solo las funciones esenciales para el desarrollo de aplicaciones, volviéndolo una herramienta muy ligera y sencilla de aprender; mientras que por otro lado, si se desea hacer uso de librerías para expandir las funcionalidad de *Flask*, se pueden incorporar sencillamente.

Al ser un *framework* minimalista, su curva de aprendizaje es bastante poco pronunciada debido a que sus funcionalidades son bastante concisas y descriptivas, por lo que su uso se vuelve mucho más sencillo en comparación a otras opciones más robustas (como por ejemplo *Django* [22]).

Uno de los usos que más se le da a *Flask* es el desarrollo de servicios web (como *API RestAPI*) ya que posee enrutamiento nativo, el cual permite estructurar la aplicación según las rutas y los controladores a los que se comunican. Además, existen librerías que permiten conectar *Flask* a motores de base de datos, por lo que es posible incorporar estas funcionalidades en sus aplicaciones de forma sencilla.

Finalmente, al ser un *microframework*, resulta una excelente opción para la construcción de microservicios, ya que los servicios creados con *Flask* suelen ser pequeños y muy ligeros.

2.8. Bases de datos NoSQL

Las bases de datos *NoSQL* [23] son un tipo de bases de datos que almacena la información de una manera distinta a como lo hacen las bases de datos tradicionales. Al no ser relacionales, suelen proponer alternativas para las consultas de los datos, por lo que ya no usan el lenguaje de consulta *SQL*, característica que les da su nombre. A pesar de lo disruptivo que este concepto pueda sonar, su uso se ha masificado en gran medida en los últimos años, ya que suelen poseer elevados niveles de escalabilidad y disponibilidad. Además, al no tener una estructura rígida como las bases de datos relacionales, se suelen moldear para que entreguen mayor conveniencia al negocio y sea más fácil de entender no solo para los desarrolladores, sino que también a otro tipo de usuarios. Es por esta misma razón que su uso en proyectos muy cambiantes (como los desarrollados incrementalmente) se ha vuelto una excelente opción, ya que permiten incorporar modificaciones al esquema de la misma con facilidad debido a su gran flexibilidad. Entre sus principales características se encuentran:

- Flexibilidad: pueden almacenar la información de forma más libre sin la rigidez de los sistemas relacionales. Esto permite almacenar datos estructurados, semiestructurados y no estructurados en un único almacén sin tener la preocupación de seguir un esquema.
- Escalabilidad: En lugar de escalar verticalmente agregando más servidores, las bases de datos *NoSQL* pueden escalar horizontalmente utilizando *hardware* básico. Así se puede permitir un tiempo de respuesta óptimo frente a un aumento en el volumen de tráfico.
- Alto rendimiento: al tener un escalado horizontal, se pueden garantizar tiempos de respuesta rápidos y fiables, siendo una de las mejores opciones para aplicaciones que tengan un amplio tráfico de datos o grandes cantidades de información.
- Disponibilidad: la información se suele replicar en servidores, centros de datos o la nube para reducir la carga en la base de datos, minimizando la latencia de los usuarios y reduciendo la carga sobre la misma.
- Altamente funcional: están diseñadas para almacenes de datos distribuidos que precisan una capacidad de almacenamiento de datos extremadamente grande. Esto

vuelve a las bases de datos *NoSql* una de las mejores opciones para el manejo de información masiva como *big data*, juegos en línea u otros.

2.8.1. MongoDB

Una de las bases de datos *NoSQL* más populares según una encuesta realizada por la plataforma *Stack Overflow* [24] es *MongoDB*, siendo la más utilizada por los desarrolladores. *MongoDB* es una base de datos no relacional de documentos que ofrece gran escalabilidad y flexibilidad [25]. Los documentos que almacena tienen una estructura similar a *JSON*, permitiendo que su estructura pueda variar entre documentos o con el tiempo. Al tener una estructura tipo *JSON*, se vuelve muy sencillo trabajar con los datos dentro de las aplicaciones, donde las librerías de *MongoDB* entregan la información en alguna estructura de datos que soporte esta estructura dependiendo del lenguaje utilizado (como diccionarios en *Python* u objetos en *Javascript*). Sumado a lo anterior, la indexación que este motor realiza sobre sus documentos y otras características como las operaciones de agregación ofrecen maneras potentes de acceder a los datos y analizarlos. Finalmente, existe un servicio en la nube llamado *Mongo Atlas* [26], el cual permite manejar bases de datos *MongoDB* completamente en línea, distribuyendo y optimizando los datos de manera automática, lo que permite al usuario enfocarse solo en hacer uso de esta.

2.9. Amazon Simple Storage Service: S3

El servicio *S3* que ofrece *Amazon* [27] es un sistema de almacenamiento de objetos en la nube creado para recuperar cualquier volumen de datos desde cualquier ubicación que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Este permite almacenar y proteger cualquier cantidad de datos para prácticamente cualquier caso de uso, como por ejemplo archivos de datos (imágenes, documentos, entre otros.) o ejecución de aplicaciones nativas. Este puede ser manipulado directamente en la plataforma ofrecida por *Amazon*, y ofrece diversos canales de comunicación a través de su *API* con la cual puede conectarse con distintas aplicaciones.

2.10. Amazon Amplify

El servicio *Amplify* que ofrece *Amazon* [28] es un conjunto de herramientas (marco de código abierto, entorno de desarrollo visual, consola) y servicios (alojamiento de sitios web estáticos y aplicaciones web) para acelerar el desarrollo de aplicaciones web y móviles. Esta permite alojar aplicaciones web de forma rápida y sencilla en la nube de *Amazon*, permitiendo una gran gamma de librerías y *frameworks* existentes para su construcción automática. Por último, ofrece distintas herramientas como el despliegue automático haciendo uso de *CI/CD* (mencionado en la **Sección 2.1**, la posibilidad de incluir *backend* y bases de datos en el mismo proyecto, escalado, certificado *SSL/TLS*, entre otros.

2.11. Amazon EC2

El servicio *EC2* que ofrece *Amazon* [29] proporciona capacidad de computación escalable bajo demanda en la nube de *AWS*. El uso de este servicio reduce los costos de *hardware* para que se puedan desarrollar e implementar aplicaciones con mayor rapidez, y puede ser utilizado para lanzar una mayor cantidad de servidores virtuales, configurar la seguridad y las redes, además de administrar el almacenamiento de forma manual.

2.12. Amazon Route 53

El servicio *EC2* que ofrece *Amazon* [30] es un servicio web de sistema de nombres de dominio o *DNS* escalable y de alta disponibilidad. *Route 53* conecta fácilmente las solicitudes de los usuarios con las aplicaciones de internet que se ejecutan en *AWS* o en cualquier sitio seguro.

2.13. Principios de Nielsen

Los 10 principios de Nielsen (también conocidas como las heurísticas de Nielsen) son, según [31], una guía para el desarrollo de interfaces usuarias que permiten mejorar su usabilidad y la experiencia del usuario en los sitios web. Estos principios son usados como un estándar a la hora de desarrollar interfaces, y ha permitido disminuir las barreras en el uso de la web para distintos grupos etarios. Los principios de Nielsen, de forma resumida, son los siguientes:

1. Visibilidad del estado del sistema: este principio elude a que el sistema entregue información constantemente sobre lo que está haciendo por dentro para que, de tal forma, el usuario tenga noción de lo que ocurre y disminuya su incertidumbre. Esto se ejemplifica con barras de carga cuando la aplicación realiza una actividad que tarda más tiempo, un mensaje de éxito al hacer una acción, cambios de colores en botones luego de clicarlos, entre otros.
2. Concordancia entre el mundo real y el sistema: este principio elude a que el sistema debe ser lo más parecido al entorno real de los usuarios, lo que significa hacer uso de su lenguaje y de imágenes que recuerden a objetos o señales de la vida real, permitiendo así que los usuarios puedan reconocerlos rápidamente y que se sientan en comodidad al hacer uso del sistema. Esto se ejemplifica no utilizando tecnicismos para entregar información, al hacer uso de iconos que representen objetos de la vida real (como un lápiz para editar o una goma para borrar), entre otros.
3. Controles de usuario y libertad: este principio elude a la capacidad de dar libertad a los usuarios en su exploración del sistema mediante la recuperación rápida de errores o su fácil corrección, lo que a su vez entrega más confianza a la hora de su interacción. Un ejemplo de esto sería la opción de deshacer la última acción realizada.
4. Consistencia y estándares: este principio elude a la consistencia que una aplicación debe tener tanto internamente como externamente. Esto significa que el sistema debe ser consistente entre sus componentes, por ejemplo manteniendo los mismos colores, fuentes, diseños u otros, mientras que por otro lado, debe presentar consistencia sobre elementos que se han vuelto comunes por otras aplicaciones

populares, como por ejemplo el presentar las secciones de la aplicación en la barra superior, el tener información de contacto en la barra inferior, el botón de menú tipo “hamburguesa” en dispositivos móviles, entre otros.

5. Prevención de errores: este principio elude a la capacidad de un sistema para prevenir errores del usuario mediante señales, mensajes, validaciones u otros. De esta forma se da ayuda a los usuarios para que incurran en menos errores, como por ejemplo el señalar cuando se escribe mal una contraseña o el validar si realmente se desea eliminar un elemento.
6. Reconocer es mejor que recordar: este principio elude a añadir elementos que sean fácilmente reconocibles en lugar de información que requiera un mayor proceso mental para su comprensión. Por ejemplo, es más rápido leer la hora en un reloj digital que en uno analógico, porque ver la hora directamente no conlleva el esfuerzo de interpretar la posición de las manecillas del reloj. Otro ejemplo sería tener el acceso a un carro de compras en la barra de navegación con el icono del carro en lugar de un texto plano que diga “Carro de compras”, donde el icono es fácilmente reconocible.
7. Flexibilidad y eficiencia de uso: este principio elude a la capacidad adaptativa de los sistemas a los distintos tipos de usuarios que podrían hacer uso de este, entregando ayuda a usuarios que tengan un menor conocimiento técnico, mientras se entregan aceleradores y atajos que permitan optimizar el manejo de quienes sean más experimentados. Como ejemplo se puede nombrar la posibilidad de aumentar el tamaño de fuente para usuarios con visión reducida, o el permitir hacer uso de atajos de teclado para usuarios más experimentados.
8. Estética y diseño minimalista: este principio elude a la presentación de la información que solo sea fundamental, sin contaminar una vista con elementos innecesarios que puedan distraer de lo realmente importante; en otras palabras, menos es más. Ejemplo de esto serían no repetir información que ya se encuentra en la pantalla, el hacer uso de pocos colores para no sobrecargar la visión o tratar de hacer uso de menor espacio en pantalla para centrar la atención en una menor más reducida.
9. Ayudar a los usuarios a reconocer, diagnosticar y solucionar errores: este principio elude a la correcta información por parte del sistema ante la ocurrencia de errores.

Además, el sistema debe ser capaz de entregar las herramientas para solucionar estos errores, como por ejemplo la opción de recargar la página cuando se entrega un error 404, o la opción de mantener la información de un formulario si su envío fallase.

10. Ayuda y documentación: este principio elude a la capacidad de un sistema para educar a los usuarios sobre su funcionamiento, ya sea mediante el uso de manuales, guías o mensajes del sistema que den explicación breve sobre su funcionamiento. Ejemplos de esto son los manuales de uso o los procesos de *onboarding* en una pantalla.

Capítulo 3

Propuesta de Solución

3.1. Alternativas de Soluciones

Luego de la investigación realizada se buscaron posibles opciones para solucionar el problema previamente propuesto. La solución debe contar con distintas características esenciales para que pueda ser de real utilidad dentro de las asignaturas de la Feria de Software. Entre las necesidades se pueden nombrar las siguientes:

- Se debe permitir gestionar los proyectos tanto por los docentes, ayudantes, jueces y alumnos.
- Debe contar con una interfaz que permita la visualización del estado actual de los proyectos para una correcta trazabilidad de estos.
- Debe permitirse la gestión de documentos para que los docentes, alumnos y ayudantes puedan administrar los entregables de la asignatura.
- Debe existir la capacidad de enlazar los entregables a sus respectivas rúbricas para tener consistencia y transparencia en las evaluaciones.
- Debe existir una forma de ver las evaluaciones y los resultados de los entregables de la asignatura de forma directa y sencilla tanto para alumnos, docentes y ayudantes. Además, quienes evalúan deben ser capaces de ingresar tales evaluaciones en base a las rúbricas.

- El sistema debe contar con la administración de tareas como historias de usuario para su constante actualización por parte de los alumnos y su correcta visualización por parte del cuerpo docente.
- Se deben poder organizar los tableros de los distintos *sprints* para el análisis de cada etapa.
- Debe existir restricción de permisos dependiendo del rol de cada usuario para limitar el acceso a ciertas áreas y tareas.
- Debe ser posible realizar una planificación temporal de los elementos a tratar, de tal forma que se habiliten secciones de los formularios y otros elementos dependiendo del periodo en que se encuentre.
- La restricción de permisos debe abarcar la vista y edición de elementos como formularios, rúbricas, evaluaciones, *backlog* de *sprints*, contenido de historias de usuarios, entre otros. Además, la redefinición de una historia de usuario debe estar restringida dependiendo de la etapa y solo un docente podría aprobarlo.
- Se debe contar con un registro de todos los cambios importantes.
- Se debe poder administrar los distintos roles.
- El acceso a cada elemento debe ser sencillo de localizar, esperando que esté todo lo más centralizado posible.

Con todos los requerimientos previamente mencionados se barajaron 3 opciones que podrían resolver la problemática, las cuales se detallan a continuación:

3.1.1. Reorganización e integración de elementos actuales

En la actualidad la asignatura se administra haciendo uso de distintas herramientas que permiten la gestión de la asignatura, pero como se mencionó previamente en el **Capítulo 1**, estas son tantas y tan variadas que dificultan las distintas tareas al tener presente la información de cada una. Es por esta razón que se propone realizar una reorganización de las herramientas según lo visto en el **Capítulo 2** mediante la integración de las herramientas. Para esto se propone una administración de los entregables mediante

el versionamiento proporcionado por *Google Drive*, donde los docentes tendrían que organizar los distintos directorios y archivos (como formularios y rúbricas) para que los alumnos tengan sus entregables centralizados, ordenados, versionados y enlazados. A esto se le complementaría con el enlace a otras aplicaciones como *YouNoodle* y *Aula* a través de hipervínculos bien organizados, de tal forma que no sea posible perder sus conexiones. Complementando, se asumiría el uso de *Asana* como herramienta de gestión gratuita para enlazar las historias de usuario, donde además se realizaría una integración de la herramienta con las características de *Google Drive* para que se puedan enlazar distintas evaluaciones y comentarios a las distintas historias trabajadas por los equipos.

Los inconvenientes de esta opción es que la administración de los perfiles dentro de *Google Drive* sería de forma manual ya que la plataforma no cuenta con un nivel de automatización capaz de funcionar por periodos o de bloquear ciertos elementos, además de que la centralización de la información mejora, pero sigue sin ser tan efectiva.

3.1.2. Uso de una herramienta de gestión especializada: *Monday*

Como se revisó en el **Capítulo 2**, *Monday* es una de las herramientas más completas dentro del mercado digital y la más completa dentro de las analizadas, por lo que se vuelve inmediatamente una opción de gran fuerza. Su capacidad de personalización permite el manejo de todas las herramientas necesarias para la gestión, además del manejo de documentos enlazados para los entregables. Cuenta con un *dashboard* para que el cuerpo docente puedan analizar el progreso de todos los proyectos a la vez y proporciona métricas de cada uno de estos. Por otro lado, permite el manejo de perfiles y permisos para limitar el acceso a ciertas secciones, de tal forma que se puedan diferenciar a los docentes de los demás.

Entre los inconvenientes de esta opción se encuentra el costo, ya que la opción más viable dentro de sus planes es la **Estándar**, la cual tiene un costo de 10 *USD* por usuario al mes, lo cual implicaría un costo cercano a los 1000 *USD* para la totalidad del curso de forma mensual (cercano a 750.000 CLP). Por otro lado, y a pesar de lo personalizable de la herramienta, aún habrían varias características que se escapan de la asignatura como la liberación de elementos según el periodo o el versionamiento de historias de usuario y documentos.

3.1.3. Creación de una herramienta a medida

Como última alternativa se barajó la opción de crear una herramienta de gestión de proyectos con enfoque en la docencia según los requerimientos enumerados previamente, haciendo uso de diversas tecnologías y lo presentado en el **Capítulo 2**. Para esto, se diseñaría un sistema que permita el manejo de entregables y la creación de sus respectivas rúbricas, entregando la capacidad de realizar las evaluaciones dentro de la misma plataforma. Se incorporaría el sistema de roles y permisos mencionados, sincronizado con el manejo de los periodos dentro de la aplicación. Tendría presente un *dashboard* con la información vigente de los proyectos, más el manejo de historias de usuario por parte de los alumnos y docentes. Además, de forma nativa se sumarían los requerimientos que las otras alternativas no presentan, formulándola en tal caso como una plataforma diseñada en conjunto con los docentes que cumpla con lo solicitado haciendo uso de tecnologías de vanguardia.

Las desventajas de esta opción es el tiempo de desarrollo que implica, ya que el número de elementos requeridos y su complejidad conllevan a un plazo de trabajo equivalente a varios meses, dependiendo del número de desarrolladores. Además, tiene incorporado el costo que pueda llevar su desarrollo más el costo de las distintas plataformas (como servidores) para su funcionamiento. Finalmente, esta misma tendría que ser mantenida por los docentes o algún encargado dentro del departamento de informática, entregando más responsabilidad a estos mismos.

3.2. Opción seleccionada

Luego de barajas todas las opciones con los docentes, la opción seleccionada fue la creación de una herramienta a medida para la asignatura debido a que es la opción que ofrece mayor facilidad de uso, menor costo y por sobretodo, que abarca la mayor cantidad de funcionalidades requeridas; por el contrario, es la opción que requiere mayor mantención y tiempo de desarrollo, pero los profesores consideraron que estos factores son completamente abordables haciendo una relación costo/beneficio. Tal como se planteó previamente, la opción de reorganizar los elementos actuales seguía sin solucionar varios de los problemas presentes en la asignatura, incluso dificultando

aún más la gestión de los proyectos al tener más variables y trabajo previo a tener en consideración. Por otro lado, el uso de una herramienta especializada como *Monday* tenía un costo excesivamente elevado para lo que realmente se necesita, ya que esta cuenta con muchas características principalmente enfocadas en la industria en contraste a una asignatura informática. Dado todo lo anterior, el desarrollo de una herramienta cubre todas las necesidades presentadas previamente, además de incurrir en un costo aceptable para el departamento de informática. Sobre el tiempo de desarrollo, sería el equivalente al tiempo invertido en la implementación de este documento, por lo que está aceptado dentro de los límites de los docentes y su desarrollador. Finalmente, su uso permitiría que en un futuro se pudiesen añadir nuevas características o hacer uso de su información para generar métricas relevantes, dependiendo solamente de la iniciativa del departamento de informática y sus docentes.

3.3. Requerimientos

Para comenzar con el desarrollo de la herramienta a medida, lo primero a tener en consideración es la captura de los requerimientos necesarios para la aplicación, los cuales se presentan a continuación mediante el uso de historias de usuario:

Historia de Usuario 1
Nombre: <i>Dashboard</i> general de proyectos
Descripción: Como docente quiero poder ver el estado actual de todos los proyectos de mis campus para analizar su progreso y tomar decisiones al respecto.
Criterios de aceptación: - Debe haber una vista de todos los proyectos, la cual debe estar organizada alfabéticamente y con su información desplegada. - Se debe presentar el porcentaje de avance general de los proyectos, además de los porcentajes de avance de cada <i>sprint</i>. - Se debe presentar el nombre del proyecto y del equipo, además de los logos. - Los ayudantes deben poder ver esta información también, mientras que los alumnos solo podrán ver la información asociada a su proyecto.

Tabla 3.1: HU 1: *dashboard* general de proyectos.

Historia de Usuario 2
Nombre: Creación de actividades
Descripción: Como docente quiero crear y editar las actividades de la asignatura para gestionar su funcionamiento y remarcar las etapas correctamente.
Criterios de aceptación: - Al momento de crear se debe seleccionar el periodo de tiempo que abarca. - Debe haber una vista de todas las actividades de la asignatura, organizada por su fecha de fin del desarrollo. - Las actividades deben reflejar su duración y tiempo restante. - Debe existir la opción de aceptar entregas atrasadas, donde se tendrá que ingresar la fecha de atraso y de cierre por separado. - Al ingresar en una actividad se deben presentar todos los elementos asociados a esta, como su descripción, formulario, rúbrica, entre otros. - Se debe poder seleccionar el tipo de actividad, ya sea la entrega de un Documento, una presentación, el desarrollo de un <i>sprint</i>, la creación o edición de historias de usuario, o la asignación de estas últimas a los distintos <i>sprints</i>. - Se debe poder seleccionar el campus al cual pertenece la actividad, siendo las opciones el campus al que esté asociado el profesor o todos los campus. - Se deben poder seleccionar el o los evaluadores de la actividad, pudiendo elegir entre los profesores, ayudantes y jueces asociados al campus de la actividad.

Tabla 3.2: HU 2: creación de actividades.

