Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema de evaluación en línea para programas estructurados

Tesina que presenta

Alberto Antonio Rivera García

Para obtener el grado de

Licenciado en Ciencias de la Computación

Asesor

Mtra. Karina Rosales López

Puebla, Puebla. Otoño de 2014

Dedicatorias

Dedicado de manera muy especial a toda mi familia con la siempre he contado y a Mayra por caminar todos los días a mi lado.

Agradecimientos

Finalmente ha llegado el tiempo de agradecer a todas las personas que me ayudado a salir adelante desde que tengo uso de razón.

Agradezco a todos los profesores que dejaron huella en mi desarrollo académico, al M.C. Miguel Rodríguez Hernández, a la Dra. Etelvina Archundia Sierra y al M.C. Alfonso Garcés Báez por apoyarme durante mis estudios profesionales. A mis sinodales, al Dr. Juan González Calleros y al Mtro. Luis Montealegre por dedicarle tiempo a mi trabajo y a mi asesora la Mtra. Karina Rosales López a quien jamás comprendí ni un poquito pero siempre me la pasé muy bien con ella.

Agradezco profundamente a todos y cada uno de los miembros de mi familia por estar siempre conmigo. Gracias a mi abuela y al inmenso cariño que tiene por todos los que somos sus hijos. Gracias a mi abuelo que a pesar de los años continúa con esa voluntad tan inquebrantable que siempre le he admirado. A mis tíos Oscar y Roberto, mis verdaderos padres y mis tías Mary y Laura por su incondicional afecto.

Agradezco a mi madre que a pesar de tantos acontecimientos y tantos años nunca ha dejado de verme como un niño. Ahora entiendo que si peleamos tanto es porque tenemos el mismo maldito carácter. Gracias a ti Alejandro por soportarme durante mi niñez (que aún no acaba). Hoy sé que mi madre no pudo haber elegido a un hombre mejor.

Un agradecimiento muy especial a mis amigos: Saúl, Gerardo y Arturo quienes jamás me abandonaron a pesar de mis imprudencias y mis estupideces. Créanme o no, para mis son mis hermanos y debo confesar que no existe día que no tenga un rato libre para recordar las cosas vivimos, que divagamos, las que planeamos y las que construimos (y destruimos) hace ya muchos ayeres. De igual manera, agradezco a los ‘ausentes’, a aquellos que estuvieron a mi lado en situaciones tan difíciles y de quienes tuve que separarme por razones que sigo sin entender.

Gracias a Sandra, Karen, a Janeth, a Diana, a Miguel, a Nelly Karen y a todos aquellos con quienes tuve la fortuna de compartir un poco de lo poco que realmente sé. Muchas gracias por hacerme sentir el hombre más inteligente del planeta.

Y te agradezco sobre todo a ti Mayra, que nunca me has abandonado desde que tuve la dicha de conocerte. Quiero que sepas que este logro te pertenece más a ti de lo que me pertenece a mí por todo lo que ha implicado. Nunca podré dejar de agradecerte por todas tus atenciones y en especial por todos tus sacrificios. Simplemente quiero expresarte que no he conocido a nadie que dedique tanto amor y cariño por otra persona de la misma forma en que tú haces conmigo día a día. Gracias por ayudarme a seguir adelante y gracias por estar escribiendo nuevamente nuestra historia.

Finalmente, agradezco a aquella persona que a pesar de su corta edad me enseñó la lección más importante de mi vida y que hoy comparto con aquellas personas que se aventuren a leer mi trabajo.

La vida no se trata de encontrarte a ti mismo, la vida se trata de construirte a ti mismo.

"Life isn't about finding yourself, life's about building yourself"

Alberto Antonio Rivera Garcia

Contenido

Introducción

En las instituciones educativas las metodologías de evaluación están compuestas por procedimientos que determinan las capacidades de sus estudiantes en ámbitos de resolución de problemas mediante la implementación de exámenes escritos.

Aunque estas metodologías cumplen con sus objetivos en la mayoría de las asignaturas existentes, ofrecen resultados cuestionables frete a asignaturas de naturaleza pragmática, como lo son las materias de que sientan las bases de la programación estructurada hablando propiamente de las tecnologías de la información.

El paradigma de programación estructurada es uno de los más difundidos en las áreas de la computación, debido a que su flujo imperativo resulta fácil de comprender por principiantes en estas áreas. Sin embargo, los procesos de enseñanza actuales en estos niveles no garantizan resultados eficientes debido a sus técnicas mal enfocadas.

Las limitaciones de estos procesos tienden a repercutir a largo plazo en las capacidades de los estudiantes, quienes pueden verse incapaces de afrontar problemáticas reales que les sean planteadas en diversas etapas de su desarrollo educativo e incluso profesional, por lo que son necesarios mecanismos de evaluación que fomenten el desarrollo de sus capacidades pragmáticas y prácticas.

Capítulo 1 . Marco teórico

1.1. Proceso de Desarrollo de Software

Se denomina proceso de desarrollo de software al conjunto de fases que integran el ciclo de producción de un sistema informático desde sus orígenes representados mediante modelos conceptuales hasta su distribución como un producto tangible. Dichas fases dependen directamente de la metodología de desarrollo que sea elegida para la implementación del software. [1]

Metodología de desarrollo de software

Una metodología de desarrollo es un mecanismo que define el conjunto de detalles organizativos que permiten implementar las fases dentro de proceso de desarrollo de un proyecto de software. Estos detalles involucran técnicas de desarrollo, herramientas y modelos a seguir con la finalidad de obtener un 'producto' de calidad. [2]

Metodologías de desarrollo de tradicionales

Este tipo de metodologías tiende a imponer una estricta disciplina de trabajo sobre los procesos de diseño y documentación del software. Se enfocan en una planificación rigurosa y detallada antes de iniciar las fases de implementación y producción. Sin embargo, ofrecen muy poca flexibilidad frente a cambios en los requisitos de los usuarios.

Entre las metodologías tradicionales se destacan:

- Rational Unified Process o RUP
- Microsoft Solution Framework o MSF

Metodologías de desarrollo ágil

Este tipo de metodologías ofrecen excelentes resultados frente a cambios súbitos en los requisitos del sistema ya que fomentan la participación de sus futuros usuarios en los procesos de desarrollo y retroalimentación. Favorecen el trabajo arduo sobre los componentes de software por encima los procesos de control y la documentación detallada.

La flexibilidad brindada por estás metodologías permite mitigar de manera considerable efectos producidos por errores detectados en las fases de prueba y aceptación de los usuarios finales facilitando las labores de mantenimiento y extensión de las funcionalidades existentes. [3]

Entre las metodologías de desarrollo ágil podemos mencionar:

- SCRUM
- eXtreme Programming o XP
- Crystal Clear

- Dynamic Systems Development Method o DSDM

Principios de desarrollo

Es indispensable tener siempre en mente ciertos principios que permitan diseñar una buena arquitectura dentro de un sistema en aras de minimizar sus costos de mantenimiento y que promuevan la reusabilidad y extensibilidad de sus componentes sin sacrificar su desempeño. Aunque existe un gran número de ellos y sus descripciones varían en la literatura, existe un conjunto reducido de ellos aceptados de manera general como imprescindibles por diversas comunidades de desarrolladores.