Historia de Usuario 3
Nombre: Registro e ingreso de usuarios
Descripción: Como usuario quiero poder registrarme e ingresar en el sistema para poder interactuar con él haciendo uso de mi información.
Criterios de aceptación: - Por defecto, al registrarse un nuevo usuario tendrá rol de alumno. - Se debe registrar la información personal del usuario, tal como su nombre, apellido, correo, campus y contraseña. - La contraseña debe poseer un mínimo de 8 caracteres y se debe confirmar si esta es correcta. - La contraseña debe almacenarse de forma encriptada, por lo que no se podrá visualizar por los desarrolladores, administradores, entre otros. - Al registrarse o ingresar un usuario, el sistema debe generar y almacenar un <i>token</i> con el cual se podrá hacer uso de la <i>API</i> de la plataforma.

Tabla 3.3: HU 3: registro e ingreso de usuarios.

Historia de Usuario 4
Nombre: Realización de entregables
Descripción: Como alumno quiero poder subir documentos y dar por terminadas mis actividades para que puedan ser evaluadas.
Criterios de aceptación: - Se deben poder hacer entregables de las actividades asociadas al campus del alumno. - Una vez que haya dado por finalizada una actividad, el equipo ya no podrá seguir haciendo cambios relacionados a esta. - Cuando termine el plazo de la actividad, los alumnos harán sus envíos con la marca de tiempo del atraso si este se encuentra habilitado. - Una vez que se cierre una actividad, los alumnos no podrán realizar más envíos. - Los alumnos deben poder ver todos los entregables pendientes que estos tienen con la información asociada, tal como el tiempo restante o la fecha de término. - Los profesores deben poder ver el estado de los entregables de cada equipo y el número de entregas realizadas para cada actividad.

Tabla 3.4: HU 4: realización de entregables.

Historia de Usuario 5
Nombre: Creación de rúbricas
Descripción: Como docente quiero crear y editar las distintas rúbricas de las actividades de mi campus para que los alumnos tengan claridad de cómo se realizarán las evaluaciones.
Criterios de aceptación: - La rúbrica debe presentarse campo por campo, lo que incluye su descripción y puntaje asociado. - Se podrán crear rúbricas desde cero o editar las ya existentes. - Las rúbricas tendrán que estar asociadas directamente a una actividad. - Los alumnos podrán tener acceso a las rúbricas como lectores. - Las actividades de tipo <i>sprint</i> no podrán tener una rúbrica asociada, ya que su evaluación cumple con un proceso aparte.

Tabla 3.5: HU 5: creación de rúbricas.

Historia de Usuario 6
Nombre: Evaluación de entregables con rúbrica
Descripción: Como docente quiero evaluar a los alumnos mediante el uso de rúbricas para que puedan recibir los resultados de sus actividades.
Criterios de aceptación: - La evaluación se realizará seleccionando el puntaje que más se ajuste a cada criterio de la rúbrica. - En cada sección de la rúbrica se podrá ingresar retroalimentación para que los alumnos comprendan las razones de su evaluación. - Al finalizar la evaluación se podrán ingresar comentarios generales acerca del desempeño. - Cada alumno podrá ver las evaluaciones (la nota final o el detalle) de equipo. - La evaluación solo podrá ser realizada por los evaluadores asignados a aquella actividad. - Deseable: las evaluaciones deberían ser descargables en <i>Excel</i> .

Tabla 3.6: HU 6: evaluación de entregables con rúbrica.

Historia de Usuario 7
Nombre: Listado de usuarios
Descripción: Como profesor quiero poder ver un listado de todos los usuarios del sistema para administrarlos en caso de ser necesario.
Criterios de aceptación: - Se deben poder ver los usuarios del sistema con su información, tal como nombre, apellido, correo, campus y rol. - Se debe poder cambiar el campus y el rol de cada usuario. - Se debe poder filtrar a los usuarios por su nombre, apellido o correo.

Tabla 3.7: HU 7: listado de usuarios.

Historia de Usuario 8
Nombre: Creación de equipos
Descripción: Como alumno quiero poder crear un equipo para poder interactuar en la plataforma con este.
Criterios de aceptación: - Cualquier alumno debe ser capaz de crear un equipo si no posee uno. - Se debe poder ingresar la información del equipo, tal como su nombre, su frase representativa y sus miembros. - Para elegir miembros solo se debiesen tener como opciones aquellos alumnos que no posean un equipo y que sean del mismo campus. - Se debe poder editar esta información posteriormente por si llegan a haber cambios dentro del equipo. - Al crear un equipo se debe crear un proyecto automáticamente, el cual se podrá actualizar posteriormente con la siguiente información: nombre, descripción, correo y redes sociales.

Tabla 3.8: HU 8: creación de equipos.

Historia de Usuario 9
Nombre: Creación y edición de historias de usuario
Descripción: Como alumno quiero tener la opción de crear y editar historias de usuario para planificar el desarrollo de mi proyecto y gestionarlo de mejor manera.
Criterios de aceptación: - Tendrá que existir una actividad para crear historias de usuario, las cuales tendrán automáticamente el estado “Backlog” para su posterior asignación. - Las historias deben contener los siguientes valores como mínimo en su creación: nombre, descripción y número. - Tendrán que existir actividades que permitan añadir los siguientes campos: puntos de historia, criticidad, criterios de aceptación y responsables. - Tendrá que existir una actividad que permita asignar las historias a uno de los <i>sprints</i> del sistema, las cuales quedarán comprometidas y por lo tanto, inalterables. - Tendrá que existir una actividad por sprint que permita asignar a los responsables de cada historia. Estos además podrán modificar el porcentaje de avance de sus historias.

Tabla 3.9: HU 9: creación y edición de historias de usuario.

Historia de Usuario 10
Nombre: Tablón de historias
Descripción: Como alumno quiero disponer de un tablón donde pueda organizar mis historias de usuario para tener control sobre ellas y reflejar la información correctamente.
Criterios de aceptación: - Las historias en el tablero deben mostrar los siguientes valores: Nombre, número, puntos de historia, criticidad, responsables y porcentaje de avance. - Las historias deben ser capaces de moverse en el tablero entre las etapas de “Por Hacer”, “En Progreso”, “Testing” y “Finalizado”. Esto será posible durante los periodos de <i>sprint</i> y solo con las historias que pertenezcan a tal <i>sprint</i>. - Se debe poder acceder al detalle de cada historia desde el tablero. - Los docentes y ayudantes tendrán que tener acceso a este tablero en modo lector.

Tabla 3.10: HU 10: tablón de historias de usuario.

Historia de Usuario 11
Nombre: Evaluación de <i>sprints</i>
Descripción: Como docente y ayudante quiero evaluar las distintas historias de usuario asociadas a un <i>sprint</i> para obtener la nota final del equipo.
Criterios de aceptación: - Al entrar en la evaluación de un <i>sprint</i> se debe listar cada historia de usuario con sus criterios de aceptación desglosados. - Se debe poder asignar un puntaje entre 0 a 4 para cada criterio de aceptación dependiendo de su cumplimiento. - Se debe poder agregar <i>feedback</i> a cada criterio para que se comprendan las razones del puntaje. - Al finalizar la evaluación de un <i>sprint</i>, el sistema debe calcular la nota final para el equipo evaluado haciendo uso de la siguiente ecuación, donde NF es la nota final, N es el número de historias, HU_i es el puntaje total obtenido para la historia i, PHU_i es el número de puntos de historia para la historia i, PHU_T es el número total de puntos de historia del <i>sprint</i>, $\overline{pi_{Profesores}}$ es el puntaje promedio dado por los profesores para la historia i, y $\overline{pi_{Ayudantes}}$ es el puntaje promedio dado por los ayudantes para la historia i: $NF = \left(\frac{\sum_1^N HU_i}{4} \right) \cdot 100$ $HU_i = \frac{PHU_i}{PHU_T} \left(\overline{pi_{Profesores}} \cdot 0,7 + \overline{pi_{Ayudantes}} \cdot 0,3 \right)$ - Cada vez que un profesor o un ayudante ingrese o actualice una evaluación, la nota final debe calcularse nuevamente para mantener una nota real en cada momento. - Los ayudantes y profesores deben poder acceder al detalle de la evaluación de un <i>sprint</i> para cada equipo, donde tendrán que ver el puntaje asignado y los comentarios de cada evaluador. Los alumnos solo podrán ver esta información asociada a su equipo.

Tabla 3.11: HU 11: evaluación de *sprints*.

3.4. Herramientas de Desarrollo

Las herramientas consideradas para el desarrollo de la plataforma son las siguientes:

■ Frontend

- Javascript 18: lenguaje de programación utilizado para aplicaciones web.
- React 17: librería para el desarrollo de interfaces de usuarios. Más información en la **Sección 2.6**
- reduxjs: librería que permite manejar el estado de la aplicación de forma anexa al incorporado por *React*.
- react-redux: librería que permite acoplar *Redux* con *React* de manera sencilla.
- react-router-dom: librería para implementar enrutamiento en aplicaciones *React*.
- Bulma: librería de estilos para aplicaciones web.
- Fullcalendar: librería que incorpora un calendario dinámico y personalizable.
- react-beautiful-dnd: librería que incorpora funcionalidades de *drag and drop* visual.
- Font Awesome: librería para hacer uso de iconos.

■ Backend

- Python 3: lenguaje de programación basado en *scripting* de funcionamiento eficiente, fácil aprendizaje y que ofrece buenas herramientas para el manejo de diversos datos.
- Flask 2: *microframework* para el desarrollo de aplicaciones y servicios web. Más información en la **Sección 2.7**.
- flask-api: librería que permite hacer uso de funcionalidades de *APIs* como el manejo de protocolos *HTTP*.
- Flask-PyMongo: librería que permite conectar *Flask* con bases de datos *MongoDB*.
- werkzeug: librería para encriptar y desencriptar datos.
- pytz: librería que ayuda al manejo de zonas horarias.
- pyjwt: librería para la creación y decodificación de *tokens JWT* de autenticación.
- MongoDB: motor de base de datos NoSQL documental. Más información en la **Sección 2.8.1**.

- Amazon S3: sistema de almacenamiento de datos y archivos basado en la nube. Más información en la **Sección 2.9**.
 - Boto3: librería para la comunicación con *AWS S3*.
- Versionamiento
 - Git: herramienta de control de versiones. Más información en la **Sección 2.5.1**
 - Github: plataforma web que permite alojar repositorios de git de forma remota.

3.5. Modelos de Datos

Para comenzar con el desarrollo de la herramienta a medida, lo primero a tener en consideración es la forma en que se relacionarán los datos para que estos estén en sintonía con lo que se busca desarrollar, además de que puedan ser trabajados de manera óptima. Para esto se construyeron dos modelos: uno conceptual y uno relacional, los cuales pueden verse a continuación.

3.5.1. Modelo conceptual de datos

Se compuso el modelo conceptual presente en la **Figura 3.1** con el fin de plasmar de una forma menos técnica los datos que tendrá la aplicación y de qué forma se relacionarán. Este puede verse en detalle en la sección **Apéndice A.3**

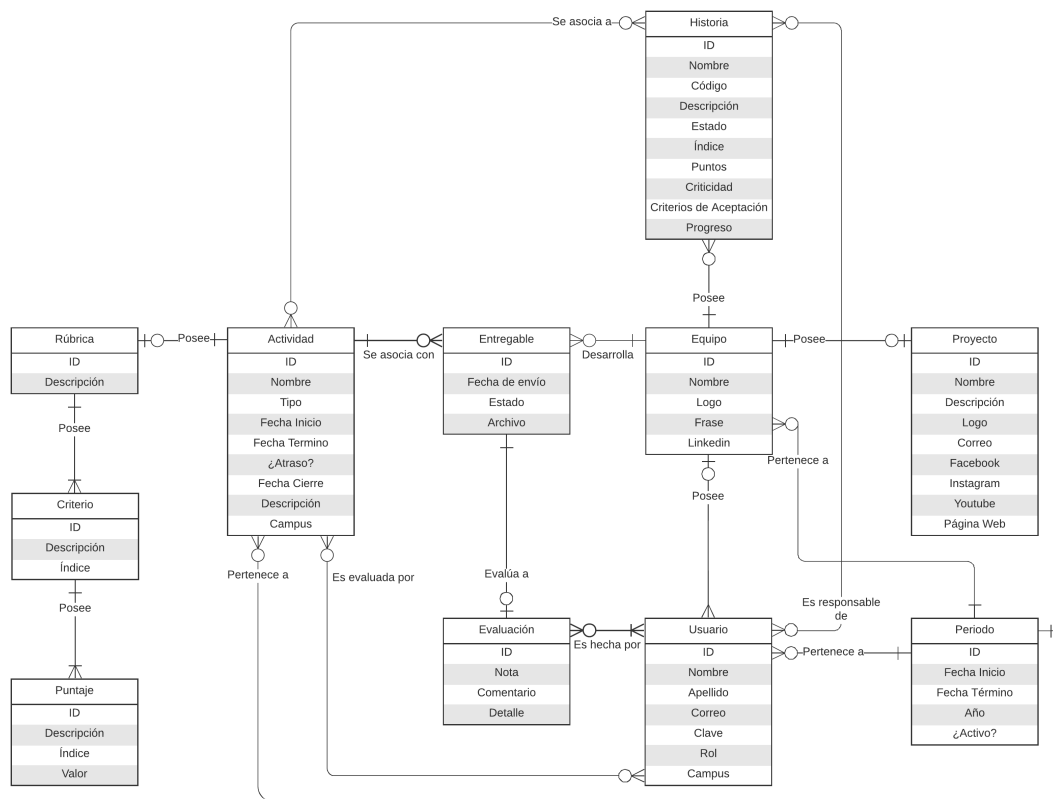


Figura 3.1: Modelo conceptual de datos de la aplicación a desarrollar. Fuente: elaboración propia.

A continuación se presenta el detalle de las distintas tablas de la **Figura 3.1**:

1. **Periodo**: representa los distintos periodos en los cuales se desarrollará el curso, siendo pensado en el plazo de un año lectivo. Se incluye su fecha de inicio y de término como registro histórico, y un campo nombrado **Activo** que indica si el periodo es el vigente o no. Este se relaciona con las tablas **Equipo**, **Usuario** y **Actividad** para indicar a cual periodo pertenece tal información, de modo que no exista duplicidad de los datos y los registros antiguos no interfieran con los nuevos, como por ejemplo que se desarrolle una actividad de años pasados en el presente o que un equipo antiguo desarrolle una actividad actual.
2. **Usuario**: representa a los usuarios del sistema, discriminando su rol con el campo

Rol, el cual puede tener el valor de alumno, ayudante, profesor o juez. Se incluye otra información pertinente al usuario, donde se destacan el **Email** y la **Clave** para ingresar al sistema. Por último, el campo **Campus** busca distinguir a los alumnos entre las sedes San Joaquín y Casa Central, de modo que solo tengan interacción con sus actividades pertinentes.

3. **Equipo**: representa a los equipos del curso, uno de los elementos más importantes de la plataforma. Posee información distintiva de cada equipo y se relaciona con la tabla **Usuario**, ya que hace referencia a sus miembros (usuarios con rol de alumno). Posee relación con las tablas **Entregable** e **Historia**, ya que estas últimas están ligadas directamente con el equipo que las desarrolla.
4. **Proyecto**: representa los proyectos que se estarán desarrollando en el curso, por lo que se listan los distintos datos que éste puede tener (como por ejemplo sus redes sociales). Está relacionado con la tabla **Equipo**, donde cada uno de estos puede tener solo un proyecto en desarrollo.
5. **Actividad**: representa las distintas actividades que se desarrollarán y evaluarán en el curso, por lo que es uno de los elementos esenciales del sistema. Entre sus campos más importantes destacan **Tipo**, el cual representa a qué tipo proceso o entregable está ligada la actividad, siendo las opciones *sprint*, creación de historias, edición de historias, asignación de historias a *sprint*, documento o presentación; **Fecha Inicio**, fecha en la cual comienza el desarrollo de la actividad (y por lo tanto, cuando ya se pueden comenzar a realizar entregables); **Fecha Término**, la cual representa el fin oficial de la actividad (y el término oficial de los entregables); **Atraso**, la cual señala si la actividad permite entregables atrasados (posteriores a la fecha de término) o no; **Fecha Cierre**, que indica el cierre definitivo de la actividad en el caso de permitirse atrasos y por ende, cuando se dejan de recibir entregables; **Descripción**, para detallar a los alumnos qué es lo que tienen que hacer; y finalmente **Campus**, el cual indica la sede en la que la actividad podrá ser desarrollada. Esta entidad está relacionada con la tabla **Usuario**, ya que cada actividad posee evaluadores designados (usuarios con rol de profesor, ayudante o juez) para que puedan calificar los entregables de esta.
6. **Entregable**: representa las entregas que hacen los equipos para responder a una actividad, y es el medio por el cual se evaluará su desempeño en el curso. Posee

el campo **Fecha de envío** como registro histórico; **Estado** para señalar si el entregable está pendiente, entregado con atraso, entregado o evaluado; y **Archivo** para aquellas actividades que requieran que el entregable sea un documento. La relación con la tabla **Actividad** representa como cada entregable es enviado para responder a solo una actividad en particular.

7. **Rúbrica:** representa y unifica la estructura de las rúbricas que tendrá cada actividad, por lo que será la plantilla para realizar las evaluaciones. Esta tabla posee el campo de **Descripción**, el cual será utilizado por los docentes para dar directrices sobre la evaluación. los datos para establecer el **Puntaje** que debe ser asignado en cada **Criterio** dependiendo del **Desempeño** obtenido en los entregables. Cada rúbrica está enlazada a solo una tabla **Actividad** ya que entrega las pautas para su evaluación.
8. **Criterio:** representa cada criterio que estará presente en una rúbrica y que será la guía para determinar la nota final. Esta cuenta con un campo **Descripción**, el cual describe el criterio en cuestión; y con un campo **Índice**, el cual sirve para considerar la posición en que el criterio debe aparecer. Esta tabla pertenece a una **Rúbrica** en específico.
9. **Puntaje:** representa la distinta asignación de puntaje que se le puede dar a un criterio. Posee el campo **Descripción** para describir como se puede obtener tal puntaje, **Índice** para almacenar la posición de este puntaje con respecto a los otros, y **Valor** para señalar cuanto valor tiene este puntaje a nivel numérico. Esta tabla está relacionada directamente con **Criterio**, ya que un criterio está compuesto de un conjunto de puntajes.
10. **Evaluación:** representa la evaluación que se realiza de un entregable en particular haciendo uso de una rúbrica como pauta. Contiene los campos **Nota** para asignar una nota final, **Comentario** para agregar retroalimentación por parte de los evaluadores al equipo, y **Detalle** para mostrar cual fue el desempeño obtenido en cada criterio de la rúbrica asignada. Posee enlace con la tabla **Entregable** ya que evalúa un entregable en particular. Finalmente, posee relación con la tabla **Usuario** para representar a los usuarios que participan en esta evaluación, siendo estos los asignados previamente como evaluadores de la actividad asociada al entregable evaluado.

11. **Historia:** representa las historias de usuario que serán desarrolladas por los equipos a lo largo del curso y culminarán en el término de los proyectos. Entre sus campos se encuentran **Nombre**, que identifica la historia con un nombre representativo; **Código**, que identifica la historia con un número a criterio de los equipos; **Descripción**, que entrega detalles de la historia en formato “como <actor> quiero <necesidad> para <objetivo>”; **Estado**, que indica si una historia está por hacerse, en desarrollo, en *testing* o finalizada; **Índice**, que es un indicador numérico para ordenar las historias en los tableros de la plataforma; **Puntos**, que son los puntos de historia que representan el grado de esfuerzo que esta implica; **Críticidad**, que señala qué tan importante es la historia, pudiendo ser crítica, esencial, deseable y opcional; **Criterios de Aceptación**, los cuales dan pauta para indicar si una historia ha sido completada correctamente; y finalmente **Progreso**, el cual identifica en qué porcentaje de avance se encuentra la historia. Esta entidad se enlaza con la tabla **Usuario**, cuya relación representa a los miembros del equipo que serán responsables de desarrollar la historia. Además, se enlaza con la tabla **Actividad** ya que cada historia puede ser desarrollada en un *sprint* específico.

3.5.2. Modelo relacional de datos

A partir del modelo conceptual presentado anteriormente se decidió crear un modelo relacional de datos, el cual busca plasmar de una manera más técnica los datos señalados y caracterizarlos según su tipo, relación, formato, entre otros. Cabe destacar que al ser *MongoDB* la base de datos con la cual se trabajará en esta plataforma, un diagrama relacional no representará correctamente su modelado ya que obedecen a paradigmas completamente distintos; sin embargo, y considerando que no existe una forma oficial de realizar un diagrama que plasme una base de datos *MongoDB*, se cree que un diagrama relacional puede aproximarse bastante a su estructura y que, en este caso, ayudará enormemente a su legibilidad. Es por todo lo mencionado que algunas tablas no contarán con claves foráneas, sino que con claves que representen a la entidad misma (como se haría en una base de datos *MongoDB*), obteniendo como resultado la **Figura 3.2** (el cual puede verse ampliado en la sección **Apéndice A.4**).