Separation of concerns (SoC). Un sistema debe subdividirse en diversos componentes que superponen sus funciones en el menor grado posible.

Single responsibility principle (SRP). Cada componente del sistema debe tener una única funcionalidad.

Principle of least knowledge. Un componente no debe conocer detalles de la implementación interna de otro componente.

Don not repeat yourself (DRY). Un componente no debe duplicar la constitución interna de otro componente.

Minimize upfront design. Se debe buscar diseñar solo los componentes necesarios minimizandos sus funciones.

Open-close principle (OCP). El diseño de cada componente debe realizarse pensando en su crecimiento, no en su modificación.

Patrones de diseño

Los patrones de diseño definen arquetipos que ofrecen soluciones a problemas comunes surgidos en las fases de diseño y desarrollo de aplicaciones reales. Fomentan la reusabilidad de los modelos que definen los componentes de un sistema y facilitan su interrelación. [4]

Por lo general, se catalogan en tres grupos diferentes: creacionales, estructurales y de comportamiento.

Patrones de diseño creacionales

Se encargan de abstraer los procesos de instanciación encapsulando la lógica que describe la manera en que los componentes de un sistema son creados, combinados o representados.

Patrones de diseño estructurales

Definen modelos que los componentes de un sistema pueden seguir para facilitar su combinación resultando en estructuras complejas.

Patrones de diseño de comportamiento

Facilitan la comunicación entre los componentes de un sistema fomentando su flexibilidad y facilitando su mantenimiento.

Historias de usuario

En cualquier proyecto de software existirán cambios en los requisitos del cliente que afectarán en diferentes grados los costos de implementación de una o más de sus fases. Es imposible predecir estos cambios, por lo que las especificaciones de los requisitos pueden verse comprometidas si las herramientas usadas para su documentación no ofrecen suficiente flexibilidad.

Una historia de usuario es una herramienta usada comúnmente en metodologías de desarrollo ágil de aplicaciones, caracterizada por su facilidad para representar mediante sentencias bien formadas aquellas tareas que darán a lugar a alguna funcionalidad del sistema. [5]

Modelo INVEST

El modelo INVEST es un esquema que determina las características que debe seguir una historia de usuario.

Caracteristica	Descripción
Independent	No dependerá de otra y permitirá ser revisada en cualquier orden sin que esto afecte su significado ni propósito
Negotiable	No describirá detalles técnicos relacionados con su implementación
Valuable	Deberá ofrecer algún valor real para el usuario, por lo que tendrá que describir alguna funcionalidad para el usuario que la dirige
Estimable	Ofrecerá nociones con respecto al tiempo necesario para su desarrollo
Small	Deberá ser presisa y concisa
Testable	Desde la perspectiva del desarrollador, generará resultados que permitan ser verificables

Tabla 1.1 El modelo INVEST

Tareas

Dentro de cualquier proceso existente, dígase de control o de gestión se desenvuelven múltiples escenarios que son necesarios para completar su ciclo y poder considerarlo como exitoso o no.

En el desarrollo ágil de aplicaciones, es habitual representar mediante historias de usuario estos procesos, descritas en el apartado anterior. Cada uno de estos escenarios puede dividirse en tareas. Una tarea representa una única actividad dentro un proceso más complejo o dicho de otra manera, un proceso es la suma de sus tareas inherentes. [6]

Modelo SMART

Smart es un modelo conceptual que determina las características que deben poseer las tareas que integran un proceso. Este modelo es descrito en la siguiente tabla.

Caracteristica	Descripción
Specific	Deberá desribir detalles precisos de las acciones que la componen
Measurable	Permitirá ser medida bajo parámetros medibles o estimables
Achievable	Deberá ser realista
Relevant	Aportará un valor real al el escenario en el que se desenvuelve
Time-boxed	Será finita y su duración por lo menos deberá ser estimable

Tabla 1.2 El modelo SMART

1.2. Desarrollo dirigido por pruebas

El Desarrollo Dirigido por Pruebas o*Test Driven Development* es una técnica de desarrollo de aplicaciones donde las pruebas dirigen la construcción de las funcionalidades del sistema. El resultado de esta técnica es un conjunto de pruebas que satisfacen cada uno de los requisitos de los usuarios íntegramente. [7]

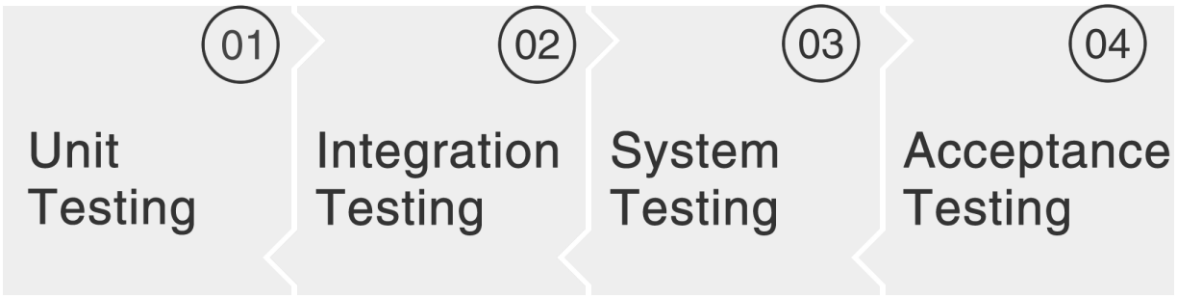


Figura 1.1 Las etapas del proceso de pruebas.

Uso de la programación dirigida por pruebas

Aunque existen diferentes maneras de aplicar está técnica, por lo general, podemos describirla en los siguientes pasos:

1. Entender los requisitos de la funcionalidad que se desean solventar
2. Codificar una prueba para dicha funcionalidad
3. Ejecutar la prueba y esperar que fracase
4. Generar únicamente el código necesario para conseguir que la prueba sea existosa
5. Ejecutar la prueba y verificar que tenga éxito
6. Refactorizar el código existente

De manera convencional, se acepta el uso del enfoque "*Red*, *Green*, *Re-factor*" para esquematizar los pasos descritos anteriormente.

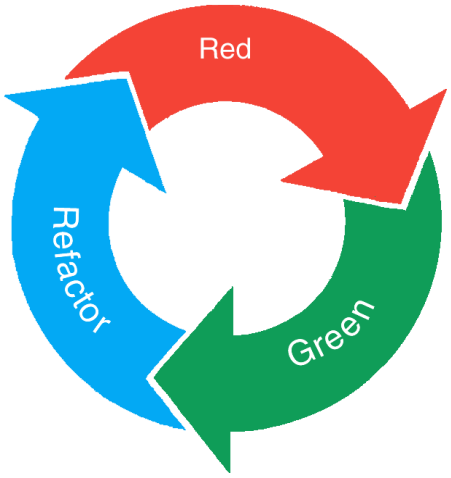


Figura 1.2 El enfoque *"Red, Green, Re-factor!"*.