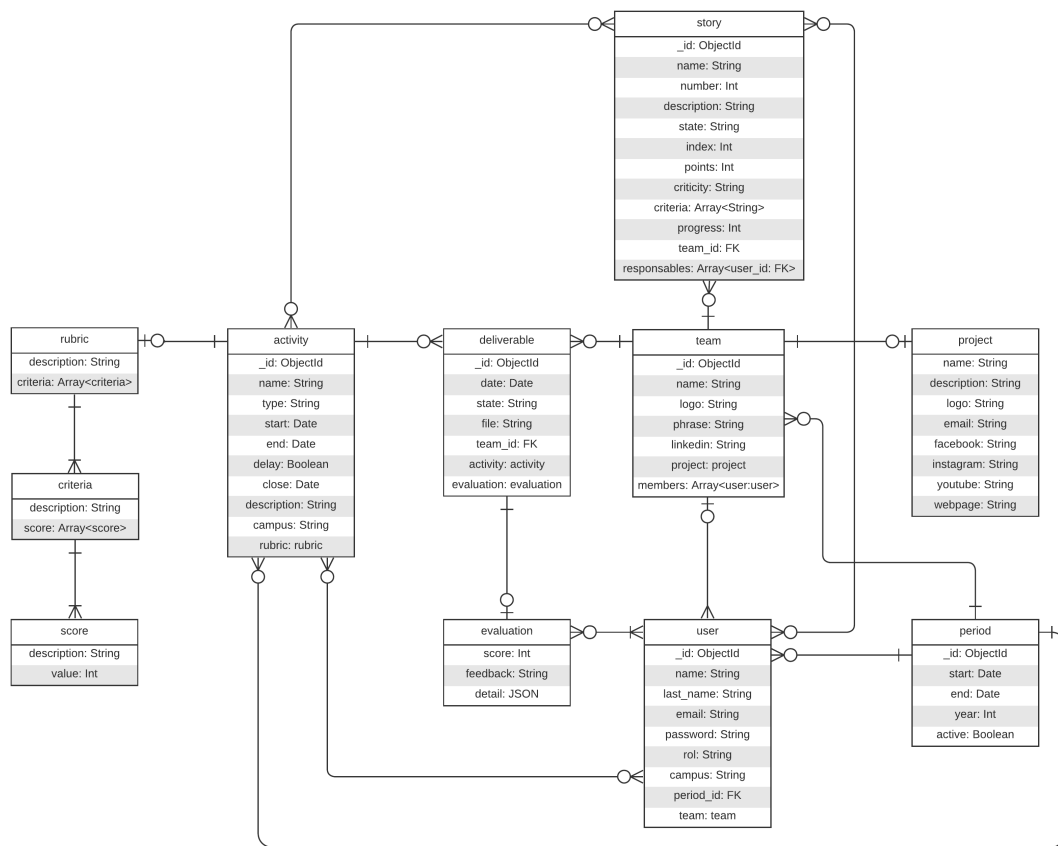


Figura 3.2: Modelo relacional de datos de la aplicación a desarrollar, adaptado para su compatibilidad con *MongoDB*. Fuente: elaboración propia.

Del anterior modelo se deben señalar algunas aclaraciones:

- A pesar de existir una relación entre *story* y *activity*, y sabiendo que las historias de usuario están directamente ligadas a las actividades de tipo *sprint*, no se creó una relación a nivel de base de datos ya que no resultaba ser una operación óptima al volver los documentos más complejos innecesariamente. En su lugar, ambas tablas se relacionan al compartir el campo *sprint* de *state* con el campo *name* de *activity*, pero se está considerando cambiar esta interacción como trabajo futuro para que se plasme mejor el enlace de los datos.
- En la tabla *team*, el campo *members* es un arreglo de *user* que excluye su campo

team, ya que de estar presente se formaría un ciclo infinito al contenerse a sí mismo.

- En la tabla *evaluation*, el campo *detail* solo se señala del tipo *JSON* ya que su estructura es variable y dependerá del tipo de actividad, siendo igual a su rúbrica en caso de ser de un tipo distinto a *sprint*, o de una estructura dinámica para las actividades de tipo *sprint*.
- Las tablas *project*, *evaluation*, *rubric*, *criteria* y *score* no poseen clave primaria ya que no se almacenan en colecciones independientes, sino que existen solo dentro de otra tabla contenedora (como por ejemplo, *project* dentro de *team*). Por ende, no requieren un identificador ya que sus tablas padre poseen uno por sí mismas.

3.6. Arquitectura de la Plataforma

A partir de las historias de usuario definidas anteriormente, del modelo de datos y las tecnologías mencionadas, se diseñó el diagrama de arquitectura que tendrá la aplicación, el cual se puede encontrar en la **Figura 3.3**.

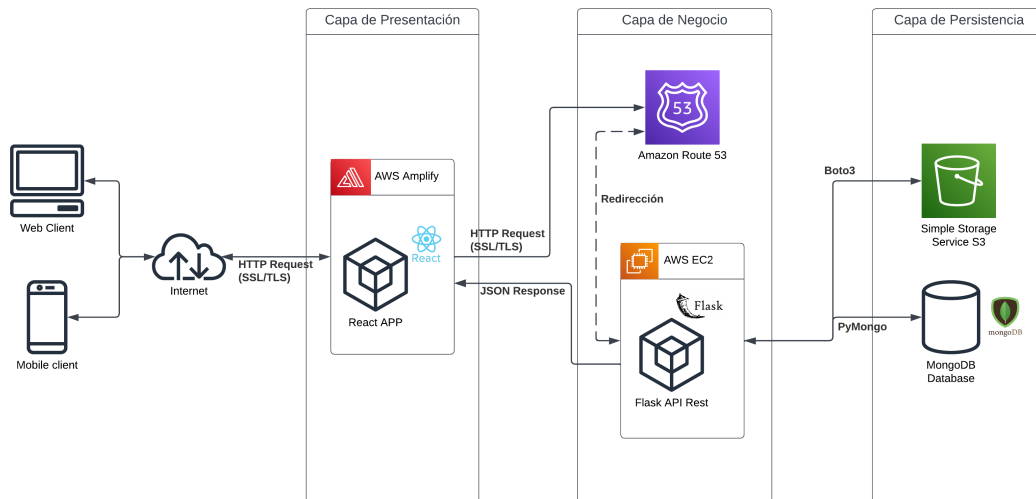


Figura 3.3: Diagrama de arquitectura por capas de la aplicación. Fuente: elaboración propia.

Este diagrama muestra como un cliente (un navegador web de escritorio o *Mobile*) puede acceder a la plataforma mediante internet, llegando así a la interfaz de la aplicación creada con *React* mediante un servicio de *AWS Amplify*. Esta a su vez se comunicaría con la *API Rest* creada con *Flask* y montada en un servicio de *AWS EC2*, a través de una conexión *SSL/TLS* hacia el servicio de dominios entregado por *AWS Route 53*. A la *API* se le haría envío de peticiones *HTTP* y recibiría una respuesta en formato *JSON*. Este servicio se comunicaría tanto con el servidor de *MongoDB* (Atlas) mediante la librería de *PyMongo* y con el servicio de almacenamiento *Amazon S3* a través de la librería *robo3*, los cuales son los encargados de almacenar los datos y archivos respectivamente. A continuación se entregará más detalle de la arquitectura de los distintos elementos desarrollados para la aplicación.

3.6.1. *React APP*

La aplicación *React* desarrollada se encuentra almacenada en un repositorio propio de *Git*, la cual posee una arquitectura de *SPA* (*Single Page App*) caracterizada por ser una aplicación de solo una página. Una vez que el usuario accede a la aplicación no necesita navegar a ninguna otra página web ya que la propia aplicación es lo suficientemente dinámica como para obtener y renderizar pantallas que satisfagan los requisitos de los usuarios en su navegación.

Dado que la aplicación tiene una arquitectura *SPA*, su organización fue hecha de tal manera que las pantallas renderizadas sean un componente de vista (o *View*), en la cual se montan y jerarquizan otros componentes para darle estructura. Para esto se hizo uso de la librería *react-router-dom*, la cual interpreta la dirección web ingresada como una instrucción para montar otra vista en la aplicación y así dar el efecto del cambio de página web. Las rutas y los componentes a los cuales refieren son las siguientes:

- Sin usuario ingresado:
 - `/`: redirige a `/login`.
 - `/login`: muestra *LoginView*, vista de inicio de sesión y registros.
- Con usuario ingresado:
 - `/`: redirige a `/actividades`.

- /actividades: muestra *ActivitiesView*, vista de calendario y listado de actividades.
 - /usuario: muestra *ProfileView*, vista del perfil de usuario.
 - /rubrica/:activityId: muestra *RubricView*, vista de creación y edición de rúbrica para una actividad específica.
 - /evaluacion/:deliverableId: muestra *EvaluateView*, vista de evaluación para un entregable en específico.
 - /evaluacionDetallada/:deliverableId: muestra *DetailedEvaluationView*, vista del detalle de una evaluación para un entregable en específico.
- Con rol de Profesor, Ayudante y Juez:
 - /entregables: muestra *AllDeliverablesView*, vista que reúne todas las actividades y el acceso a sus entregables.
 - /usuarios: muestra *UserListView*, vista que lista a todos los usuarios de la plataforma y permite administrarlos.
 - /entregables/:activityId: muestra *EvaluationsView*, vista que reúne todas los entregables de una actividad en específico y da acceso a su evaluación.
 - /historias/:teamId: muestra *TeamStoriesView*, vista que muestra el tablero de historias de usuario de un equipo específico.
 - /dashboard: muestra *DashboardView*, vista que muestra el *dashboard* de todos los proyectos del periodo en curso.
 - Con rol de Alumno:
 - /historias: muestra *StoriesView*, vista que muestra el tablero de historias de usuario del equipo, permitiendo su creación o edición.
 - /entregables: muestra *DeliverablesView*, vista que muestra los entregables enviados y pendientes de un equipo.
 - /equipo: muestra *TeamView*, vista que permite ver, crear y editar un equipo y su proyecto asociado.

Otro elemento importante de la aplicación *React* es la creación de un conjunto de componentes de formulario para su utilización a lo largo de la plataforma, los cuales

comparten el mismo diseño y comportamiento. Entre todas sus características, a continuación se presentan las más importantes:

- Dentro de un componente *Form* se puede incluir cualquiera de los componentes *Input*, *Select*, *Textarea* o *Checkbox*, todos de elaboración propia y estilizados haciendo uso de la librería de *CSS Bulma*.
- Dentro de un componente *Form* se puede agregar la propiedad *form*, la cual debe recibir un elemento devuelto por un *custom hook* llamado *useForm*. Este *hook* recibe como parámetro un estado con estructura de objeto, cuyas propiedades podrán ser directamente enlazadas con un elemento del formulario.
- Los componentes de un formulario pueden recibir la propiedad *name*, la cual debe ser igual al elemento del estado del formulario a la cual busca referenciar. De esta forma, cualquier interacción con este campo del formulario se verá reflejada en el estado proporcionado.
- Los componentes de un formulario pueden recibir la propiedad *validations*, la cual permite agregar cualquier tipo de validación que se desee para este campo. Estas validaciones pueden ser predefinidas en un archivo llamado *validations* o pueden ser completamente personalizadas por el desarrollador.
- El resultado de un *custom hook* devuelve un estado de validación que muestra en todo momento si los campos del formulario cumplen con sus validaciones. Además, se incluye una función que permite mostrar los mensajes de alerta de cada campo dependiendo de la validación que esté incumpliendo en ese momento.

Otra característica de la aplicación es que sus componentes se encuentran desarrollados haciendo uso de *hooks*, lo cual consta en escribir estos componentes como una función de tipo flecha de *Javascript* (*arrow function*) en lugar de su antigua forma de definición, la cual era orientada a objetos. Por otro lado, se hizo uso de las librerías *redux* y *react-redux* para almacenar un estado global dentro de la aplicación, por lo que se decidió guardar la información del usuario ingresado, las actividades que cambian el comportamiento de la aplicación y el periodo vigente para facilitar el acceso a estas propiedades en todo momento y desde cualquier componente. Por último, se diseñaron un grupo de funciones para que enviaran peticiones a la *API Rest* denominadas servicios (o *services*). Estos se

encargaban de crear la llamada con las propiedades necesarias, tales como el *token* de autorización, los *headers* necesarios, el *body* con la información, *queryparams*, entre otros.

Implementación

Para ejecutar la aplicación de forma local hay que tener instalado *Node* con la versión 18 como mínimo. Luego, para disponer de la aplicación para su uso en la dirección *localhost:3000*, se debe ingresar en una consola, situar en la raíz del proyecto clonado desde *Github* y utilizar los siguientes comandos:

```
npm i
npm start
```

Para el despliegue de la aplicación en un ambiente productivo se hizo uso del servicio *AWS Amplify* (definido en la **Sección 2.10**), al cual se le ingresaron instrucciones para el montaje de la aplicación y se le entregaron los permisos requeridos para esto. Este despliegue puede realizarse en cualquier servicio automatizado de alojamiento de aplicaciones web, siendo estas las mejores opciones al ser escalables, rápidas y seguras, incluyendo funcionalidades como el uso de despliegue automático mediante *CI/CD* (explicado en la **Sección 2.1**). De todas maneras, la aplicación puede ser desplegada en un servicio web convencional haciendo uso de herramientas como *nginx* y construyendo la aplicación manualmente mediante el siguiente comando:

```
npm build
```

3.6.2. *Flask API Rest*

La aplicación *React* desarrollada se encuentra almacenada en un repositorio propio de *Git*, la cual posee una arquitectura orientada al servicio, lo cual se caracteriza por dividir la aplicación en servicios que representan una unidad funcional de negocio cada uno, los que a su vez están débilmente acoplados e interactúan entre sí mediante el paso de mensajes. Cada servicio está dispuesto en una ruta y un controlador en específico, los

cuales hacen uso de operaciones de base de datos (que también se llaman servicios, pero que no se refieren a lo mismo) para responder a la operación solicitada. A continuación se entrega la lista de las rutas definidas en esta aplicación para su uso posterior:

Método	Ruta	Descripción
POST	/activity/create	Crea y devuelve una actividad.
PUT	/activity/edit	Actualiza y devuelve una actividad.
GET	/activity/	Obtiene y devuelve todas las actividades del periodo activo.
GET	/activity/:activityid	Obtiene y devuelve las actividad solicitada.
GET	/activity/getAppActivities	Obtiene y devuelve todas las actividades del periodo activo que pudiesen alterar el funcionamiento de la aplicación.

Tabla 3.12: Listado de rutas ofrecidas por la *API Rest* para la entidad “Actividad”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/deliverable/create/:activityId	Crea y devuelve un entregable asociado a una actividad específica.
GET	/deliverable/getByTeamId/:teamId	Obtiene y devuelve todos los entregables de un equipo específico.
GET	/deliverable/getByActivity/:activityId	Obtiene y devuelve todos los entregables asociados a una actividad específica.
GET	/deliverable/:deliverableId	Obtiene y devuelve un entregable específico.
POST	/deliverable/evaluate/:deliverableId	Asocia una evaluación con un entregable específico y devuelve el entregable evaluado.

Tabla 3.13: Listado de rutas ofrecidas por la *API Rest* para la entidad “Entregable”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/period/create	Crea y devuelve un periodo.
GET	/period/active	Obtiene y devuelve el periodo activo.

Tabla 3.14: Listado de rutas ofrecidas por la *API Rest* para la entidad “Periodo”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/rubric/create/:activityId	Crea una rúbrica para una actividad en específico y devuelve la actividad.
PUT	/rubric/update/:activityId	Actualiza una rúbrica para una actividad en específico y devuelve la actividad.

Tabla 3.15: Listado de rutas ofrecidas por la *API Rest* para la entidad “Rubrica”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/story/create	Crea y devuelve una historia de usuario.
GET	/story/getStoriesBySprint	Obtiene y devuelve el listado de historias asociadas a un sprint. Recibe <i>teamId</i> y <i>sprint</i> como <i>queryparams</i> .
PUT	/story/updateState	Actualiza el campo <i>state</i> de una historia y la devuelve.
PUT	/story/update/:storyId	Actualiza y devuelve una historia de usuario.

Tabla 3.16: Listado de rutas ofrecidas por la *API Rest* para la entidad “Historia de Usuario”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/team/create	Crea y devuelve un equipo.
PUT	/team/update	Actualiza y devuelve un equipo.
GET	/team/dashboard	Obtiene y devuelve el listado de equipos con los porcentajes de avance.
GET	/team/:teamId	Obtiene y devuelve un equipo en específico.

Tabla 3.17: Listado de rutas ofrecidas por la *API Rest* para la entidad “Equipo”. Fuente: elaboración propia.

Método	Ruta	Descripción
POST	/user/register	Registra y devuelve un nuevo usuario con su <i>token</i> .
POST	/user/login	Valida un usuario y devuelve su información con su <i>token</i> .
GET	/user/:id	Obtiene y devuelve un usuario en específico.
GET	/user/all	Obtiene y devuelve un listado de todos los usuarios. Recibe <i>roles</i> , <i>active</i> y <i>campus</i> como <i>queryparams</i> para filtrar usuarios.
PUT	/user/update	Actualiza y devuelve un usuario en específico.

Tabla 3.18: Listado de rutas ofrecidas por la *API Rest* para la entidad “Usuario”. Fuente: elaboración propia.

Implementación

Para ejecutar la aplicación de forma local hay que tener instalado *Python* con la versión 3.9 como mínimo. Luego, para disponer de la aplicación para su uso en la dirección *localhost:5000*, se debe ingresar en una consola, situar en la raíz del proyecto clonado desde *Github* y utilizar los siguientes comandos:

```
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
flask run
```

Cabe destacar que para hacer uso de la aplicación debe crearse un archivo *.env* que posea la siguiente estructura:

```
FLASK_ENV=development
SECRET_KEY={a complex key}
MONGO_URI={your mongo uri}
```

Para el despliegue de la aplicación en un ambiente productivo se hizo uso del servicio *AWS EC2* (definido en la **Sección 2.11**), en el cual se hizo su montaje haciendo uso de *gunicorn* y *nginx* para servir la aplicación, donde además se entregaron los permisos para acceder al servicio desde otras redes *IP*. Para exponer el servicio mediante una red segura *SSL/TLS* se tuvo que asociar un dominio de red, lo cual se realizó con *AWS Route 53* (definido en la **Sección 2.12**, por lo que posteriormente se generó el certificado de seguridad utilizando *letsencrypt*, un servicio de *SSL/TLS* gratuito. Este despliegue puede realizarse en cualquier servicio automatizado de alojamiento de aplicaciones web, siendo estas la mejores opciones al ser escalables, rápidas y seguras, incluyendo funcionalidades como el uso de despliegue automático mediante *CI/CD* (explicado en la **Sección 2.1**). Para este caso se recomendaría el uso de *AWS Elastic Beanstalk* o *AWS Lightsail*. De todas maneras, la aplicación fue desplegada de forma manual en un servicio web convencional en la nube, en este caso, *AWS EC2*.

3.7. Metodología de Trabajo

Al buscar implementar una herramienta a medida resulta difícil tener claridad de todos los requerimientos desde un inicio, ya que a medida que se va explorando en el desarrollo de esta es cuando van surgiendo nuevas ideas, haciendo que el proyecto tome forma. Es por esto que se decidió seguir una metodología de trabajo iterativa incremental, donde se comienza con mínimas características de la plataforma según lo definido en las historias de usuario para, posteriormente, dar paso a una revisión de lo obtenido y discutir posibles mejoras que debiesen ser agregadas en una próxima iteración.

El desarrollo de la aplicación tomó cerca de un año de trabajo constante, tratando de mantener reuniones regulares con el profesor guía una vez al mes donde se dio revisión al desarrollo y se entregaron mejoras para este. Tal forma de trabajo fue posible durante

los primeros 6 meses, donde se desarrolló el tablero de historias de usuario, el registro e ingreso de usuarios, la creación de equipos y proyectos, además de la creación de actividades junto a sus respectivas rúbricas. Sin embargo, y debido a distintos factores que afectaron al desarrollador, no se pudieron sostener las reuniones pactadas ni el ritmo de trabajo llevado, por lo que se continuó el desarrollo en solitario durante los siguientes 6 meses, donde se refinaron los módulos antes mencionados y se añadió el envío de entregables, las evaluaciones en base a rúbricas, el *dashboard* de profesores y las vistas de lectura de los demás módulos. Lo anterior fue expuesto al profesor guía, quien validó el cumplimiento de las historias de usuario, pero que recalcó la falta del módulo de evaluación de *sprints*, el cual fue añadido y validado posteriormente. Por otra parte, al haber hecho uso de *Git*, se tiene registro de las distintas versiones que la aplicación ha tenido en su proceso evolutivo.

3.8. Vistas de la Aplicación

A continuación se presenta una serie de capturas de pantalla obtenidas de la aplicación desarrollada, las cuales exponen los módulos más importantes de esta:

3.8.1. Login

La **Figura 3.4** muestra la vista del *login*, la que permite que los distintos usuarios (sean alumnos, profesores u otros) puedan registrarse e ingresar en la aplicación. Esta puede verse ampliada en la sección **Apéndice A.5**.

3.8.2. Listado de actividades

La **Figura 3.5** muestra la vista de las actividades de la asignatura, las cuales se encuentran dispuestas en el calendario de modo informativo. Además, se puede ver el listado de actividades que aún están dentro del plazo para ser completadas, donde se incluye su descripción, tiempo restante, entre otros. Esta puede verse ampliada en la sección **Apéndice A.6**.

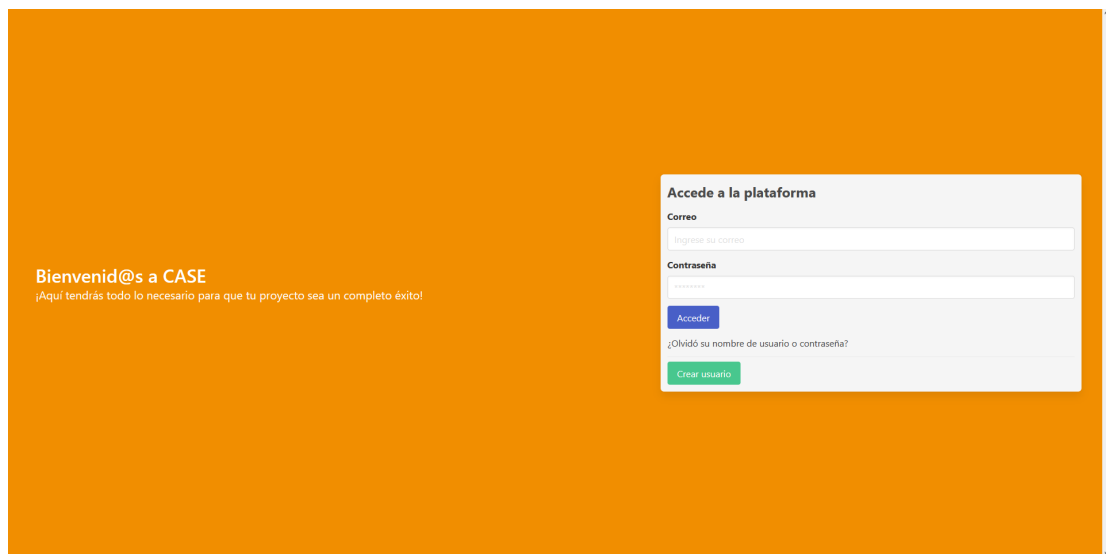


Figura 3.4: Pantalla de *login* de la plataforma. Fuente: elaboración propia.

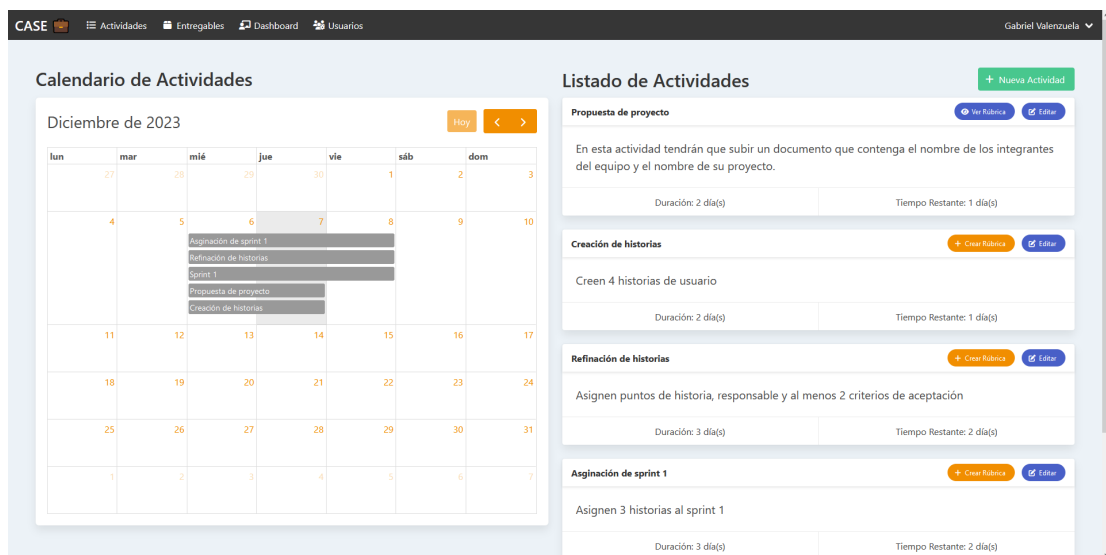


Figura 3.5: Pantalla de actividades de la plataforma. Fuente: elaboración propia.

3.8.3. Entregables

La **Figura 3.6** muestra la vista donde se agrupan los entregables que los alumnos han hecho para una actividad en específico, exponiendo la calificación obtenida. Por otro lado, aquí es donde se pueden realizar evaluaciones y, al mismo tiempo, asignar una rúbrica a la actividad. Esta puede verse ampliada en la sección **Apéndice A.7**.

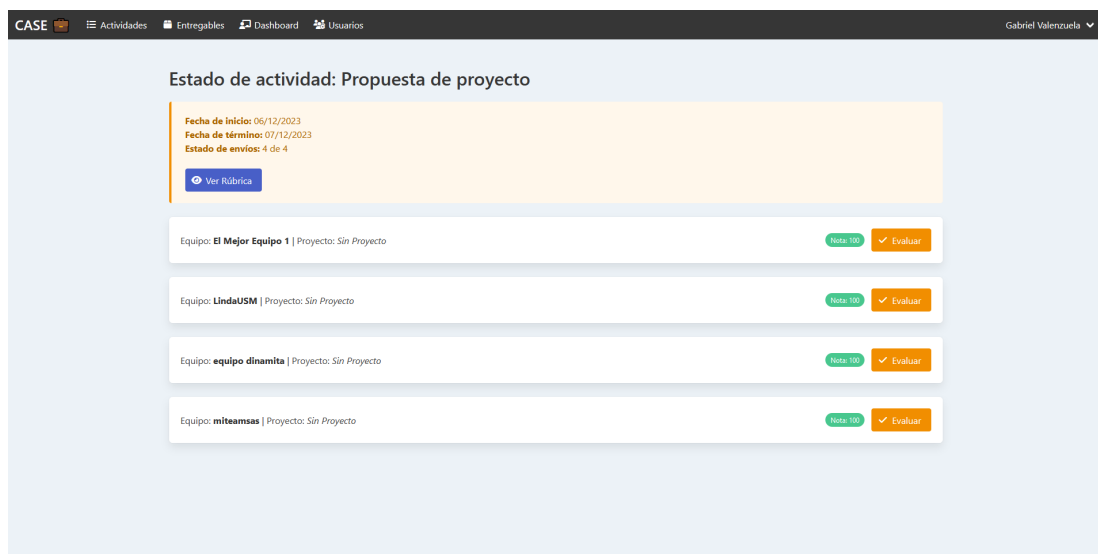


Figura 3.6: Pantalla de entregables de los alumnos en la plataforma. Fuente: elaboración propia.

3.8.4. Rúbricas

La **Figura 3.7** muestra la vista que permite crear rúbricas a las actividades. En esta se definen distintos criterios y, en cada uno, se asigna un puntaje distinto dependiendo del desempeño de los equipos. Esta puede verse ampliada en la sección **Apéndice A.8**.

Figura 3.7: Pantalla de creación de rúbricas de la plataforma. Fuente: elaboración propia.

3.8.5. Evaluación

La **Figura 3.8** muestra la vista que permite realizar evaluaciones de los entregables de los alumnos a partir de la rúbrica asignada a su actividad. Las notas son calculadas a partir del puntaje obtenido en cada criterio, lo cual se realiza tomando como base la rúbrica asignada. Esta puede verse ampliada en la sección **Apéndice A.9**.

3.8.6. Tablero de historias de usuario

La **Figura 3.9** muestra la vista que permite gestionar las historias de usuario de un equipo. En este módulo los alumnos pueden trabajar directamente sobre las historias (como crearlas, editarlas o moverlas de columna), mientras que los profesores pueden ingresar solo para su revisión. Esta puede verse ampliada en la sección **Apéndice A.10**.

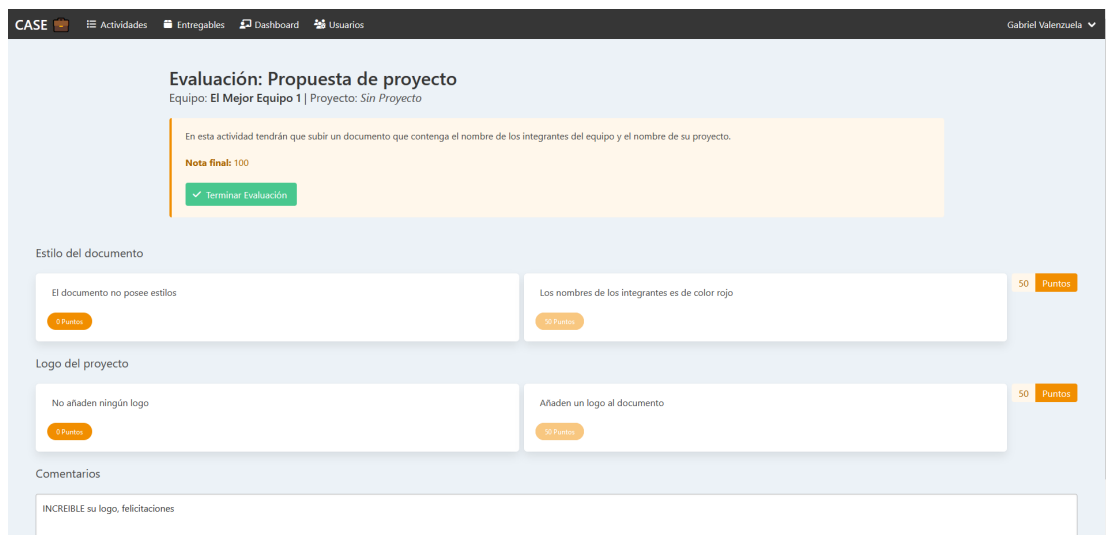


Figura 3.8: Pantalla de evaluación de rúbricas de la plataforma. Fuente: elaboración propia.

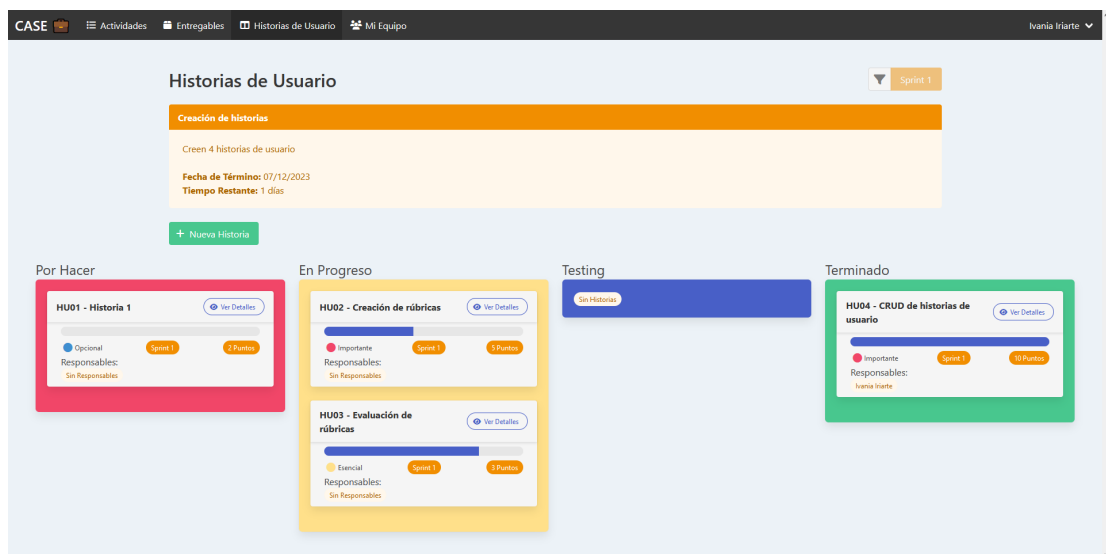


Figura 3.9: Pantalla del tablero de historias de usuario de la plataforma. Fuente: elaboración propia.

3.8.7. Dashboard de equipos

La **Figura 3.10** muestra la vista que permite tener control sobre el estado de avance de los equipos. Aquí se muestra el progreso de cada *sprint* y del proyecto completo, lo cual se calcula de forma ponderada según los puntos y el progreso de cada historia, dependiendo de la fase a la que esté asignada. Esta puede verse ampliada en la sección **Apéndice A.11**.

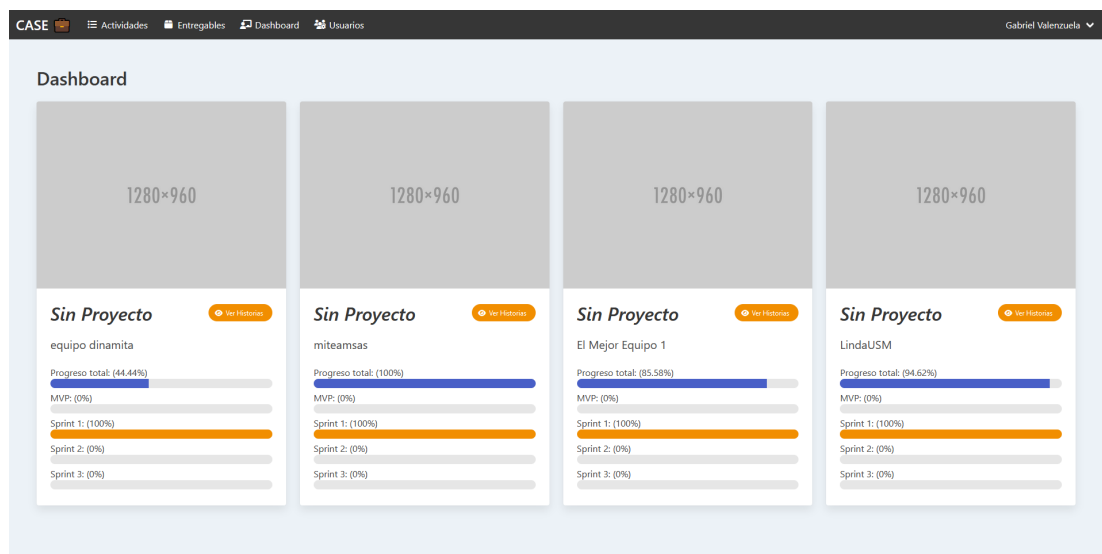


Figura 3.10: Pantalla de progreso de los proyectos o *dashboard*. Fuente: elaboración propia.

3.8.8. Evaluación de *sprints*

La **Figura 3.11** muestra la vista que permite realizar evaluaciones sobre los *sprints* de la asignatura para cada equipo. Estas no hacen uso de una rúbrica definida, sino que siguen un camino aparte donde, para las historias del *sprint* evaluado, se asigna un puntaje a cada criterio de aceptación. Aquí se tiene una gran cantidad de evaluadores, los cuales ingresan su puntaje y retroalimentación en conjunto con los demás para obtener la calificación final, permitiendo que los alumnos puedan ver sus resultados de forma clara. Esta puede verse ampliada en la sección **Apéndice A.12**.

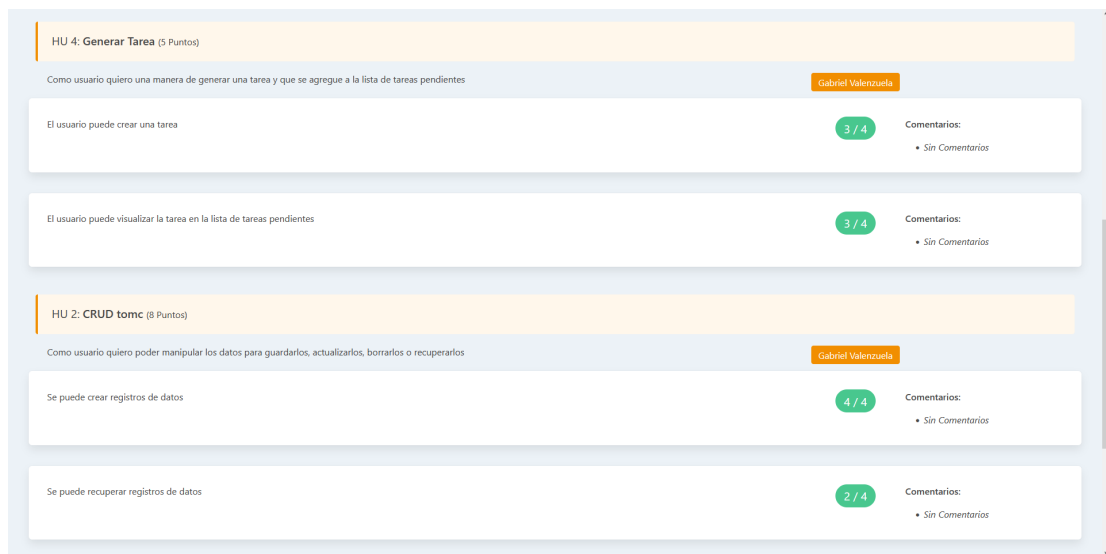


Figura 3.11: Pantalla de evaluación de *sprints*. Fuente: elaboración propia.

3.8.9. Equipo

La **Figura 3.12** muestra la vista que permite crear y editar la información del equipo y el proyecto del alumno. Aquí solo son requeridos los campos básicos para crear estos elementos, de tal forma que no se entorpezca el proceso en fases tempranas de la asignatura y, además, permitiendo refinar esta información más adelante. Esta puede verse ampliada en la sección **Apéndice A.13**.

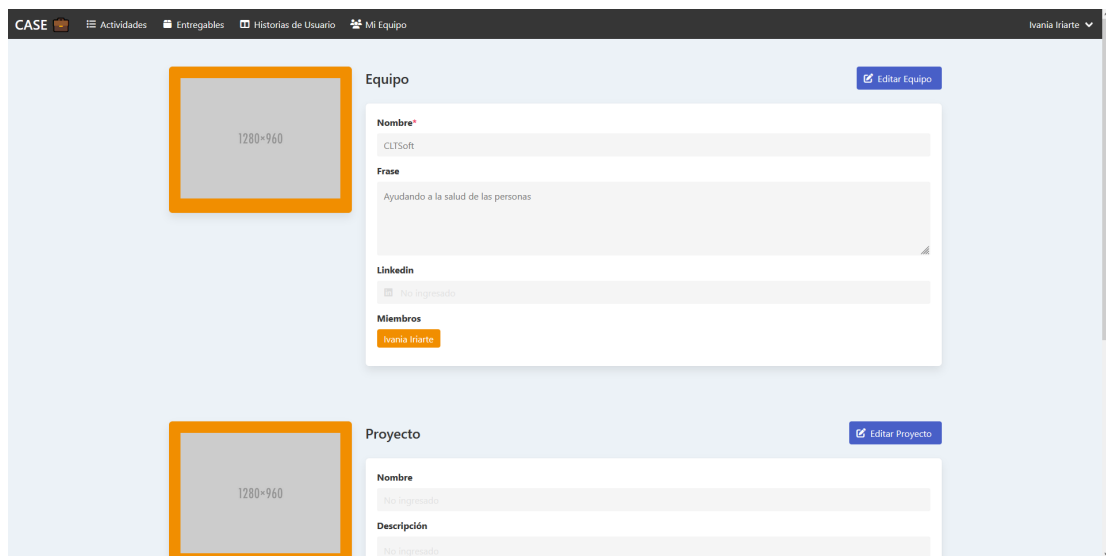


Figura 3.12: Pantalla de equipo y proyecto. Fuente: elaboración propia.

Capítulo 4

Validación de la Solución

Para validar si los objetivos de la aplicación fueron logrados se consideraron tres factores: el cumplimiento de los principios de Nielsen, la validación de los criterios de aceptación por parte de profesores y la opinión de los alumnos que hayan cursado Feria de Software.

4.1. Cumplimiento de los principios de Nielsen

Como se mencionó en la **Sección 2.13**, los principios de Nielsen permiten mejorar la usabilidad y experiencia de usuario en los sitios web, por lo que una validación de la aplicación respecto a estos principios permitiría tener claridad sobre el estado de su interfaz. Es por esto que se detallará a continuación si la herramienta cumple con las distintas heurísticas.

4.1.1. Visibilidad del estado del sistema

La herramienta muestra en todo momento barras de carga cuando se realiza un proceso por detrás, tal como se muestra en la **Figura 4.1**, la cual muestra la carga que existe en el listado de actividades mientras la plataforma carga los datos. Además, existen barras de carga en los botones cuando estos realizan una operación por detrás, tal como se puede ver en la **Figura 4.2**.

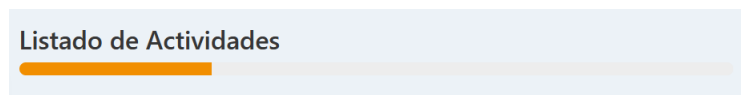


Figura 4.1: Evidencia de barras de carga en la aplicación. Fuente: elaboración propia.

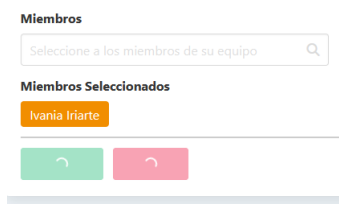


Figura 4.2: Evidencia de barras de carga en los botones de la aplicación. Fuente: elaboración propia.

Además, los botones, formularios y otros elementos se destacan cuando son clicados o cambian de color al pasar el *mouse* por encima, entre otros detalles que también respetan este principio. Esto puede ser apreciado en la **Figura 4.3** donde el borde del campo en el que se está escribiendo se encuentra destacado.