La fase roja o *Red*

Antes de escribir una sola línea de código es necesario familiarizarse con el requisito que se desea satisfacer provocando que la prueba fracase para determinar sus alcances.

La fase verde o *Green*

Teniendo en mente los alcances del requisito que se necesita implementar, es necesario escribir la mínima cantidad de código necesaria para considerar la prueba como exitosa, esto significa llegar a la fase verde.

La fase de refactorización o *Refactor*

Esta fase describe el proceso iterativo de re-estructurar el código existente sin afectar su funcionamiento y los resultados que de él se esperan. La refactorización permite un buen diseño del código generado asegurando un desempeño óptimo, fácil de mantener y extender.

Pruebas unitarias

Una prueba unitaria, también denominada prueba de caja blanca (*White Box*) representa la unidad fundamental del desarrollo dirigido por pruebas. Su objetivo es aislar una fracción del código fuente y determinar si genera los resultados que le corresponden en tiempo y forma. [8]

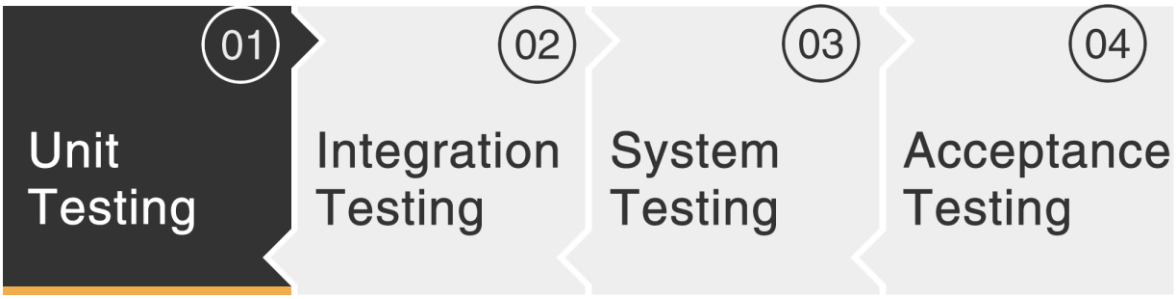


Figura 1.3 Etapa de *unit testing*. Cada funcionalidad del sistema dará a lugar a una nueva prueba unitaria. Cada prueba unitaria será implementada de acuerdo a las tres fases descritas en el enfoque "*Red, Green, Re-factor*".

La siguiente tabla describe las características que debe poseer una buena prueba unitaria.

Característica	Descripción
Automatic	Debe ser ejecutada sin necesidad de la intervención humana
Repeatable	Permitirá ser ejecutada tantas veces como se requiera
Independent	No afectará la ejecución o el resultados de otra prueba unitaria
Predictable	Generará siempre los mismos resultados

Tabla 1.3 Características de una buena prueba de unidad.

Pruebas de integración

Una vez culminadas las pruebas unitarias se dará inicio al proceso de integración de sus funcionalidades resultando en componentes más complejos. Durante este proceso será necesario verificar que los resultados de dichas funcionalidades no sean comprometidos y es responsabilidad de las pruebas de integración realizar esta actividad.

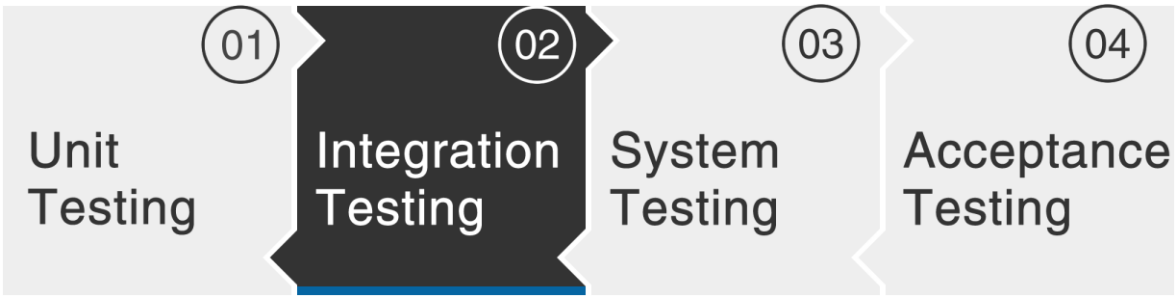


Figura 1.4 La etapa de *integration testing*. Los componentes resultantes de las pruebas de unidad se integran y se verifica que se comuniquen de manera adecuada.

Existen pruebas con mayores niveles de abstracción. Por una parte, las pruebas de sistema se encargan de verificar que los componentes resultantes del proceso de integración se desempeñan eficazmente como una unidad absoluta, tomando en cuenta diversas variables de rendimiento.

Por otra parte, las pruebas de aceptación determinan si las funcionalidades del sistema entero satisfacen íntegramente las necesidades del cliente.

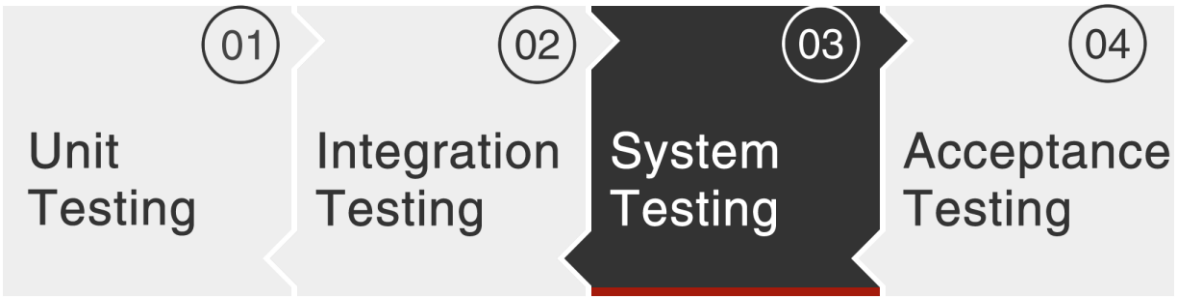


Figura 1.5 La etapa de *system testing*.