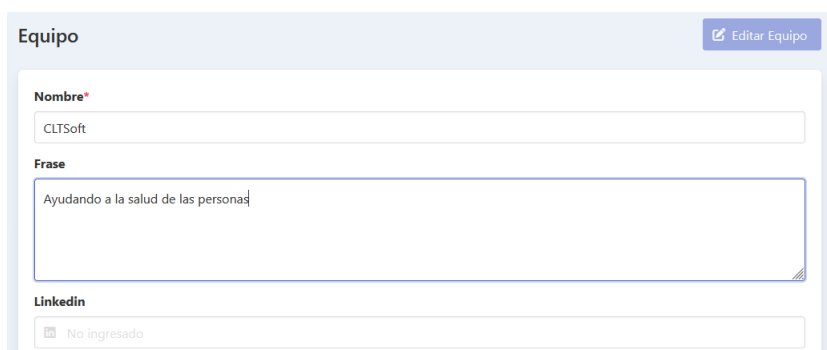
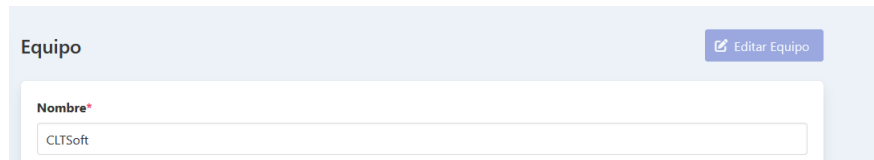


Figura 4.3: Evidencia de destaque de elementos frente a interacción con ellos. Fuente: elaboración propia.

4.1.2. Concordancia entre el mundo real y el sistema

La herramienta no presenta tecnicismos en sus textos, los cuales ya son pocos de por sí. Además, se añaden iconos reconocibles en los botones para representar la acción que

realizan como puede ser apreciado en la **Figura 4.4**, donde se representa un lápiz en un botón de “Editar Equipo”.



Equipo

Editar Equipo

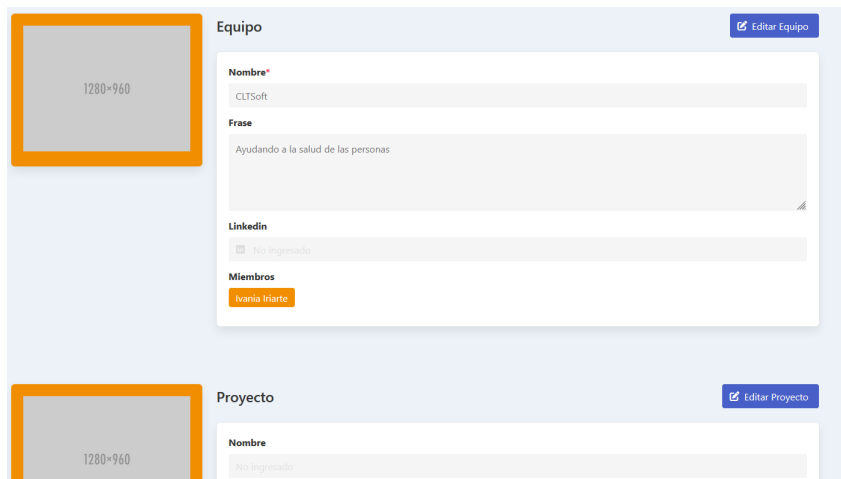
Nombre*

CLTSoft

Figura 4.4: Evidencia del uso de elementos de la vida cotidiana en la aplicación. Fuente: elaboración propia.

4.1.3. Controles de usuario y libertad

La herramienta permite editar la mayoría de la información que los usuarios puedan ingresar (como historias de usuario, actividades, rúbricas, entre otros), lo cual puede ser apreciado en las **Figuras 4.5 y 4.6**, las cuales presentan la opción de editar lo hecho previamente. Sin embargo, hay secciones específicas cuya ejecución no posee la opción de deshacerse, como por ejemplo el envío de un entregable.



Equipo

Editar Equipo

Nombre*

CLTSoft

Frase

Ayudando a la salud de las personas

Linkedin

No ingresado

Miembros

Ivanita Marte

Proyecto

Editar Proyecto

Nombre

No ingresado

Figura 4.5: Evidencia de la opción de editar el equipo y proyecto en la aplicación. Fuente: elaboración propia.

Rúbrica

Propuesta de proyecto

En esta actividad tendrán que subir un documento que contenga el nombre de los integrantes del equipo y el nombre de su proyecto.

✓ Actualizar Rúbrica

Criterio	Puntaje 1	Puntaje 2
Criterio 1: Estilo del documento	Puntaje 1: El documento no posee estilos Puntuación: 0	Puntaje 2: Los nombres de los integrantes es de color rojo Puntuación: 50
Criterio 2: Logo del proyecto	Puntaje 1: No añaden ningún logo Puntuación: 0	Puntaje 2: Añaden un logo al documento Puntuación: 50

+ Añadir Criterio

Figura 4.6: Evidencia de la opción de editar una rúbrica en la aplicación. Fuente: elaboración propia.

4.1.4. Consistencia y estándares

La aplicación es consistente en sus colores, fuentes, iconos, diseño de vistas y diseño de formularios, lo cual puede apreciarse al comparar el formulario de creación de actividades (**Figura 4.7**) con el formulario de creación de historias de usuario (**Figura 4.8**).

Crear Actividad

Tipo Actividad*
 Seleccione una opción ▼

Título Actividad*
 Creación de informes...

Campus*
 Ambos ▼

Fecha de Inicio*
 dd - mm - aaaa 📅

Fecha de Término*
 dd - mm - aaaa 📅

☐ ¿Acepta atrasos?

Descripción Actividad*
 Los alumnos tendrán que crear sus historias de usuario para...

Evaluador(es)
 Seleccione una opción ▼

Sin Evaluadores

Crear **Cancelar**

Figura 4.7: Evidencia del diseño del formulario de creación de actividades. Fuente: elaboración propia.

Crear Historia de Usuario

Número Historia*
 HU 10

Título Historia*
 Creación de usuarios, CRUD profiles, etc...

Descripción Historia*
 Como <sujeeto> quiero <deseo> para <objetivo>...

Crear **Cancelar**

Figura 4.8: Evidencia del diseño del formulario de creación de historias de usuario. Fuente: elaboración propia.

Al mismo tiempo, la aplicación es consistente con elementos externos como la presencia de una barra de navegación superior, la cual se puede apreciar en la **Figura 4.9**.

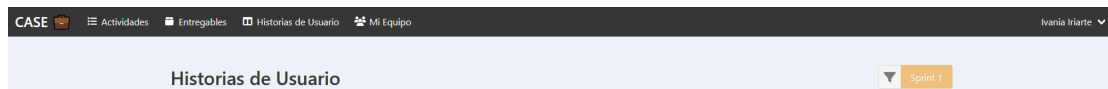


Figura 4.9: Evidencia de la barra de navegación como elemento consistente con las páginas web. Fuente: elaboración propia.

4.1.5. Prevención de errores

La aplicación en su mayoría valida cada dato de los formularios, por lo que ayuda a prevenir que se ingrese información incorrecta o la falta de esta, lo cual se puede apreciar en la **Figura 4.10**, donde se destaca todo elemento faltante o mal ingresado. Sin embargo, hay acciones que se detonan apenas son clicadas y no solicitan una validación extra, permitiendo así que pudiesen ocurrir ciertos errores.

Figura 4.10: Evidencia de la validación de un formulario de la aplicación. Fuente: elaboración propia.

4.1.6. Reconocer es mejor que recordar

La aplicación presenta una serie de iconos que son ampliamente utilizados por aplicaciones populares, y que por ende, son fácilmente reconocibles. Además, la información está dispuesta para que sea sencilla de reconocer haciendo uso de colores distintivos, además de estar ordenada por prioridad para el usuario (como las actividades, que se ordenan por su fecha de término). Todo lo anterior es fácilmente apreciable en la **Figura 4.11**.

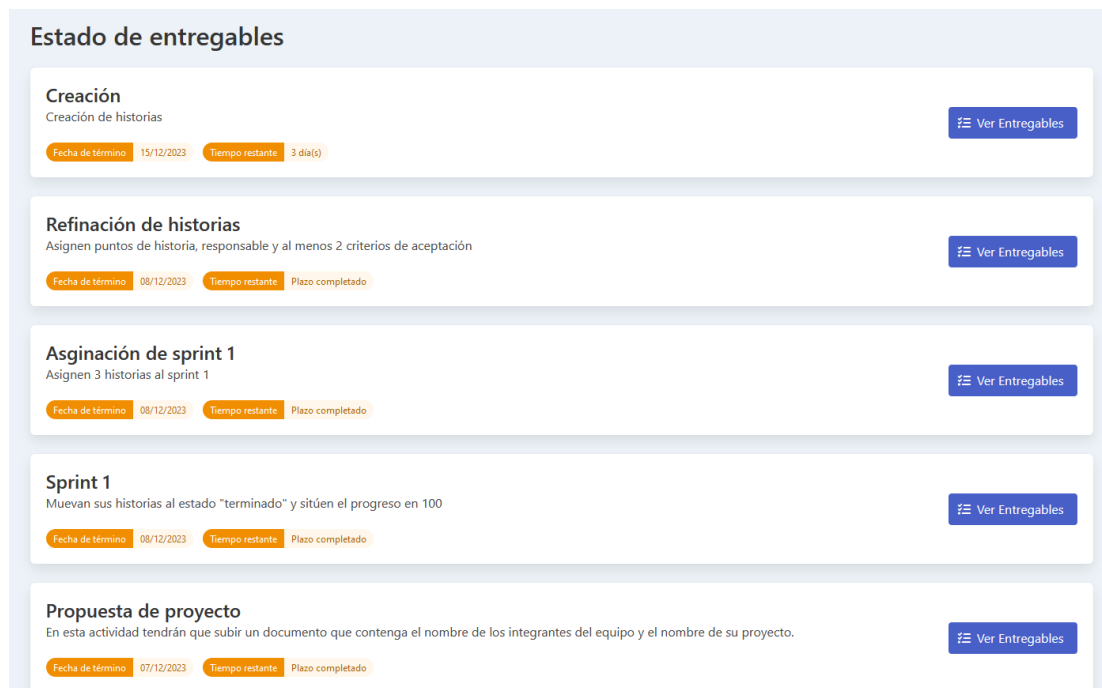


Figura 4.11: Evidencia del orden de información por fecha de término, distinción de información con colores más fuertes y el uso de íconos reconocidos por otras aplicaciones. Fuente: elaboración propia.

4.1.7. Flexibilidad y eficiencia de uso

La aplicación presenta varias características que permiten agilizar su uso, como el uso de la tabulación para desplazarse por los campos de un formulario, el guardado de la sesión para ingresar directamente a la aplicación sin tener que volver a iniciar sesión, o el uso de la tecla *Enter* al final de un formulario para hacer el envío de la información;

sin embargo, estas interacciones no son evidenciables con una imagen. Por otro lado, y entendiendo que el público objetivo de la herramienta estará inmerso en la asignatura, se podría asumir que la herramienta no requiere un gran enfoque en la flexibilidad. A pesar de esto, la herramienta fue desarrollada teniendo en consideración el trabajo colaborativo, permitiendo que los distintos miembros de un equipo (o los profesores con sus actividades) puedan interactuar con los elementos de la aplicación de forma independiente, permitiendo a usuarios menos experimentados descansar en aquellos con mayor dominio.

4.1.8. Estética y diseño minimalista

La aplicación trata, en su mayoría, de mostrar solo información relevante para los usuarios. Se hace uso de una paleta de colores reducida para no sobrecargar la visión, los textos son lo más resumido posible y, por último, se hace uso de un menor espacio de la pantalla para desplegar la información. Todo lo anterior se puede apreciar en una de las vistas con más información de la aplicación, el tablón de historias de usuario (**Figura 4.12**), en donde solo se muestra información estrictamente relevante como la actividad actual que permite la creación de historias de usuario, la que cuenta solo con los datos esenciales. La paleta de colores varía entre 5 tonos y variaciones de estos, los cuales son necesarios para distinguir las columnas del tablero. En las tarjetas de historia también se muestra solo la información necesaria, dando la opción de entrar a los detalles de cada una.

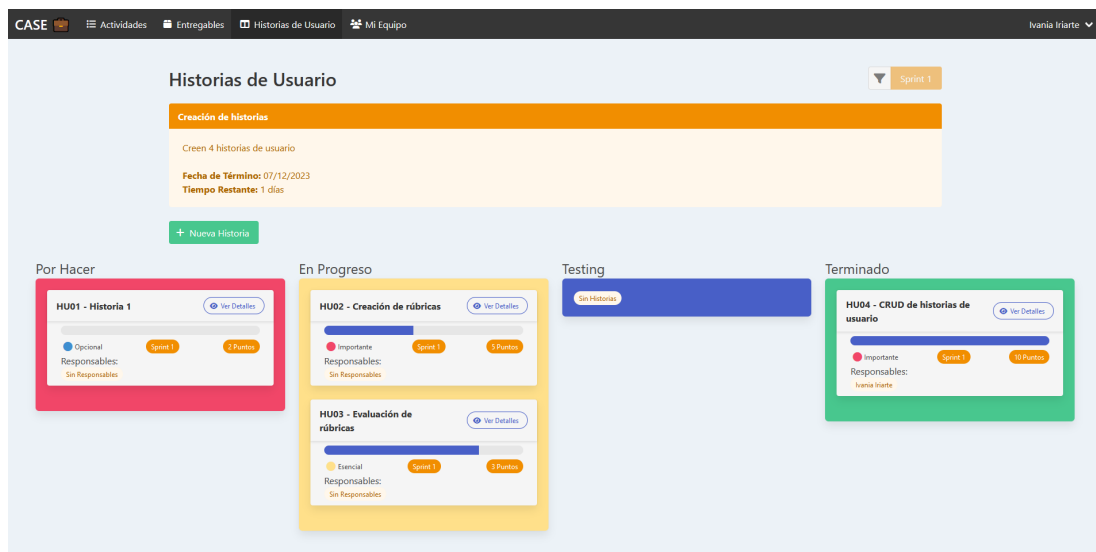


Figura 4.12: Vista del tablero de historias de usuario como evidencia de estética y diseño minimalista. Fuente: elaboración propia.

4.1.9. Ayudar a los usuarios a reconocer, diagnosticar y solucionar errores

La aplicación no presenta con claridad la mayoría de errores que pudiesen ocurrir ya que intenta recuperarse de forma autónoma de gran parte de ellos. Sin embargo, hay ciertos errores que no proponen una solución clara o que no son manejados completamente, los cuales han sido descubiertos en etapas posteriores del desarrollo de la aplicación. No existe evidencia visual para plasmar lo mencionado anteriormente, ya que en su mayoría son procesos que ocurren por detrás.

4.1.10. Ayuda y documentación

Dentro de la misma aplicación no hay guías que permitan comprender el funcionamiento de ciertas secciones más complejas, y que en ocasiones podría resultar necesaria. Para esto se propone la creación de un manual como trabajo futuro.

4.2. Pruebas de Profesores

En un principio se deseaba que la gran mayoría (si no su totalidad) de los profesores encargados de las asignaturas de la Feria de Software pudiesen realizar pruebas con la herramienta y validar sus criterios de aceptación, sin embargo esto no pudo ser posible por distintos factores. Es por esto que la validación fue realizada por el profesor guía de este documento, quien pudo probar u observar todas las funcionalidades de la aplicación y señalar las historias de usuario que fueron completadas, como se muestra en la **Tabla 4.1**.

HU	Estado	Comentarios
1 (Tabla 3.1)	Completada	Debiese desplegarse más información del equipo y el proyecto, o entregar esa información en alguna vista aparte para tener acceso a esta.
2 (Tabla 3.2)	Completada	Hay que añadir la opción de eliminar actividades.
3 (Tabla 3.3)	Completada	Falta la recuperación de contraseña. Sería deseable poder cargar a los alumnos con un Excel.
4 (Tabla 3.4)	Completada	Sería deseable poder permitir a ciertos equipos poder editar sus entregar en casos excepcionales.
5 (Tabla 3.5)	Completada	<i>Sin comentarios.</i>
6 (Tabla 3.6)	Completada	Hay que añadir casillas para ingresar puntos extra y puntos de descuento. Se mantiene el deseo de descargar las evaluaciones en un Excel.
7 (Tabla 3.7)	Completada	<i>Sin comentarios.</i>
8 (Tabla 3.8)	Completada	<i>Sin comentarios.</i>
9 (Tabla 3.9)	Completada	Hay que añadir la opción de eliminar historias.
10 (Tabla 3.10)	Completada	<i>Sin comentarios.</i>
11 (Tabla 3.11)	Completada	<i>Sin comentarios.</i>

Tabla 4.1: Completitud de Historias de Usuario según profesor guía. Fuente: elaboración propia.

Los comentarios entregados por el profesor serán propuestos como trabajo futuro.

4.3. Pruebas de Alumnos

Para realizar las pruebas con alumnos se consideró un recorrido para que puedan hacer uso de la aplicación y una encuesta para que entreguen su opinión según lo visto.

4.3.1. Recorrido en la aplicación

Se simulará un equipo de desarrollo conformado por 4 alumnos, siendo uno de ellos el jefe de proyecto. Cada uno tendrá que registrarse en la plataforma con su información respectiva, haciendo hincapié en que ingresen el mismo campus (para el caso de esta prueba, San Joaquín). Una vez dentro de la plataforma, el jefe de proyecto tendrá que crear un equipo y añadir a los miembros como parte de este, para luego dar detalles sobre algún proyecto ficticio que tendrán que desarrollar. A continuación, tendrán que ingresar a la sección de “Actividades” para que puedan revisar su calendario académico (el cual será previamente poblado con actividades de prueba). Luego, tendrán que ingresar a la sección de “Entregables” para que puedan hacer envío de un entregable al cual se le debe adjuntar un documento; para esto, tendrán que revisar la rúbrica del entregable y armar el documento en base a sus criterios, concluyendo en el envío de este a la plataforma. En ese momento, el encargado de la prueba tendrá que evaluar rápidamente este entregable, dando puntaje y comentarios, para luego pedir a los alumnos que vuelvan a ingresar a la sección de “Entregables” y que puedan ver el detalle de su evaluación.

Concluido lo anterior, los alumnos tendrán que ingresar a la sección de “Historias de Usuario” e intentar crear una historia, lo cual no se podrá al no existir una actividad para esto. Posteriormente, el encargado tendrá que crear una actividad del tipo “Creación de Historias” para que los alumnos puedan crear cinco historias de usuario, donde además tendrán que editar los datos de una para probar tal funcionalidad. Se solicitará que traten de modificar los criterios de aceptación y responsables de una historia, de asignarla a un *sprint* y de moverla a otra sección del tablero; todas estas funcionalidades estarán deshabilitadas. Gradualmente, el encargado tendrá que crear las actividades respectivas para permitir que los alumnos tengan estas interacciones, volviendo a solicitar su intervención. Una vez que ya se encuentren asignadas 3 historias al *sprint* 1 con sus criterios y responsables definidos, se solicitará, tanto a los responsables como a otros miembros, que muevan 2 de estas historias al estado “Terminado” y que modifiquen su

progreso para, posteriormente, dirigirse a la sección “Entregables” y dar por terminado el *sprint*. Finalmente, el encargado tendrá que evaluar sus historias rápidamente para que los alumnos puedan ver el detalle de su evaluación en la misma sección de “Entregables”.

4.3.2. Encuesta

Para plasmar la opinión de los alumnos respecto a la herramienta y así, conseguir ver si los objetivos propuestos en un principio fueron logrados, se decidió crear una encuesta con las siguientes preguntas:

- Preguntas de valoración, donde las respuestas son “Muy en desacuerdo”, “En desacuerdo”, “Ni de acuerdo ni en desacuerdo”, “De acuerdo” y “Muy de acuerdo”. Las siguientes preguntas tienen como enunciado: **si hubieras contado con esta herramienta cuando rendiste la asignatura...**
 1. ¿Crees que habría facilitado el desarrollo de esta, como por ejemplo, la gestión del proyecto o de los entregables y evaluaciones?
 2. ¿Crees que habría ayudado a tener más en consideración los plazos de las distintas actividades de la asignatura?
 3. ¿Crees que habría mejorado tu desempeño como estudiante?
 4. ¿Crees que habría logrado que el resultado final de tu proyecto tuviese mayor calidad?
 5. ¿Crees que habría facilitado el acceso al detalle de las evaluaciones y a las rúbricas de los entregables?
 6. ¿Crees que habría ayudado a evitar problemas en el desarrollo del proyecto o, en un peor caso, el fracaso de este?
 7. ¿Sientes que el hecho de que los docentes tengan más información acerca del desempeño de tu equipo habría ayudado a que el resultado final de tu proyecto tuviese mayor calidad?
- Preguntas de valoración. Las siguientes preguntas hacen referencia a la aplicación probada.
 8. ¿Crees que la herramienta es fácil de usar?

9. ¿Crees que la herramienta es útil?
 10. ¿Crees que la herramienta funciona de manera eficiente?
 11. ¿Te gustaría que esta herramienta fuese implementada en la asignatura?
- Preguntas de desarrollo, donde se pide al alumno que ingrese directamente su respuesta.
 12. ¿Qué es lo que te gustó de la herramienta?
 13. ¿Qué es lo que no te gustó de la herramienta?
 14. ¿Qué crees que podría ser mejorado de la herramienta?
 15. ¿Qué crees que hace falta en la herramienta?