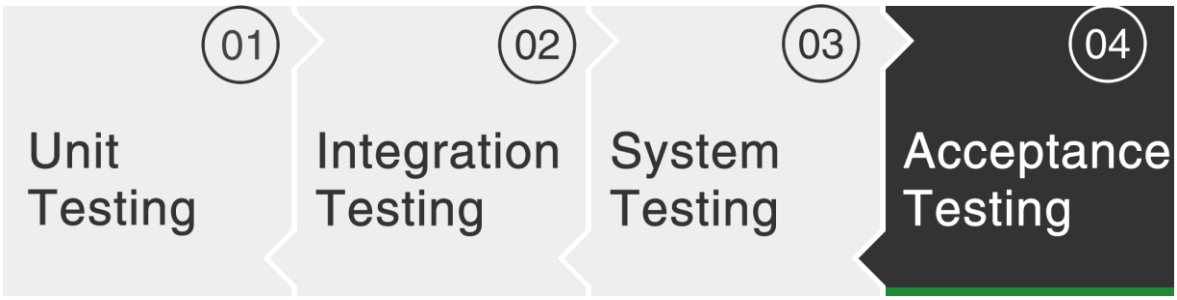


Figura 1.6 La etapa de *acceptance testing*.

1.3. Arquitectura de un sistema

El diseño de la arquitectura de un sistema es el proceso por medio de cual se define su distribución física y lógica. En ella se establecen los requisitos técnicos que competen al dominio de la aplicación. Algunos de estos requisitos son los siguientes:

- **Plataforma de desarrollo**. Establece qué sistema operativo será usado para el desarrollo del proyecto identificando sus principales características, ventajas, desventajas y compatibilidad con las tecnologías de desarrollo.
- **Marcos de desarrollo**. Se determinan que tecnologías de desarrollo serán usadas en las fases de implementación.

- **Persistencia de la información.** Se definen los mecanismos de persitencia de datos. Si se trata de una base de datos, se establece la tecnología de administración, identificando sus caracteríscias de compatibilidad con respecto al sistema operativo seleccionado y los marcos de desarrollo elegidos.
- **Presentación de la información.** Se especifica los mecanismos de entrega de resultados que serán presentados a los usuarios finales. Es muy importante resaltar, que estos mecanismos deben ser completamente independientes del resto del la aplicación.
- **Requisitos adicionales.** Es posible que el sistema requiera de funcionalidades adicionales de seguridad (acceso por credendial, lector de huyea digital) o definir si se trata de un sistema distribuido o concurrente.

Estilos arquitecturales

Existe gran cantidad de estilos arquitecturales cuya definición y clasificación varían en la literatura existente. La siguiente tabla resume algunos estilos arquitecturales más comunes y sus características más importantes. [9]

Nombre	Descripción	Ventajas
Cliente/Servidor	Define una relación entre dos aplicaciones, una del lado del cliente y otra de lado del servidor. El cliente hará solicitudes al servidor en espera de una respuesta	Seguridad, centralización de la información, disponibilidad con multiples servidores
Basado en componentes	Define un conjunto de componentes que exponen interfaces bien diseñadas y colaboran entre ellos para responder solicitudes emergentes	Reusabilidad y extensibilidad
En capas (N-Layer)	Implementa una distribución jerarquica asignando responsabilidades a entidades denominadas capas favoreciendo un bajo acoplamiento de la aplicación emergente	Aislamiento, separación de responsabilidades, optimización de desempeño, independencia del hardware
Orientado a servicios	Ofrece sus funcionalidades a otra aplicación(comsumer) mediante conjunto de operaciones denominadas servicios	Autonomía, bajo acoplamiento entre consumidor y servidor, independencia de la plataforma.
Presentación desacoplada	Matiene la funcionalidad de aplicación en diferentes niveles físicos (Tiers).	Independencia entre las funcionalidades, escalabilidad y alta disponibilidad

Tabla 1.4 Los estilos arquitecturales más comunes.

Arquitectura en N-Capas

Una arquitectura en N-Capas separa las responsabilidades de la aplicación en diversas estructuras organizacionales con funciones específicas *(SoC – Separation or Concerns)* generando un diseño flexible, extensible y de fácil manutención.

Es importante mencionar las diferencias existentes entre los conceptos ‘capas’ y ‘niveles’. Mientras que una capa representa una unidad organizacional lógica dentro de una aplicación, un nivel describe las características de la distribución física de sus componentes.

Figura 1.7 Distribución convencional de una arquitectura en N-Capas.

1.4. Persistencia de la información

Una de las consideraciones más significativas que debe realizarse antes del proceso de desarrollo es definir la manera en que los resultados de las operaciones persistirán a través de cada uno de los estados de una aplicación.

Habitualmente, la persistencia de la información puede establecerse mediante el diseño e implementación de una base de datos, mediante operaciones de lectura-escritura de ficheros almacenados en discos rígidos o incluso a través de una combinación de ambos mecanismos.

Base de datos

Una base de datos es una colección de datos interrelacionados y administrados que operan bajo un mismo contexto. [10]

Enfoques de implementación de una base de datos

El desacoplamiento de los mecanismos de gestión de bases de datos representa uno de los cambios más representativos en los últimos años en relación a la manutención, extensibilidad y escalabilidad de una aplicación.

El diseño de una aplicación a partir de sus modelos de datos suele ser complicado frente a cambios súbitos en los requisitos de sus usuarios o incluso frente a errores detectados en las fases de implementación. En el peor de los escenarios será necesario reconstruir el sistema desde sus orígenes.

Con el paso de los años, estos problemas han sido solventados gradualmente mediante nuevos enfoques que fomentan el diseño aplicaciones priorizando el valor de sus componentes internos y que independizan los mecanismos de acceso a la información. En la actualidad existen tres enfoques diferentes: *model first*, *schema first* y *code first*.

El enfoque *model first*, representa los procesos tradicionales de diseño aplicaciones en donde la base de datos define la estructura lógica del sistema emergente. *Schema first* permite generar diagramas de clases compuestos de las entidades que conforman el domino de datos de la aplicación. A partir de este diagrama es posible generar la base de datos del sistema o viceversa.

El enfoque code first

Permite generar la base de datos de una aplicación en función de su dominio de información, es decir, las entidades que componen la capa de dominio estarán relacionadas con una tabla existente en la base de datos mediante un proceso de vinculación. [11]

El proceso de vinculación se lleva a cabo mediante convención y no por configuración como en otro tipo de enfoques. En la actualidad, existen *frameworks* especializados que solventan las tareas de vinculación necesarias para este proceso, ofreciendo altos de nivel de abstracción y transparencia a los desarrolladores.

Las ventajas de este enfoque son muchas, pero una de ellas resulta realmente importante: fomenta la independencia de las tecnologías usadas para la administración la información mitigando los costos producidos por cambios en los requisitos de los usuarios.

1.5. Presentación de la información

La presentación de la información establece qué mecanismos usará el sistema para ofrecer sus funcionalidades a los usuarios que interactuarán con él. Estos mecanismos deben ser independientes del resto de la aplicación permitiendo disponer de diferentes plataformas para la presentación de los mismos resultados.

Arquitectura Model-View-Model (MVC)

Es una arquitectura orientada al desarrollo de aplicaciones web que destaca por su versatilidad para crear diferentes representaciones con los mismos conjuntos de datos. Está fundamentada a partir del ciclo natural que describe las interacciones entre las acciones del usuario y sus resultados, por lo que representa la arquitectura idónea para el desarrollo de aplicaciones de tipo cliente servidor. [12]

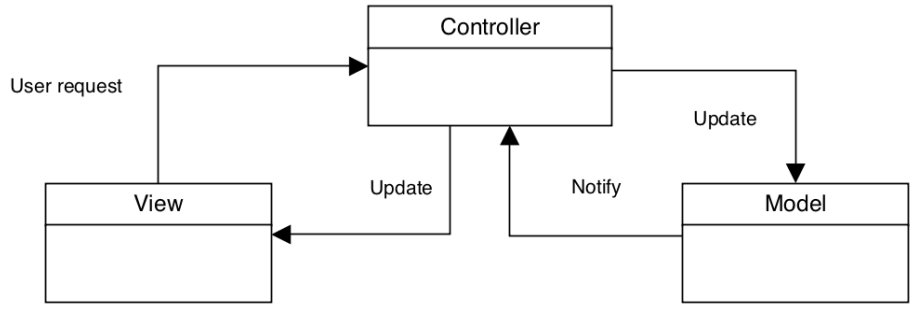


Figura 1.8 El patrón de diseño MVC.

Patrón de diseño Model-View-View-Model (MVVM)

Model-View-View-Model es un patrón de diseño estructural orientado a representar el estado y comportamiento de un conjunto de datos independientemente de la plataforma que se use para presentarlos. [13]

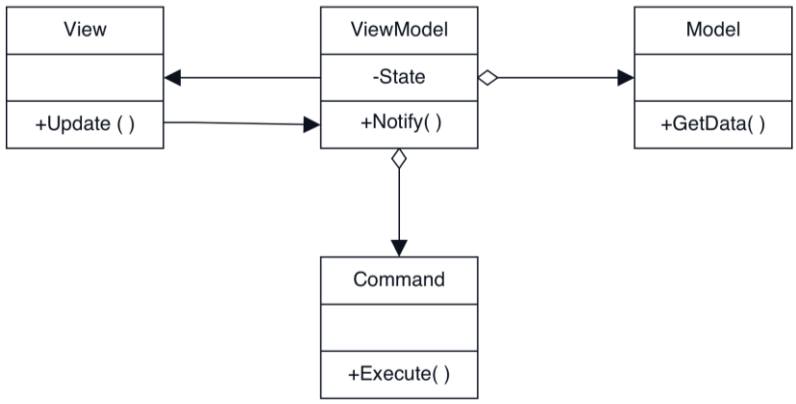


Figura 1.9 El patrón de diseño MVVM

Estrategias Push y Pull

Una de las tecnologías más fascinantes usadas para ofrecer servicios en aplicaciones para diferentes plataformas se denomina *push technology*. En ella, el servidor 'empuja' información al usuario sin que este la solicite de manera explícita, en contraste con las tecnologías convencionales (*pull technologies*) enriqueciendo la experiencia de uso.

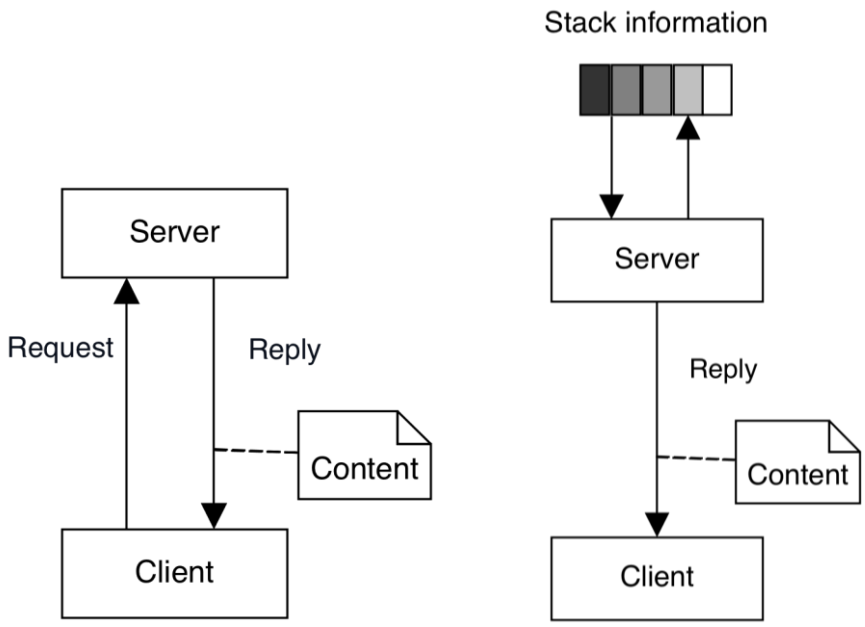


Figura 1.10 A la izquierda se muestra la tecnología convencional *pull* usada en la mayoría de las aplicaciones web existentes. La tecnología de empuje *o push (derecha)* envía información al usuario sin que este la solicite explícitamente.

El servidor envía mensajes hacia los usuarios que se encuentren 'subscritos' a algún tipo de servicio. Este tipo de tecnologías es común en redes sociales en donde se hace uso de ventanas emergentes para notificar a sus usuarios sobre algún suceso que le puede interesar.

En escenarios profesionales, son usadas para aplicaciones colaborativas donde el tratamiento de información compartida sea un requisito de alta prioridad. Cada vez que un colaborador realice un cambio a la información existente, deberá verse reflejado para todos los usuarios con los que se comparta dicha información.

Aunque existen diferentes maneras de implementar estas tecnologías, en términos de infraestructura, su funcionamiento resulta bastante sencillo de describir. Por una parte, el servidor mantiene el estado de la conexión abierta y se encuentra a la espera de que cualquier evento relacionado con el servicio al que el cliente se encuentra subscrito. Si este evento ocurre se informa al usuario o se actualiza la información del servicio.

Por otra parte, si la conexión ha sido cerrada, el servidor gestiona una cola de información en espera de que la conexión se establezca nuevamente para informar al cliente.

Capítulo 2 . Diseño del sistema

2.1 Antecedentes

Los procedimientos de evaluación existentes aplicados a técnicas de programación estructuradas, consisten en pruebas escritas formadas por secciones teóricas y prácticas, donde el estudiante deberá dar soluciones a los problemas que le serán planteados por el instructor. Sin embargo, debido a las carencias de implementación de estos procedimientos, no se enriquecen las capacidades analíticas de los estudiantes ni se fortalece su seguridad ante problemáticas reales.

Con la finalidad de mitigar estas carencias, se somete al estudiante a realizar prácticas en laboratorios de cómputo donde deberán comprobar la efectividad de sus implementaciones. Sin embargo, es común que los estudiantes busquen soluciones ya desarrolladas en algún medio de consulta, por lo que este enfoque no garantiza autosuficiencia.

Aunado a lo anterior, ambos enfoques requieren participación activa del instructor para comprobar que las técnicas aplicadas por los estudiantes resultan convenientes para los propósitos con que fueron desarrolladas. Naturalmente, esto resulta agobiante para el instructor y no es de extrañarse que comenta errores disminuyendo la calidad de su trabajo.