4.3.3. Resultados de la prueba

Una vez realizada la prueba, en donde fueron encuestados 16 alumnos que reunidos en grupos de 4 personas, se pudo obtener la siguiente información:

- Como caracterización de los alumnos, se desprende que un 68.8 % de ellos rindieron la asignatura en el año 2021, un 18.8 % en el año 2022 y un 12.5 % el año 2020. En total, todos los alumnos han rendido la asignatura en los últimos 4 años. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.14**.
- El 81.3 % de los alumnos está muy de acuerdo sobre que la herramienta habría facilitado el desarrollo de la asignatura, mientras que un 18.8 % está solo de acuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.15**.
- Un 68 % de los alumnos está muy de acuerdo respecto a que la aplicación habría ayudado a tener mayor consideración sobre los plazos y fechas de las actividades de la asignatura, mientras que un 25 % solo está de acuerdo. Un 6.3 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.16**.
- Un 81.3 % de los alumnos está muy de acuerdo sobre que la herramienta habría mejorado su desempeño como estudiante. En contraste, un 6.3 % está en desacuerdo

y un 12.5 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.17**.

- Un 37.5 % de los alumnos está muy de acuerdo y de acuerdo sobre que esta herramienta hubiese ayudado a que su proyecto tuviese mayor calidad. Por otro lado, un 6.3 % no está de acuerdo y un 18.8 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.18**.
- Un 93.8 % de los alumnos está muy de acuerdo respecto a que la herramienta habría facilitado el acceso a las rúbricas y las evaluaciones, mientras que un 6.3 % solo está de acuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.19**.
- Un 50 % de los alumnos está muy de acuerdo sobre que la aplicación habría ayudado a evitar problemas en el desarrollo del proyecto, mientras que 31.3 % considera solo estar de acuerdo. Un 18.8 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.20**.
- Un 62.5 % de los alumnos está muy de acuerdo sobre que el resultado de su proyecto hubiese tenido mayor calidad si los profesores hubiesen tenido más información de su desempeño. Un 18.8 % considera estar de acuerdo, un 6.3 % en desacuerdo y un 12.5 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.21**.
- Un 50 % de los alumnos está muy de acuerdo sobre la facilidad de uso de la aplicación, mientras que un 43.8 % solo está de acuerdo y un 6.3 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.22**.
- Un 93.8 % de los alumnos está muy de acuerdo sobre la utilidad de la aplicación, mientras que un 6.3 % considera solo estar de acuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.23**.
- Un 68 % de los alumnos está muy de acuerdo sobre la eficiencia de la aplicación, mientras que un 18.8 % solo está de acuerdo, un 6.3 % está en desacuerdo y un 6.3 % no está ni de acuerdo ni en desacuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.24**.

- Un 93.8 % de los alumnos está muy de acuerdo en implementar la herramienta en la asignatura, mientras que un 6.3 % solo está de acuerdo. Esto se aprecia de mejor manera en el gráfico de la sección **Apéndice A.25**.
- Entre los elementos que le gustaron a los alumnos se puede nombrar el diseño, funcionamiento y simpleza de la interfaz; el acceso directo a las secciones más importantes de la asignatura, la buena organización de las fechas de las actividades, la claridad sobre las rúbricas y evaluaciones de los entregables, la facilidad en la administración de las historias de usuario y la centralización de la información de la asignatura.
- Entre los elementos que no le gustaron a los alumnos se recibieron varias opiniones que señalan no tener ningún problema con la aplicación. Por otro lado, se nombra el hecho de tener que recargar la página para ver reflejados los cambios que sus compañeros implementaron, el no poder editar entregables una vez enviados, algunos colores que pueden incomodar, que algunas características puedan pasar desapercibidas por su minimalismo y que los entregables con distinto estado no se puedan distinguir fácilmente.
- Entre las cosas que podrían ser mejoradas, la mayoría de los comentarios aluden a mejoras de la interfaz para que sea más intuitiva, como añadir más colores en secciones que quizás resultan muy planas a la vista, añadir un *slider* a la asignación de progreso de las historias o el cambio de nombre de ciertas secciones. Además, se señala que sería útil una mayor guía en la aplicación para comprender mejor su funcionamiento, o el acceso directo a ciertos elementos desde otras vistas.
- Con respecto a lo que hace falta en la aplicación, varias opiniones indican que no hay funcionalidades que hagan falta. Por otro lado, algunos opinan que la aplicación podría tener notificaciones sobre el pronto fin de una actividad, la asignación de una historia u otras alertas importantes; que los profesores puedan cambiar plazos para casos excepcionales, el poder registrar información de reuniones y jornadas de trabajo, que se puedan subir recursos a las actividades e historias de usuario y una mayor guía en la aplicación.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

El desarrollo de proyectos es un proceso bastante complejo, pero que siempre busca llegar al éxito de la mejor manera posible. Hay muchos factores necesarios para que este objetivo sea logrado, y el saber gestionarlos es vital para que su desarrollo no sea un completo desastre. Esto puede verse día a día en la industria, con proyectos reales, donde muchas veces los problemas provocan retrasos en el tiempo o excesos del presupuesto; y a pesar de que muchas veces estos problemas no pueden ser evitados, si pueden ser amortiguados aplicando una correcta estrategia.

A lo largo de este documento se ha plasmado la importancia que una buena gestión de proyectos puede tener para conseguir el completo éxito de estos, y como el uso de herramientas modernas pueden ayudar en este proceso. Es en este contexto donde la necesidad de una herramienta de gestión se vuelve vital a la hora de tener control sobre varios proyectos simultáneamente como lo es el caso de la Feria de Software, asignatura y evento en el cual decenas de proyectos buscan ser completados y exhibidos por los alumnos, y que llevan a estos últimos a tener un enfrentamiento con la realidad en el desarrollo de proyectos de *software*. Viendo esta problemática, se ha decidido que una modernización en los procesos de la asignatura era algo necesario para mejorar la calidad del trabajo tanto de alumnos como de profesores.

Según se investigó, existen varios problemas en el presente a la hora de desarrollar y gestionar los proyectos de la asignatura, entre los cuales destacan la poca trazabilidad que

se tiene de los equipos, la tendencia a la desorganización debido a la gran cantidad de entregables que se van generando, la dispersión de la información por el gran número de plataformas a los cuales los alumnos y profesores deben atender para llevar la asignatura y la poca visibilidad que los docentes tienen de los proyectos, provocando que los problemas que puedan ir surgiendo en el desarrollo no sean detectados a tiempo para darles solución.

Para los problemas nombrados se barajaron distintas soluciones que podrían mejorar notablemente el proceso de gestionar los proyectos, donde finalmente se decidió optar por el desarrollo de una herramienta a medida que unifique la mayoría de elementos de la asignatura y que permita a los docentes tener control sobre el estado de los proyectos, generar rúbricas y evaluaciones de los entregables, calendarizar las actividades de la asignatura y visualizar las historias de usuario creadas por los alumnos; siendo estos últimos capaces de gestionar su equipo, trabajar sobre sus historias de usuario, enviar sus entregables y poder visualizar las rúbricas y evaluaciones de estos. Con este objetivo, se redactaron los requerimientos de la aplicación teniendo en consideración tanto a profesores, alumnos, ayudantes y jueces. Para su desarrollo se seleccionaron tecnologías modernas y sostenibles en el tiempo, de modo que pueda ser iterada constantemente y mejorada según las necesidades que pudiesen ir surgiendo.

La implementación de la aplicación fue un proceso extenso y dificultoso, ya que su complejidad era bastante alta y el trabajo en solitario provoca aún más problemas a la hora de tomar decisiones de diseño y desarrollo. En un principio, se tenía la intención de crear una herramienta con aún más funcionalidades, pero que con el paso del tiempo se volvió evidente que esto era inviable, así que los requerimientos a abarcar fueron reducidos para conseguir un producto sólido dentro del tiempo establecido. A pesar de todo lo anterior, se consiguió una herramienta eficiente con todos los requerimientos que fueron listados, permitiendo así evaluarla y probarla con alumnos que ya hubiesen dado la asignatura y que por ende, comprendería la utilidad que esta aplicación podría tener.

Luego de que un grupo de alumnos hiciera uso de la aplicación, se obtuvieron varios datos sobre su opinión respecto a su utilidad, eficacia, entre otros. A partir de estos resultados se determinó que los alumnos consideran que esta herramienta es necesaria para la asignatura, y que su implementación permitiría mejorar su desempeño, mejorar la calidad de los proyectos, facilitar el desarrollo de los procesos académicos y tener mayor consideración de los plazos, rúbricas y evaluaciones que pudiesen existir. Además, los

alumnos opinan que la aplicación es bastante competente tanto en su usabilidad, utilidad y eficiencia. A pesar de que presentaron bastantes detalles a mejorar, en su mayoría destacaron los buenos elementos que tiene la aplicación, por lo que nuevas iteraciones podrían mejorarla y establecerla como un estándar dentro de ambas asignaturas.

5.2. Cumplimiento de objetivos

A continuación se evaluará si los objetivos propuestos en la **Sección 1.3** fueron cumplidos, lo que permitirá tener un mejor panorama sobre el desempeño obtenido. Para esto se evaluarán los objetivos específicos, donde su cumplimiento denotará la realización del objetivo general.

1. Capturar y estudiar los requerimientos esenciales, en conjunto con los docentes, para cubrir las problemáticas presentadas.

En este documento fueron listados los requerimientos necesarios en forma de historias de usuario para el desarrollo de la plataforma web, lo cual fue refinado en conjunto con los profesores.

2. Crear una plataforma web con los requerimientos detectados, haciendo uso de los estándares vigentes en la producción de herramientas de gestión, que permita mejorar el desarrollo, seguimiento y evaluación de los proyectos candidatos a participar en la Feria de Software.

La herramienta fue desarrollada haciendo uso de tecnologías de vanguardia, tomando como referencia otras herramientas existentes en el mercado con funcionalidades de gran utilidad para la gestión de proyectos.

3. Validar la efectividad de la herramienta implementada a través de su uso por parte de los docentes y mediante su aplicación en ciertos equipos de las asignaturas.

La herramienta fue validada por el profesor guía, quien confirma la completitud de las historias de usuario propuestas y asegura su utilidad en las asignaturas. Por otro lado la aplicación fue validada por alumnos, lo cual conllevó a valoraciones sumamente positivas sobre su funcionamiento.

Una vez detallado el cumplimiento de estos objetivos, se puede inferir que el objetivo ge-

neral fue logrado: **Modernizar los procesos de gestión asociados a las asignaturas de Feria de Software creando una plataforma web, apoyándose en la transformación digital, que automatice el proceso de gestión, seguimiento y evaluación de cada equipo participante de estas asignaturas, de tal manera que su desarrollo sea más eficaz y eficiente.**

5.3. Trabajo futuro

A partir de lo dicho anteriormente, se puede establecer que hay mucho trabajo futuro por delante. La herramienta fue desarrollada teniendo en consideración que pudiese seguir siendo iterada constantemente según los requerimientos que fueran surgiendo a lo largo de su aplicación en la asignatura.

En el corto plazo, lo esencial sería aplicar las mejoras mencionadas por los alumnos durante su uso, ya que son la prueba más fidedigna de lo que es necesario en la plataforma. Entre estas se enumeran mejoras visuales en las secciones de mayor complejidad, añadir mayor guía para los usuarios sobre sus funcionalidades, permitir revertir cambios que pudieron realizarse por accidente y prevenir errores mediante validaciones sobre acciones que sean irreversibles. Además, se tendría que considerar el despliegue de la plataforma en servidores administrados por el Departamento de Informática para que tengan dominio sobre su funcionalidad. Considerando lo anterior, y una vez realizados estos cambios, se tendría que implementar la aplicación de forma gradual en la asignatura, partiendo con unos pocos equipos en un principio y aumentando la cantidad a medida que pase el tiempo, de tal forma que sea fácil manejar los errores y los detalles que se puedan ir presentando durante su uso. Por último, es indispensable que en el corto plazo se implemente la descarga de las evaluaciones en un documento Excel y el desarrollo de un manual de uso para que la aplicación pueda ser utilizada.

En el mediano plazo se tendrían que añadir funcionalidades que fueron descartadas por el plazo del desarrollo, pero que se consideran sumamente valiosas para la herramienta (además de los comentarios del profesor guía al evaluar la herramienta). Entre estas se puede nombrar mayor acceso a la información de equipos y proyectos, la opción de

eliminar actividades e historias de usuario, la opción de recuperar contraseña, la posibilidad de cargar los alumnos con un documento Excel, permitir a ciertos equipos poder editar sus entregar en casos excepcionales con permiso de los profesores, añadir casillas para ingresar puntos extra y puntos de descuento en las evaluaciones, el versionamiento de los distintos recursos de la plataforma, ya sean actividades, rúbricas, evaluaciones o historias de usuario; la posibilidad de desarrollar los documentos directamente en la plataforma, sin la necesidad de tener que subir documentos, lo cual permitiría trabajar de forma conjunta en su desarrollo; la auto-generación de rúbricas basadas en los elementos del formulario de una actividad, de tal forma que agilice los tiempos de entrega de las rúbricas; y, finalmente, la integración de integración continua y despliegue continuo para agilizar los procesos de entrega de *software*.

En el largo plazo sería interesante hacer que la aplicación sea más personalizable, de modo que pueda ser utilizada en otros ámbitos académicos como otras asignaturas o en otras universidades. Además, podría haber un equipo enfocado en el desarrollo de nuevas funcionalidades y mantención de la herramienta, permitiendo que el proceso evolutivo de la transformación digital sea cumplido.

Referencias

- [1] Amazon Web Services. ¿qué es la integración continua? Recuperado el 11 de Mayo de 2021 de <https://aws.amazon.com/es/devops/continuous-integration>.
- [2] Amazon Web Services. ¿qué es la entrega continua? Recuperado el 11 de Mayo de 2021 de <https://aws.amazon.com/es/devops/continuous-delivery>.
- [3] Rafael Angel Calvo. Propuesta basada en métodos ágiles para desarrollo de proyectos capstone de emprendimiento e innovación. *Memoria de titulación*, 2019.
- [4] Red Hat. ¿qué es la transformación digital? Recuperado el 06 de Mayo de 2021 de <https://www.redhat.com/es/topics/digital-transformation>.
- [5] Harvard Business Review. Replanteamiento de la transformación digital: los nuevos datos analizan el cambio imprescindible en la cultura y en los procesos en 2020. *Harvard Business Review - Analytic Services*, 2020.
- [6] Sarah Laoyan. Agile manifesto: la guía para entender la metodología agile. Recuperado el 12 de Diciembre de 2023 de <https://asana.com/es/resources/agile-methodology>, Agosto 2022.
- [7] AWS. ¿en qué consiste scrum? Recuperado el 12 de Diciembre de 2023 de <https://aws.amazon.com/es/what-is/scrum>.
- [8] Encarna Abellán. Herramientas de gestión de proyectos: ¿qué características deben cumplir? Recuperado el 07 de Mayo de 2021 de <https://www.wearemarketing.com/es/blog/herramientas-de-gestion-de-proyectos-que-caracteristicas-deben-cumplir.html>, Noviembre 2019.
- [9] Ben Aston. Las 15 mejores herramientas de gestión de proyectos de 2021: Una reseña experta. Recuperado el 07 de Mayo de 2021 de <https://www.projectmanagement.com>.

`//thedigitalprojectmanager.com/es/las-mejores-herramientas-de-\`
`software-de-gestion-de-proyectos`, Enero 2021.

- [10] Ana Isabel Sordo. 21 herramientas de gestión de proyectos destacadas para 2021. Recuperado el 07 de Mayo de 2021 de `https://blog.hubspot.es/marketing/herramientas-gestion-proyectos`, Noviembre 2020.
- [11] Pablo García. 5 herramientas para la gestión de proyectos. Recuperado el 07 de Mayo de 2021 de `https://www.infoautonomos.com/blog/5-herramientas-gestion-de-proyectos`, Mayo 2017.
- [12] Paola Pérez Sirvent. Herramientas de gestión de proyectos: Los 5 mejores softwares para tu negocio. Recuperado el 07 de Mayo de 2021 de `https://www.entrepreneur.com/article/352679`, Julio 2020.
- [13] Click Age. Descubre cuáles son las mejores herramientas online para la gestión de proyectos. Recuperado el 07 de Mayo de 2021 de `https://clickage.es/negocio/descubre-cuales-son-las-mejores-\herramientas-online-para-la-gestion-de-proyectos`, Diciembre 2018.
- [14] Carlos Sánchez. Ranking de las mejores herramientas de gestión de proyectos. Recuperado el 07 de Mayo de 2021 de `https://www.unir.net/empresa/revista/ranking-de-las-mejores-herramientas-de-gestion-de-proyectos`, Julio 2019.
- [15] Dropbox. Control efectivo de versiones de documentos. Recuperado el 10 de Mayo de 2021 de `https://experience.dropbox.com/es-la/resources/effective-document-version-control`.
- [16] Martín Arévalo. Google drive, dropbox y microsoft onedrive, pros y contras de cada una. Recuperado el 10 de Mayo de 2021 de `https://sevilla.abc.es/tecnologia/moviles/sevi-google-drive-dropbox-\y-microsoft-onedrive-pros-y-contras-cada-202003042116_noticia.html`, Enero 2021.
- [17] Anna Martí. Google drive, dropbox y microsoft onedrive: comparativa de precios y características. Recuperado el 10 de Mayo de 2021 de `https://www.xataka.com/otros/google-drive-dropbox-microsoft-\onedrive-comparativa-precios-caracteristicas`, Abril 2019.

- [18] Git. Acerca del control de versiones. Recuperado el 11 de Mayo de 2021 de <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>.
- [19] React. La biblioteca para interfaces de usuario web y nativas. Recuperado el 29 de Noviembre de 2023 de <https://es.react.dev>.
- [20] Mohit Joshi. Angular vs react vs vue: Core differences. Recuperado el 29 de Noviembre de 2023 de <https://www.browserstack.com/guide/angular-vs-react-vs-vue>, Mayo 2023.
- [21] José Domingo Muñoz. Qué es flask. Recuperado el 29 de Noviembre de 2023 de <https://openwebinars.net/blog/que-es-flask>, Noviembre 2017.
- [22] Ahmed Bahgat. Qué es flask. Recuperado el 29 de Noviembre de 2023 de <https://kinsta.com/es/blog/flask-vs-django>, Agosto 2023.
- [23] Oracle. ¿qué es nosql? Recuperado el 29 de Noviembre de 2023 de <https://www.oracle.com/cl/database/nosql/what-is-nosql>.
- [24] Stack Overflow. 2023 developer survey. Recuperado el 29 de Noviembre de 2023 de <https://survey.stackoverflow.co/2023/#most-popular-technologies-database>.
- [25] MongoDB. ¿qué es mongodb? Recuperado el 29 de Noviembre de 2023 de <https://www.mongodb.com/es/what-is-mongodb>.
- [26] MongoDB. Mongodb atlas. Recuperado el 29 de Noviembre de 2023 de <https://www.mongodb.com/es/atlas>.
- [27] Amazon. Amazon s3. Recuperado el 29 de Noviembre de 2023 de <https://aws.amazon.com/es/s3>.
- [28] Amazon. Aws amplify. Recuperado el 12 de Diciembre de 2023 de <https://aws.amazon.com/es/amplify/faqs>.
- [29] Amazon. Aws ec2. Recuperado el 12 de Diciembre de 2023 de https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html.

- [30] Amazon. Aws route 53. Recuperado el 12 de Diciembre de 2023 de <https://aws.amazon.com/es/route53>.
- [31] Aprenderuxui. Recuperado el 29 de Noviembre de 2023 de <https://aprenderuxui.com/heuristicas-10-principios-de-nielsen/aprender/uxdesign>, Septiembre 2023.

Apéndice A

Diagramas

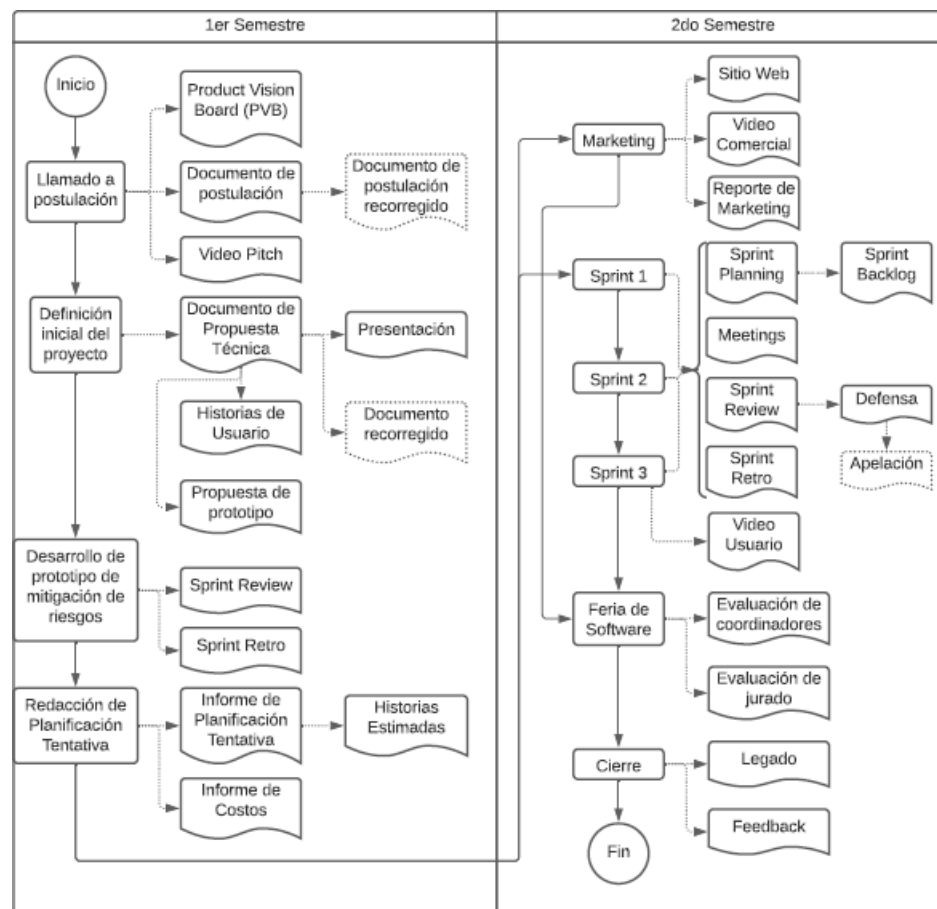


Figura A.1: Diagrama del Flujo de Feria [3]. En esta se marca el flujo de los procesos, los entregables resultantes del proceso y los entregables opcionales (en línea punteada).

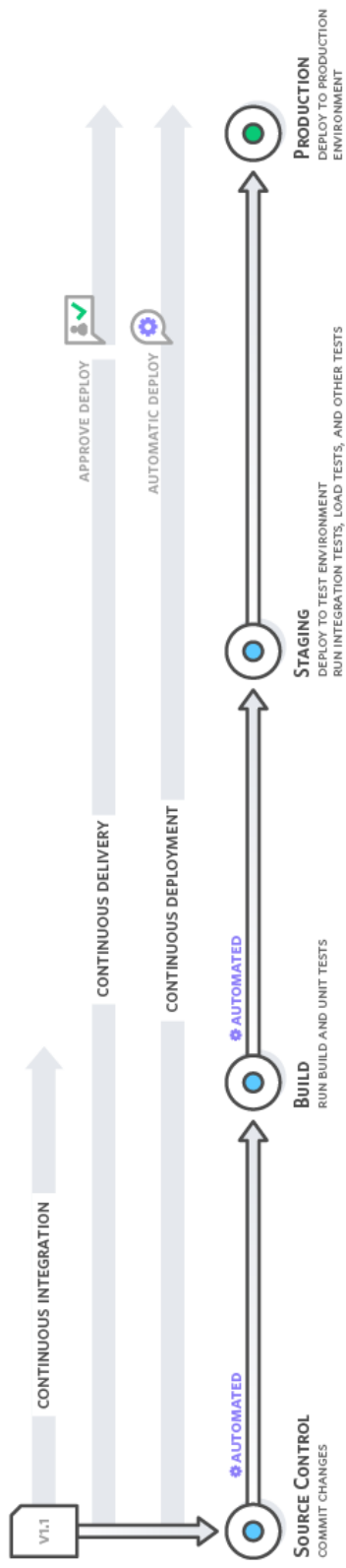


Figura A.2: Diagrama ampliado del proceso de CI/CD de un software en desarrollo [1][2].

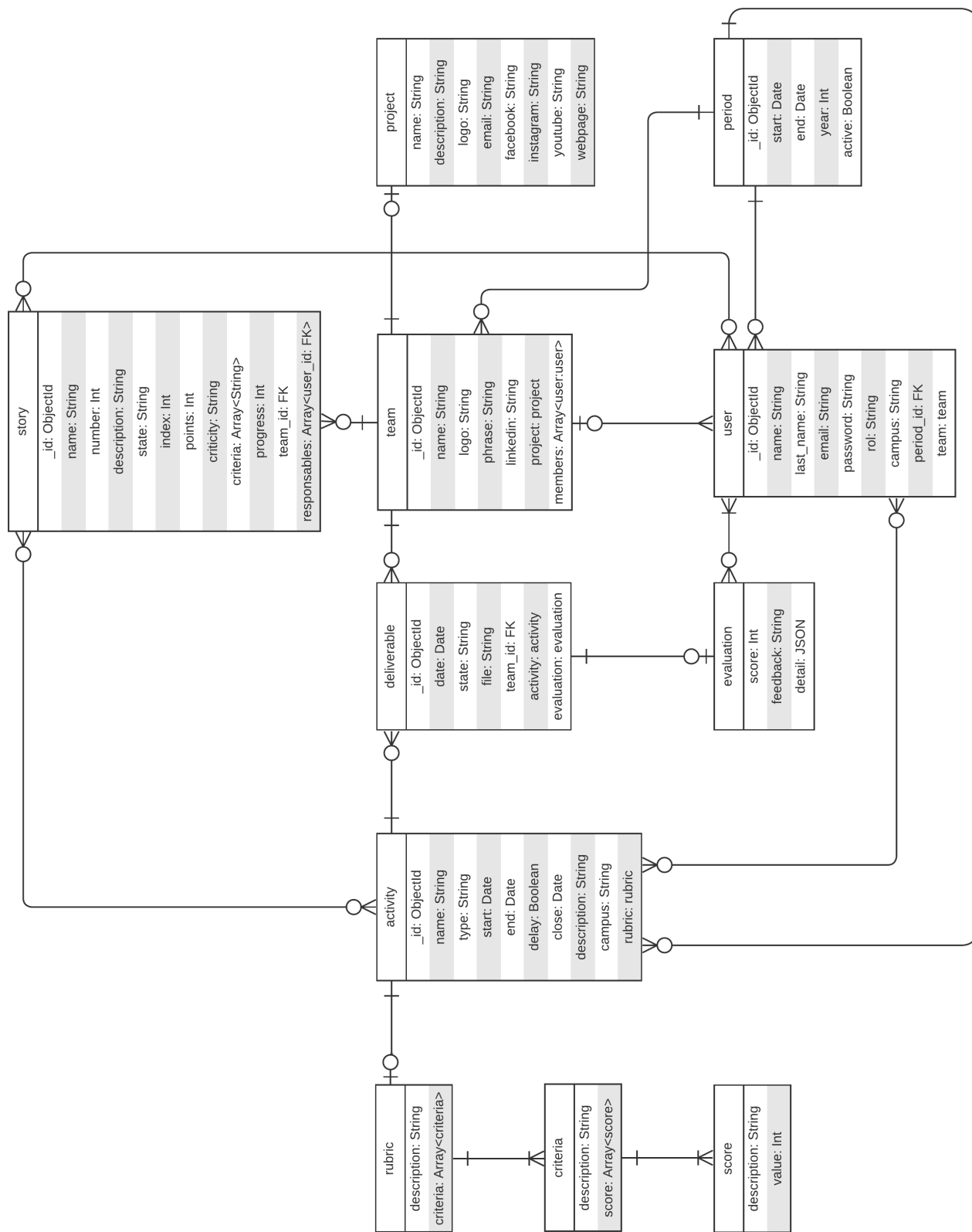
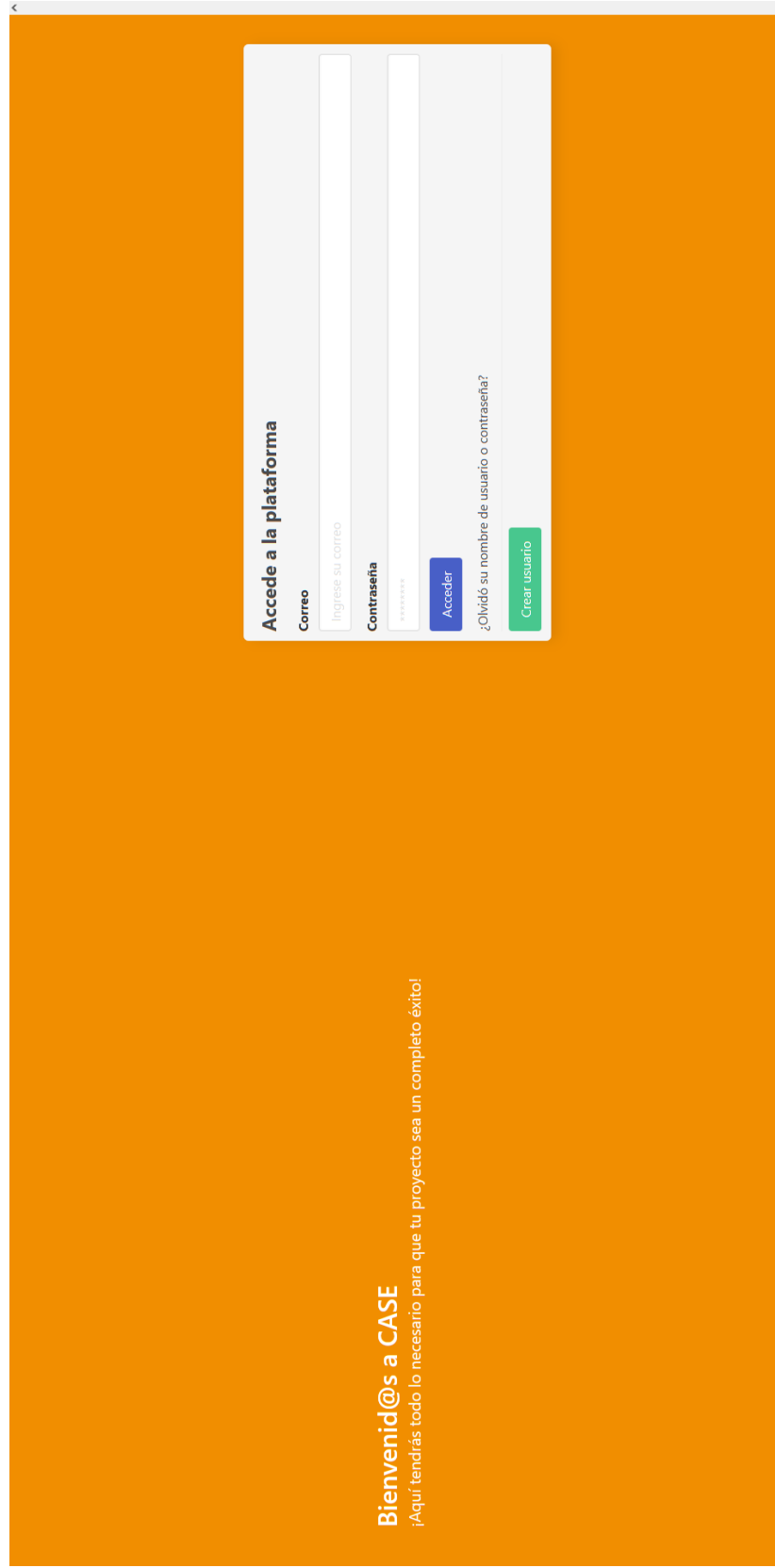


Figura A.4: Modelo relacional de datos de la aplicación adaptado a *MongoDB*, ampliado. Fuente: elaboración propia.



Bienvenid@s a CASE
¡Aquí tendrás todo lo necesario para que tu proyecto sea un completo éxito!

Accede a la plataforma

Correo

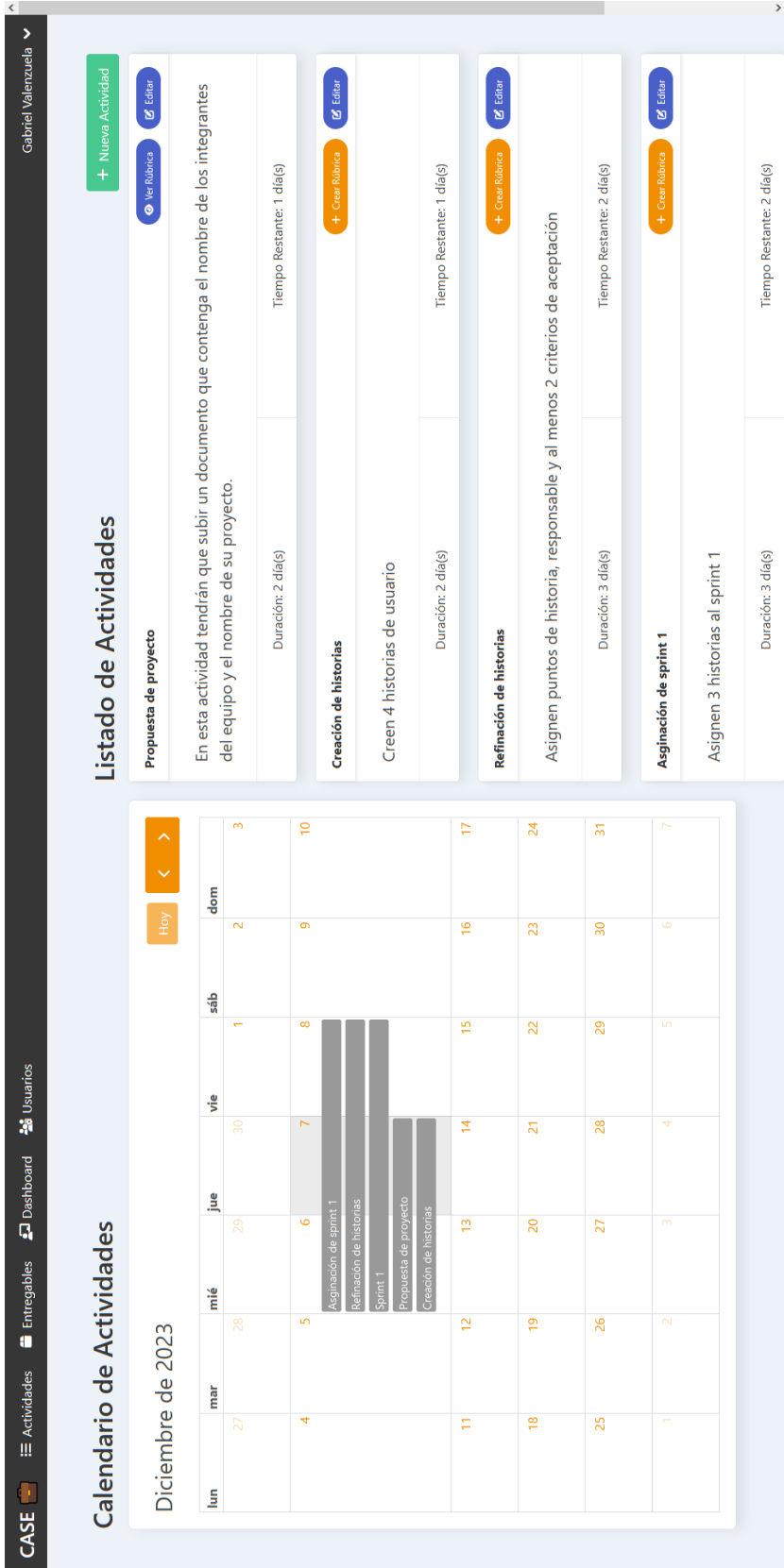
Contraseña

Acceder

¿Olvidó su nombre de usuario o contraseña?

Crear usuario

Figura A.5: Pantalla de *login* de la plataforma expandida. Fuente: elaboración propia.



CASE

Actividades

Entregables

Dashboard

Usuarios

Gabriel Valenzuela

Estado de actividad: Propuesta de proyecto

Fecha de inicio: 06/12/2023
Fecha de término: 07/12/2023
Estado de envíos: 4 de 4

Ver Rúbrica

Equipo: **El Mejor Equipo 1** | Proyecto: Sin Proyecto

Nota: 100

✓ Evaluar

Equipo: **LindaUSM** | Proyecto: Sin Proyecto

Nota: 100

✓ Evaluar

Equipo: **equipo dinamita** | Proyecto: Sin Proyecto

Nota: 100

✓ Evaluar

Equipo: **miteamsas** | Proyecto: Sin Proyecto

Nota: 100

✓ Evaluar

Figura A.7: Pantalla de entregables de los alumnos en la plataforma expandida. Fuente: elaboración propia.

CASE

ActividadesEntregablesDashboardUsuarios

Gabriel Valenzuela

Rúbrica

Propuesta de proyecto

En esta actividad tendrán que subir un documento que contenga el nombre de los integrantes del equipo y el nombre de su proyecto.

Actualizar Rúbrica

Criterio 1: Estilo del documento	Puntaje 1: El documento no posee estilos Puntuación: 0	Puntaje 2: Los nombres de los integrantes es de color rojo Puntuación: 50
Criterio 2: Logo del proyecto	Puntaje 1: No añaden ningún logo Puntuación: 0	Puntaje 2: Añaden un logo al documento Puntuación: 50

Añadir Criterio

Figura A.8: Pantalla de creación de rúbricas de la plataforma expandida. Fuente: elaboración propia.

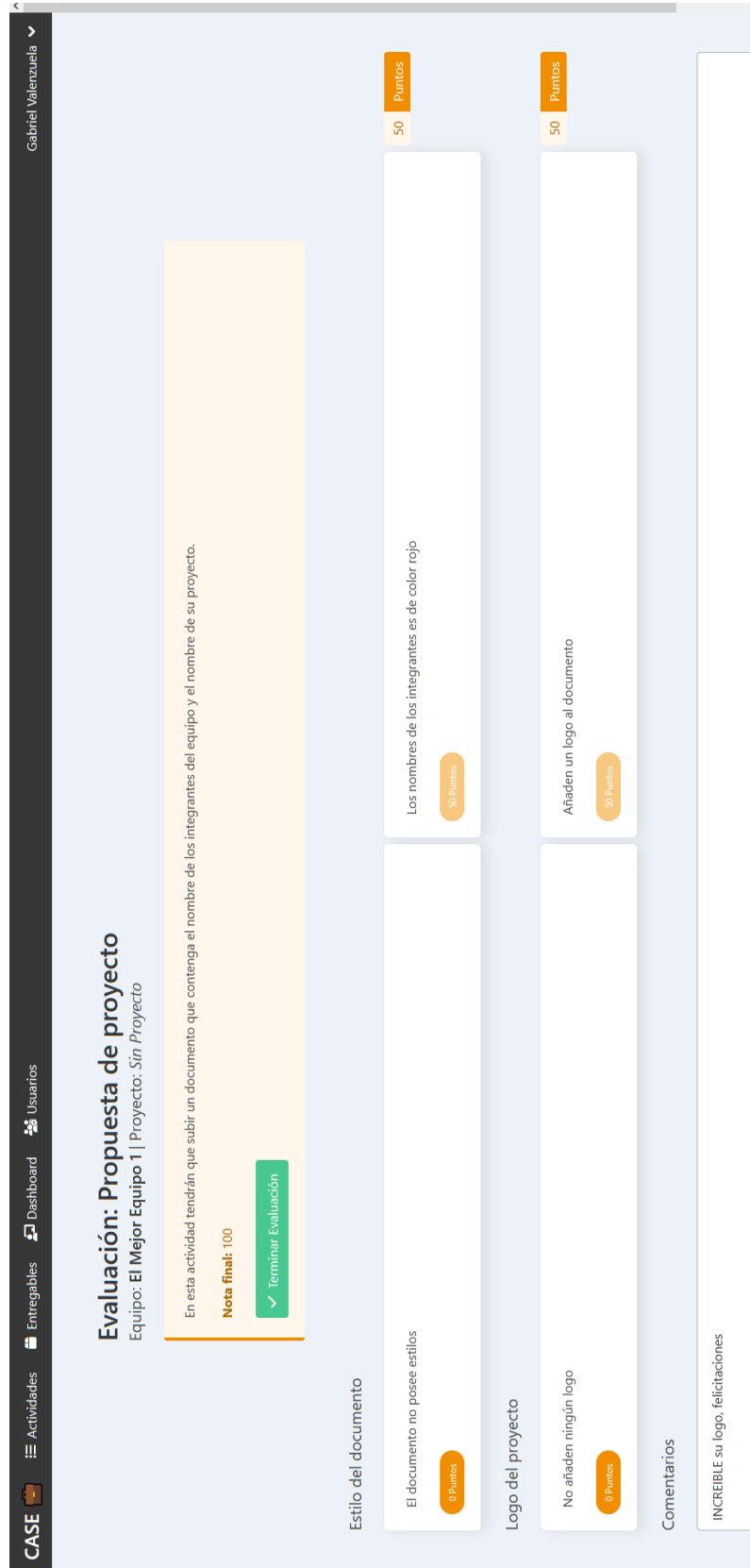


Figura A.9: Pantalla de evaluación de rúbricas de la plataforma expandida. Fuente: elaboración propia.