Aunque existen sistemas automatizados que implementan mecanismos de evaluación, ninguno de ellos ofrece solución a todas las problemáticas que serán planteadas a lo largo de este trabajo. Por lo general, están orientados a personas con un grado desarrollado en técnicas de programación, diseño de algoritmos y resolución de problemas, así que no representan una alternativa apropiada para principiantes. Además, su uso es de carácter personal, por lo que carecen de mecanismos de gestión de actividades, uno de los requisitos buscados para los instructores.

2.2 Planteamiento del problema

Las metodologías de evaluación basadas en exámenes escritos que son aplicadas a asignaturas de programación estructurada ofrecen resultados deficientes que repercuten en el aprendizaje de los estudiantes.

2.3 Objetivo general

Desarrollar un sistema automatizado que sirva como mecanismo de retroalimentación a estudiantes y que facilite a profesores e instructores sus labores de evaluación en materias de programación estructurada y diseño de algoritmos.

2.4 Objetivos específicos

- Realizar el levantamiento de requisitos
- Diseñar la arquitectura lógica del sistema
- Generar la arquitectura lógica distribuida en las siguientes capas:
  - Dominio
  - Infraestructura
  - Operación
  - Presentación

2.5 Estado del arte

Existen en la red infinidad de sistemas con diversas metodologías y mecanismos de evaluación, que pretenden poner a prueba las capacidades de sus usuarios en ámbitos de resolución de problemas de diversos tipos.

Jueces de código en línea

Un juez de código es un sistema compuesto por un compendio de problemas clasificado bajo diferentes rubros. Aunque la mecánica de participación puede variar de sistema en sistema, su finalidad consiste en evaluar el código fuente de un usuario ante un problemática específica que puede representar o no una situación real y determinar su desempeño bajo parámetros como el tiempo, la memoria consumida o la cantidad de casos de prueba resueltos.

Las características de este tipo de sistemas son idénticas entre sí, al menos en sus metodologías de evaluación. Las métricas que rigen sus procesos de evaluación están basadas en las reglas de los concursos de programación *International Collegiate Programming Contest (ICPC)* organizados por la *Association for Computing Machinery (ACM)*. [14]

Taijin University Online Judge

http://acm.tju.edu.cn/toj/

Descripción

Un usuario registrado deberá seleccionar dentro un amplio catálogo organizado en capítulos, uno de los problemas existentes. Después de elegir un problema, el usuario tendrá que leerlo, analizarlo y diseñar una solución en alguno de los lenguajes de programación soportados. Posteriormente deberá publicar su código fuente y solicitará al sistema que evalué su desempeño. El sistema, en cuestión de segundos responderá su solicitud generando una retroalimentación compuesta por un código representativo y un resumen detallado sobre la ejecución del programa.

Ventajas

- Interfaz de usuario sencilla, minimalista y fácil de usar una vez que el usuario ha comprendido la mecánica interna del sistema.
- Organización del catálogo de problemas mediante capítulos.
- Gran variedad de problemas a resolver de diferentes ámbitos.
- Permite la gestión de eventos en tiempo real ofreciendo una modalidad tipo 'concurso' donde uno o más usuarios podrán competir poniendo a prueba sus capacidades bajo parámetros como número de problemas resueltos y desempeño de las soluciones. La intención de estos eventos es fomentar la competitividad entre los usuarios del sistema.

Desventajas

- Soporte limitado a tres lenguajes de programación: C, C++ y Java.
- Carece de un mecanismo completo de administración de usuarios por lo que funcionalidades tan esenciales como la 'recuperación de una contraseña' no son soportadas.
- Su uso está fuertemente vinculado a concursos de programación de índole algorítmica y matemática por lo cual no representa un buen sistema de retroalimentación para programadores novatos.
- Algunos errores en la programación interna provocan que cierto número de soluciones sean marcadas como erróneas cuando el resultado en realidad sea favorable.
- No ofrece ninguna herramienta o mecanismo de apoyo para instructores en sus labores de evaluación.

Figura 2.1 Página de inicio de TJU Online Judge.

Problems	From	To	Problems	From	To
Volume I	1001	1100	Volume II	1101	1200
Volume III	1201	1300	Volume IV	1301	1400
Volume V	1401	1500	Volume VI	1501	1600
Volume VII	1601	1700	Volume VIII	1701	1800
Volume IX	1801	1900	Volume X	1901	2000
Volume XI	2001	2100	Volume XII	2101	2200
Volume XIII	2201	2300	Volume XIV	2301	2400
Volume XV	2401	2500	Volume XVI	2501	2600
Volume XVII	2601	2700	Volume XVIII	2701	2800
Volume XIX	2801	2900	Volume XX	2901	3000
Volume XXI	3001	3100	Volume XXII	3101	3200
Volume XXIII	3201	3300	Volume XXIV	3301	3400
Volume XXV	3401	3500	Volume XXVI	3501	3600
Volume XXVII	3601	3700	Volume XXVIII	3701	3800
Volume XXIX	3801	3900	Volume XXX	3901	4000
Volume XXXI	4001	4095			

Figura 2.2 En TJU los problemas están clasificados en conjuntos de problemas denominados volúmenes. Este tipo de clasificación es recurrente en todos los sistemas tipo ACM-ICPC.

1001. Hello, world!

Time Limit: 0.5 Seconds Memory Limit: 65536K
Total Runs: 27962 Accepted Runs: 13665 Multiple test files

This is the first problem for test.

Since all we know the ASCII code, your job is simple: *Input numbers and output corresponding messages.*

Input

The input will contain a list of positive integers separated by whitespaces(spaces, newlines, TABs).
Please process to the end of file (EOF).
The integers will be no less than 32.

Output

Output the corresponding message.
Note there is NOT a newline character in the end of output.

Figura 2.3 El problema *1001. Hello World!* es un problema de aprendizaje. Su función es involucrar al usuario en la mecánica de los procesos de evaluación.