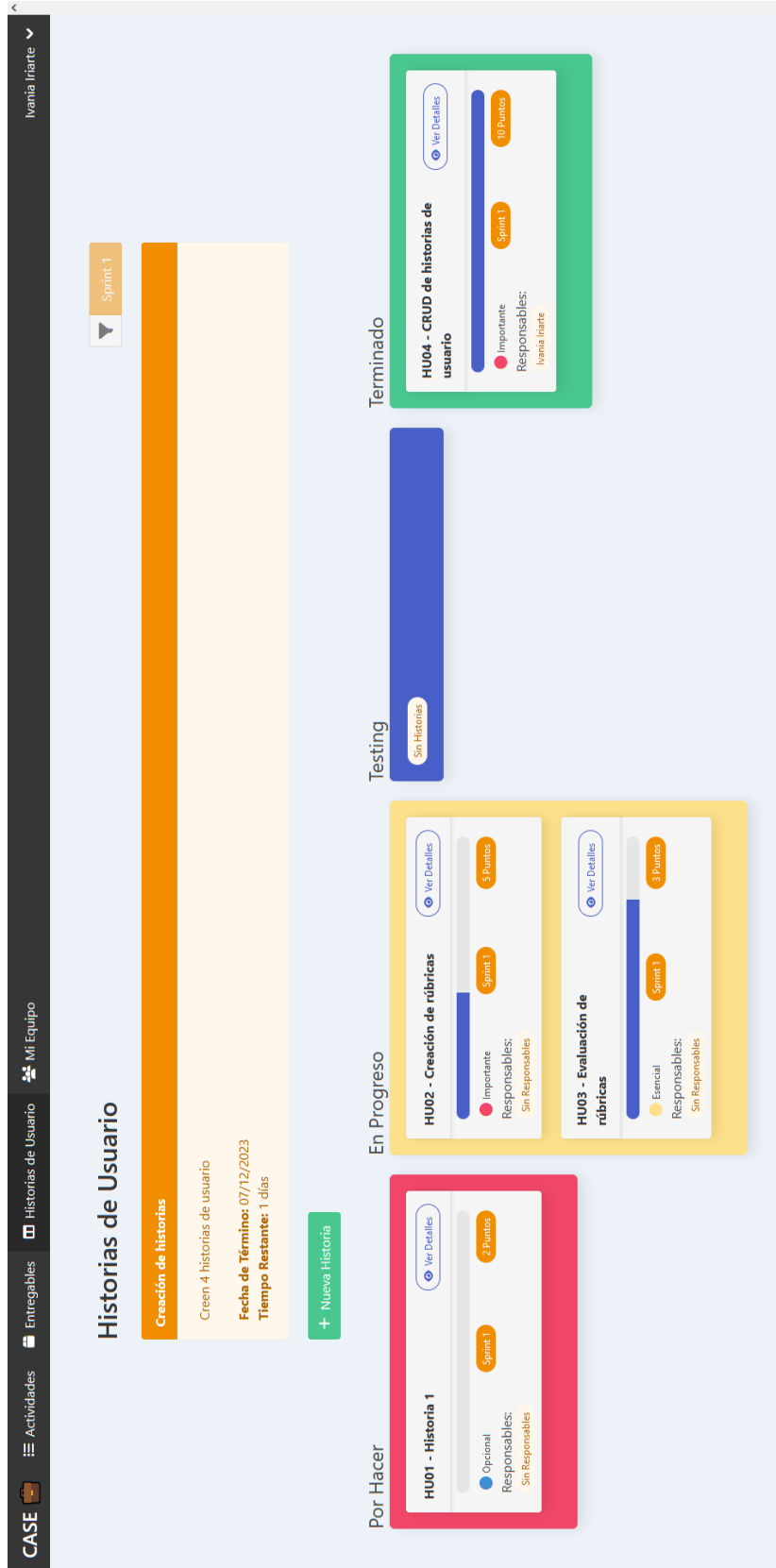


Figura A.10: Pantalla del tablero de historias de usuario de la plataforma expandida. Fuente: elaboración propia.

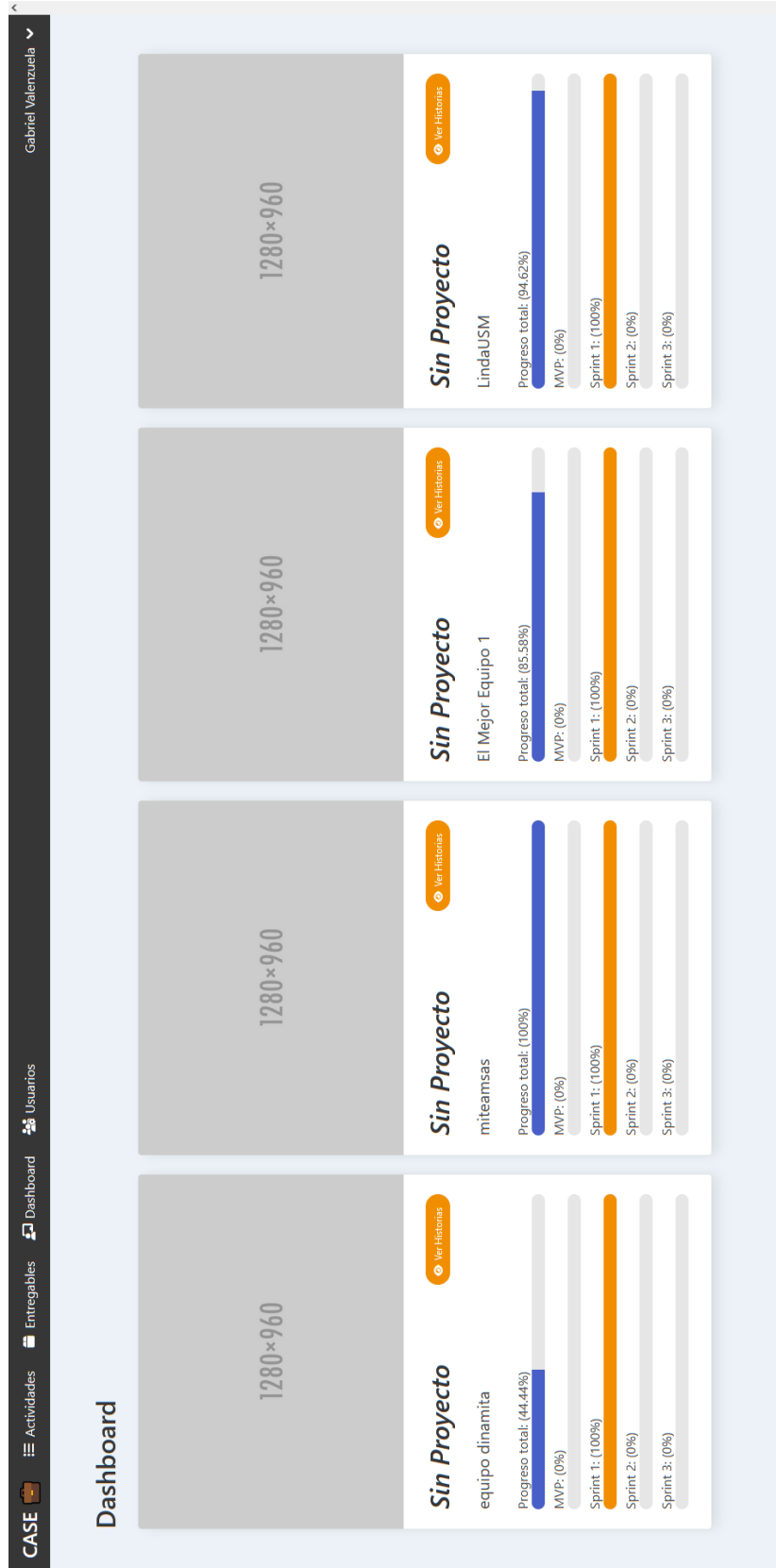


Figura A.11: Pantalla de progreso de los proyectos o *dashboard* expandida. Fuente: elaboración propia.

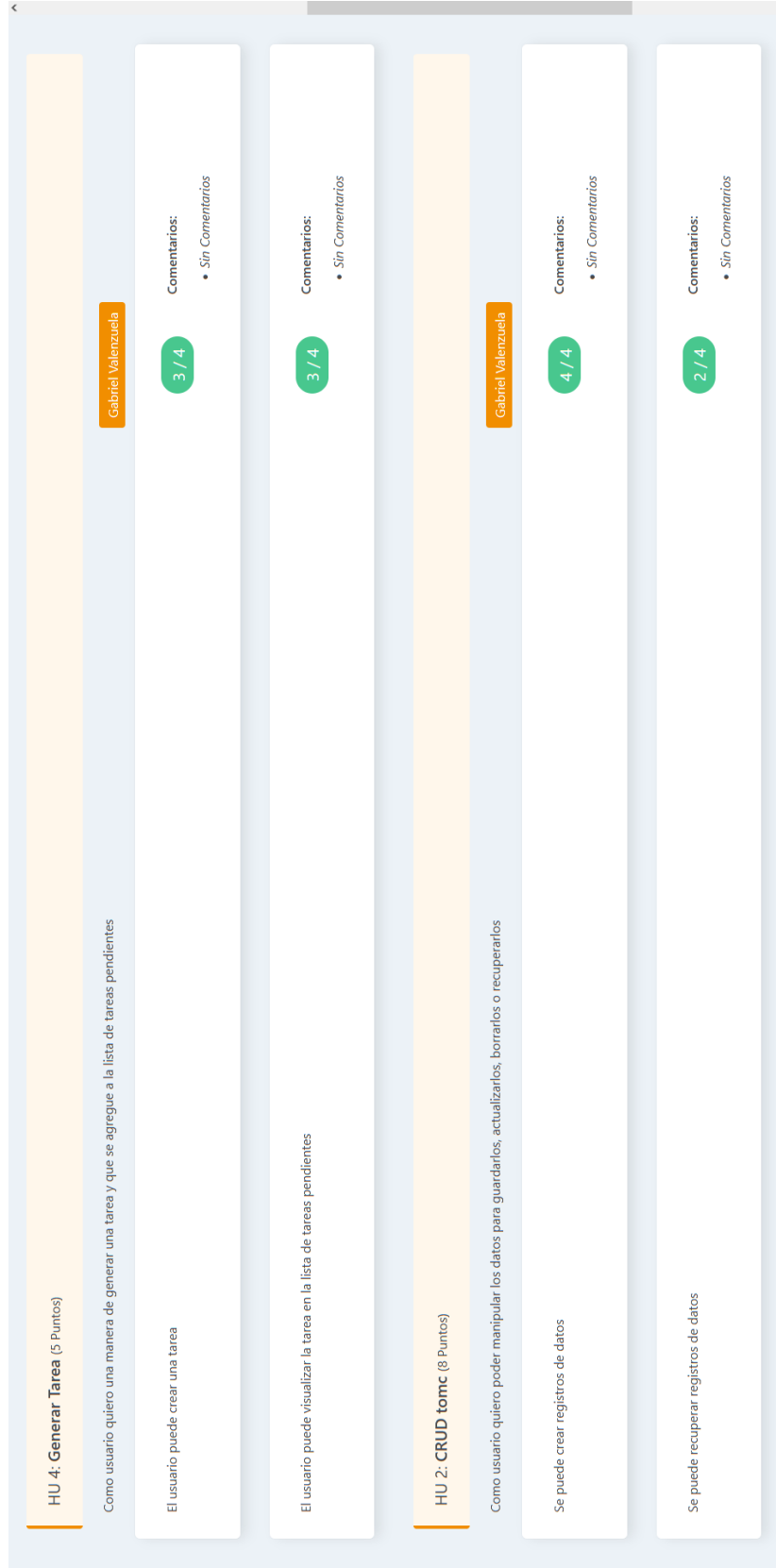


Figura A.12: Pantalla de evaluación de *sprints* expandida. Fuente: elaboración propia.

CASE

Actividades

Entregables

Historias de Usuario

Mi Equipo

Ivania Iriarte

1280x960

Equipo

Editar Equipo

Nombre*

CLTSofit

Frase

Ayudando a la salud de las personas

LinkedIn

No ingresado

Miembros

Ivania Iriarte

1280x960

Proyecto

Editar Proyecto

Nombre

No ingresado

Descripción

No ingresado

Figura A.13: Pantalla de equipo y proyecto expandida. Fuente: elaboración propia.

99

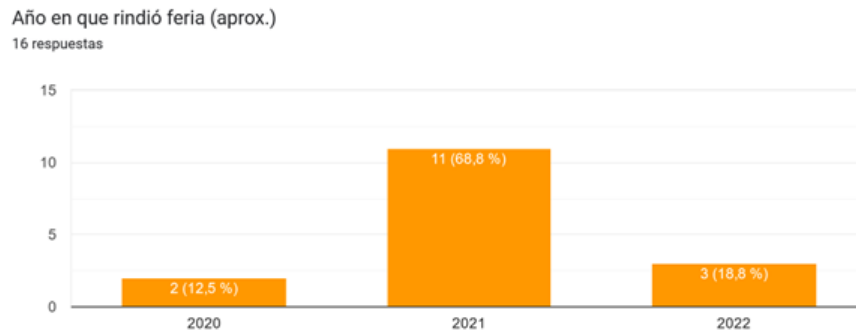


Figura A.14: Gráfico del año de rendición de la Feria de Software. Fuente: Google Forms

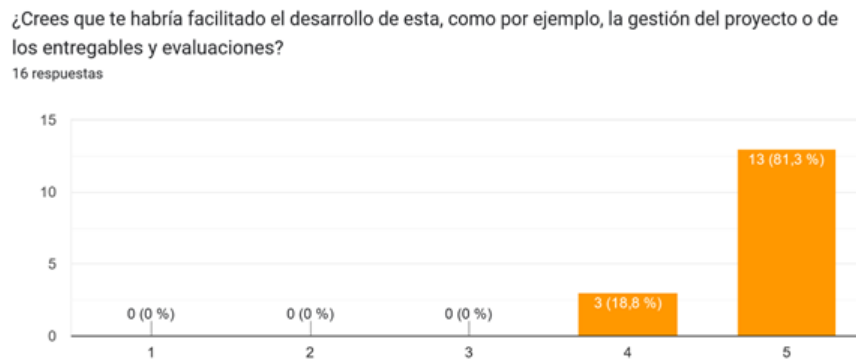


Figura A.15: Gráfico de los resultados obtenidos en la pregunta 1 de la **Sección 4.3.2**. Fuente: Google Forms

¿Crees que te habría ayudado a tener más en consideración los plazos de las distintas actividades de la asignatura?

16 respuestas

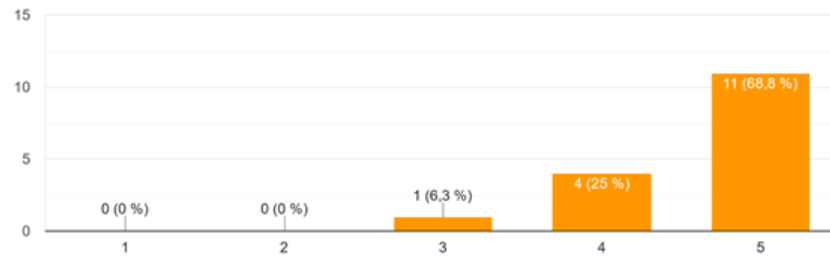


Figura A.16: Gráfico de los resultados obtenidos en la pregunta 2 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Crees que habría mejorado tu desempeño como estudiante?

16 respuestas

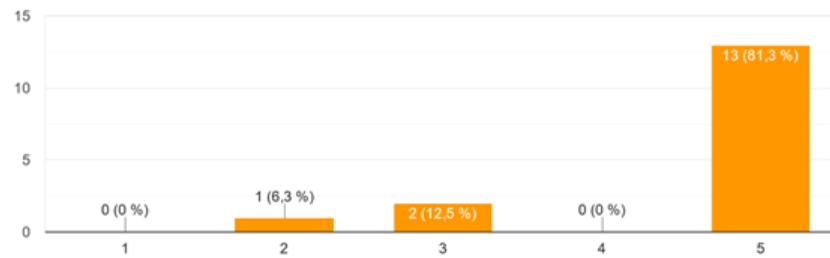


Figura A.17: Gráfico de los resultados obtenidos en la pregunta 3 de la **Sección 4.3.2**.
Fuente: Google Forms



Figura A.18: Gráfico de los resultados obtenidos en la pregunta 4 de la **Sección 4.3.2**.
Fuente: Google Forms

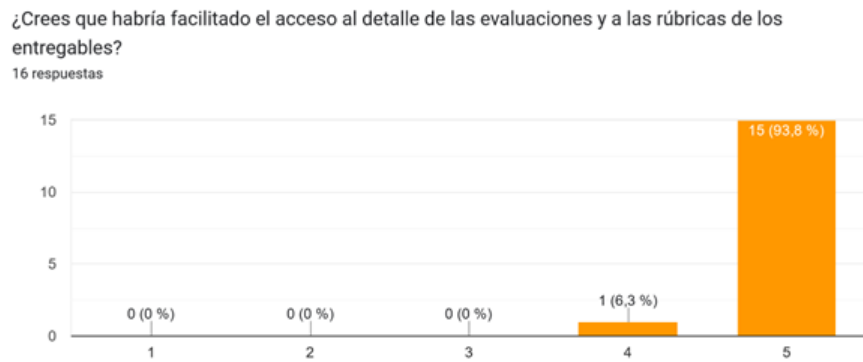


Figura A.19: Gráfico de los resultados obtenidos en la pregunta 5 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Crees que habría ayudado a evitar problemas en el desarrollo del proyecto o, en un peor caso, el fracaso de este?

16 respuestas

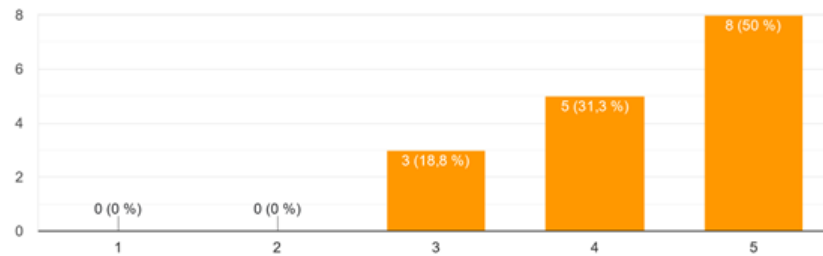


Figura A.20: Gráfico de los resultados obtenidos en la pregunta 6 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Sientes que el hecho de que los docentes tengan más información acerca del desempeño de tu equipo habría ayudado a que el resultado final de tu proyecto tuviese mayor calidad?

16 respuestas

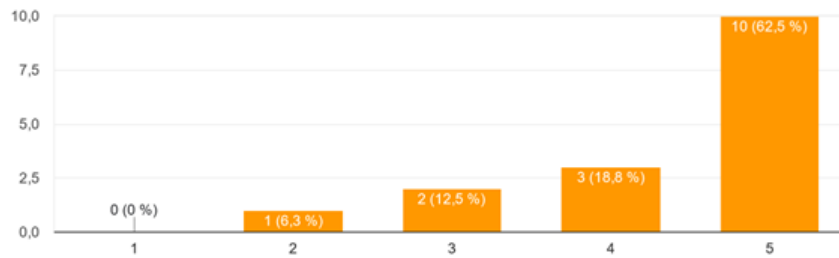


Figura A.21: Gráfico de los resultados obtenidos en la pregunta 7 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Crees que es fácil de usar?
16 respuestas

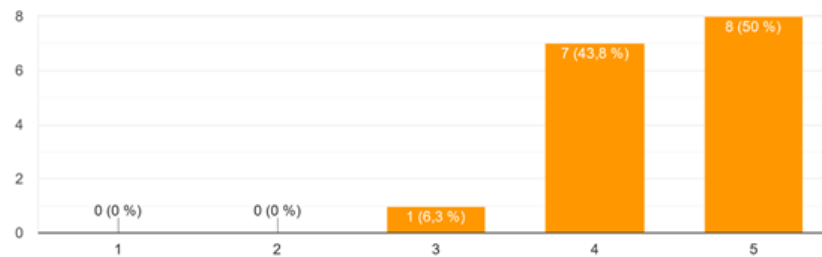


Figura A.22: Gráfico de los resultados obtenidos en la pregunta 8 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Crees que es útil?
16 respuestas

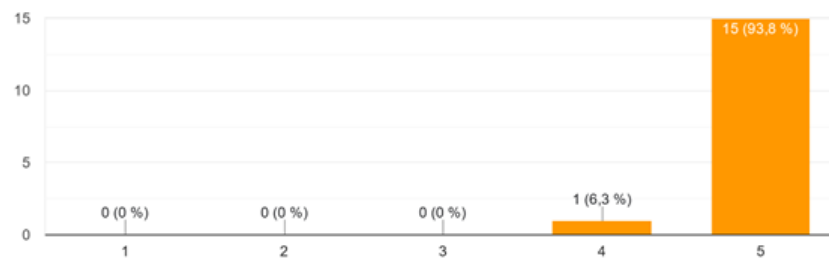


Figura A.23: Gráfico de los resultados obtenidos en la pregunta 9 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Crees que funciona de manera eficiente?
16 respuestas

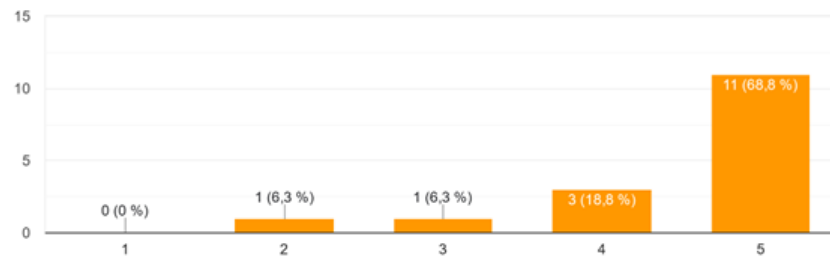


Figura A.24: Gráfico de los resultados obtenidos en la pregunta 10 de la **Sección 4.3.2**.
Fuente: Google Forms

¿Te gustaría que fuese implementada en la asignatura?
16 respuestas

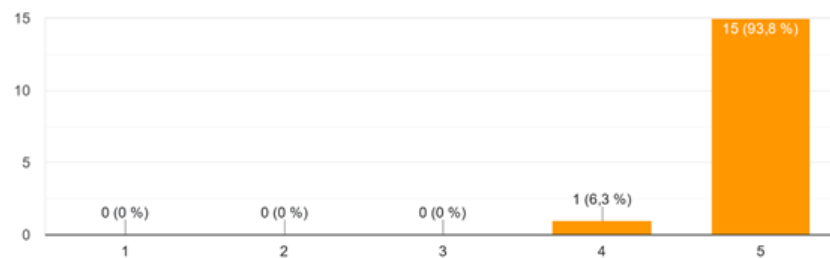


Figura A.25: Gráfico de los resultados obtenidos en la pregunta 11 de la **Sección 4.3.2**.
Fuente: Google Forms