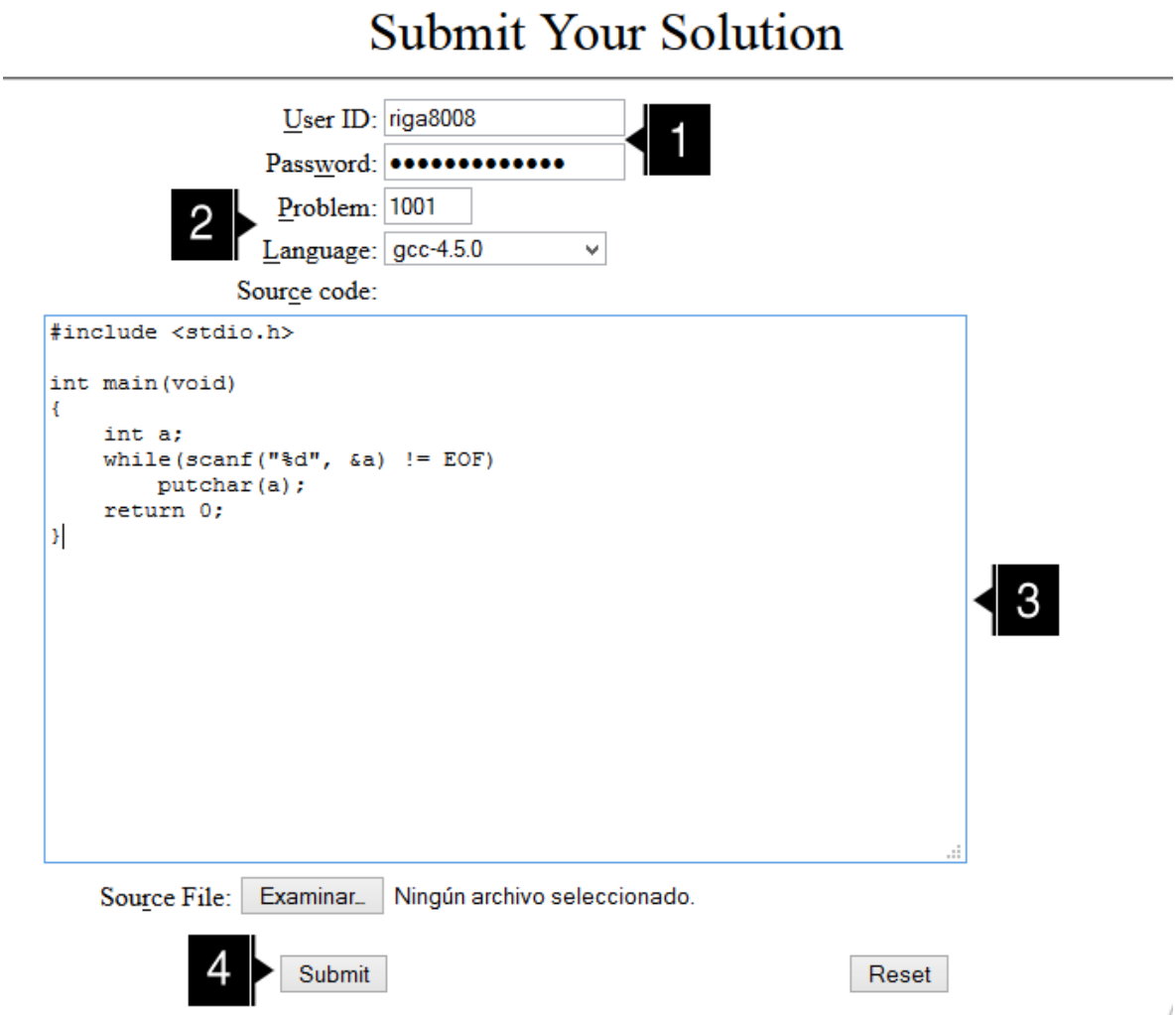


Figura 2.4 Para publicar una solución el usuario deberá proporcionar su información de usuario (1). En seguida tendrá que indicar el problema que desea resolver y el lenguaje de programación en el cual se encuentra codificada su implementación (2). El usuario podrá pegar su código fuente o bien elegir un archivo desde algún medio de almacenamiento (3) y finalmente publicar su solución (4) mediante el botón "Submit".

Judge Status	Prob.	Lang.	Code	Time	Memory	User
Accepted	1001	C	0.1K	0'00.00"	852K	riga8008

1 2 3

Figura 2.5 En la figura se muestra el resultado obtenido por un programa diseñado para solucionar el problema *1001. Hello World!*. El resultado obtenido *Accepted* (1) representa una solución satisfactoria, además es posible ver el tiempo consumido (2) por el programa y la memoria total en *kilobytes* (3).

Caribean Online Judge

http://coj.uci.cu/index.xhtml?lang=es

Descripción

Un usuario deberá ingresar al sistema, seleccionar algún problema existente, analizarlo e implementar una solución en alguno de los lenguajes de programación soportados. Posteriormente tendrá que publicar su propuesta y esperar los resultados de su evaluación.

Ventajas

- Interfaz de usuario sencilla, minimalista y fácil de usar.
- Ofrece un sistema completo de administración de usuarios, funcionalidades de comparación de resultados con otros usuarios y sugerencias de problemas a resolver basadas en perfiles.
- Ofrece soporte para gran variedad de lenguajes de programación.
- Permite la gestión de eventos en modalidades tipo concurso donde uno o más usuarios podrán poner a prueba sus capacidades bajo parámetros de tiempo y número de problemas.

Desventajas

- Su uso está fuertemente vinculado a concursos de programación de índole algorítmica y matemática, por lo cual no representa un buen sistema de práctica para programadores novatos.
- No ofrece ninguna herramienta de apoyo o soporte para instructores en sus labores de evaluación.

Figura 2.6 La pantalla de inicio de COJ Online Judge.

Figura 2.7 COJ ofrece administración de usuarios.

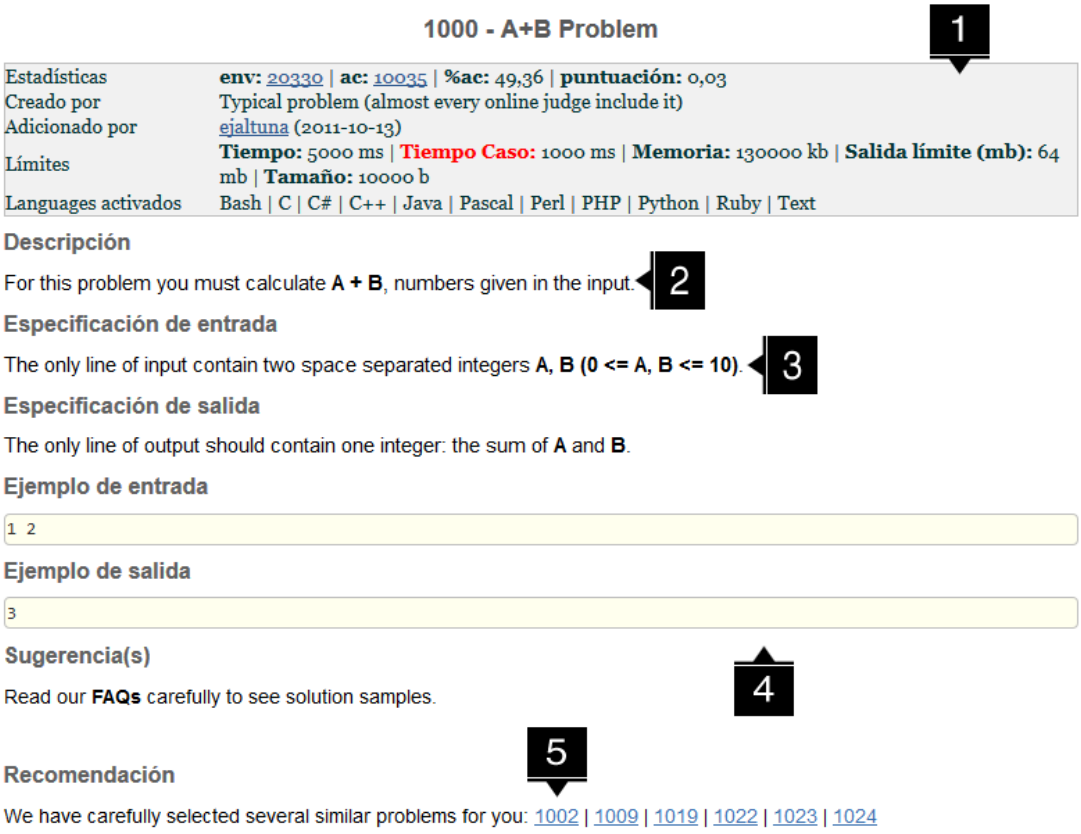


Figura 2.8 El problema “1000 – A + B Problem” presenta la estructura convencional de los problemas en los sistemas tipo ACM-ICPC. El título y criterios de evaluación se encuentra en la parte superior

(1), la descripción establece la problemática a resolver (2), las especificaciones de entrada y salida son detalles de los resultados esperados (3). Se ofrecen ejemplos de estas especificaciones mediante ejemplos (4) sencillos. Adicionalmente, COJ sugiere problemas con características similares (5) para fomentar la participación del alumno.

Figura 2.9 Los criterios de evaluación de COJ son idénticos a aquellos usados en TJU.

UVa Online Judge

http://uva.onlinejudge.org/

Descripción

Un usuario deberá ingresar al sistema, elegir un problema de entre las colecciones disponibles, analizarlo, implementar y publicar una solución para posteriormente esperar retroalimentación.

Ventajas

- Interfaz de usuario sencilla y minimalista.
- Ofrece completa administración de perfiles de usuarios.
- Permite la gestión de eventos en modalidades tipo concurso donde uno o más usuarios podrán poner a prueba sus capacidades bajo diversos parámetros.
- Ofrecen constantemente artículos de carácter científico para usuarios interesados de este tipo de temas.
- Permite descargar problemas en formato PDF para su consulta de manera *off-line*.

Desventajas

- No representa un buen sistema de práctica para programadores novatos.

- No ofrece ninguna herramienta de soporte para instructores en sus labores de evaluación.

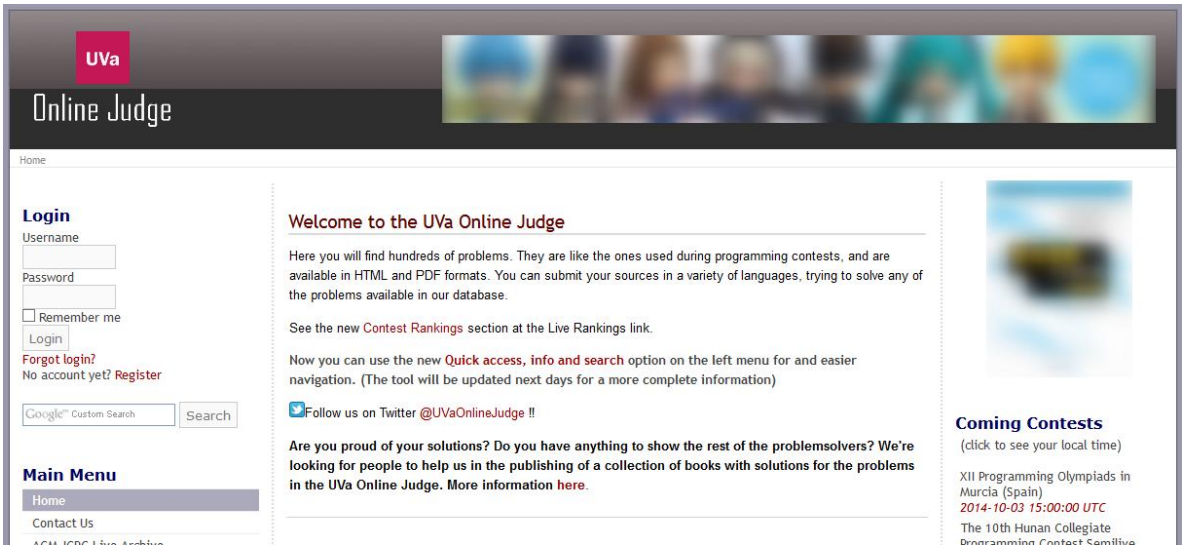


Figura 2.10 Página de inicio de *UVa Online Judge*.

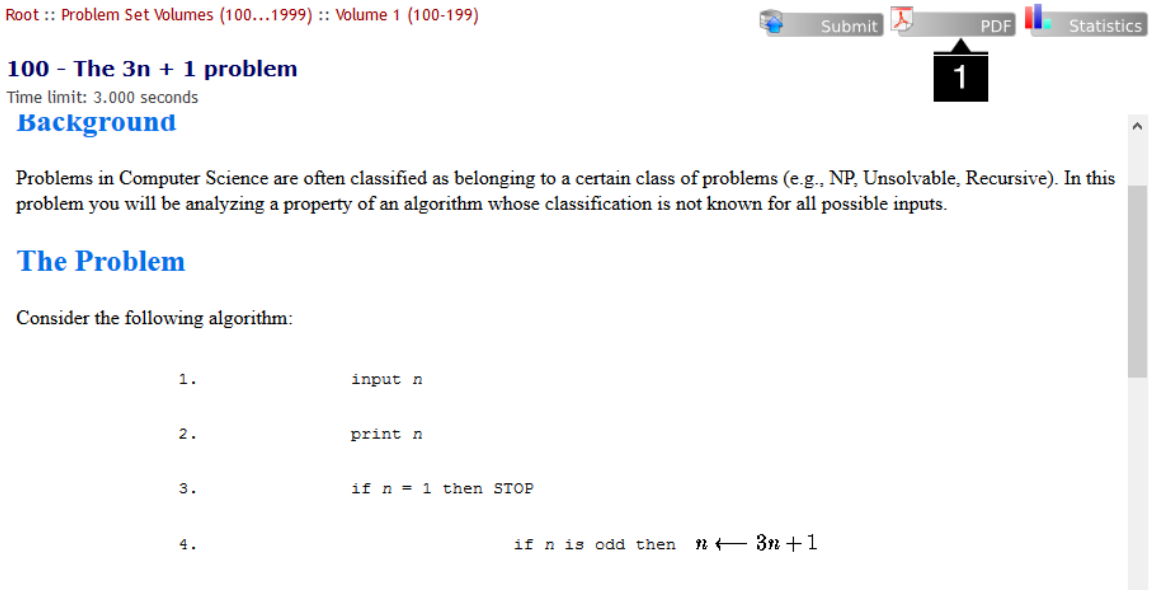
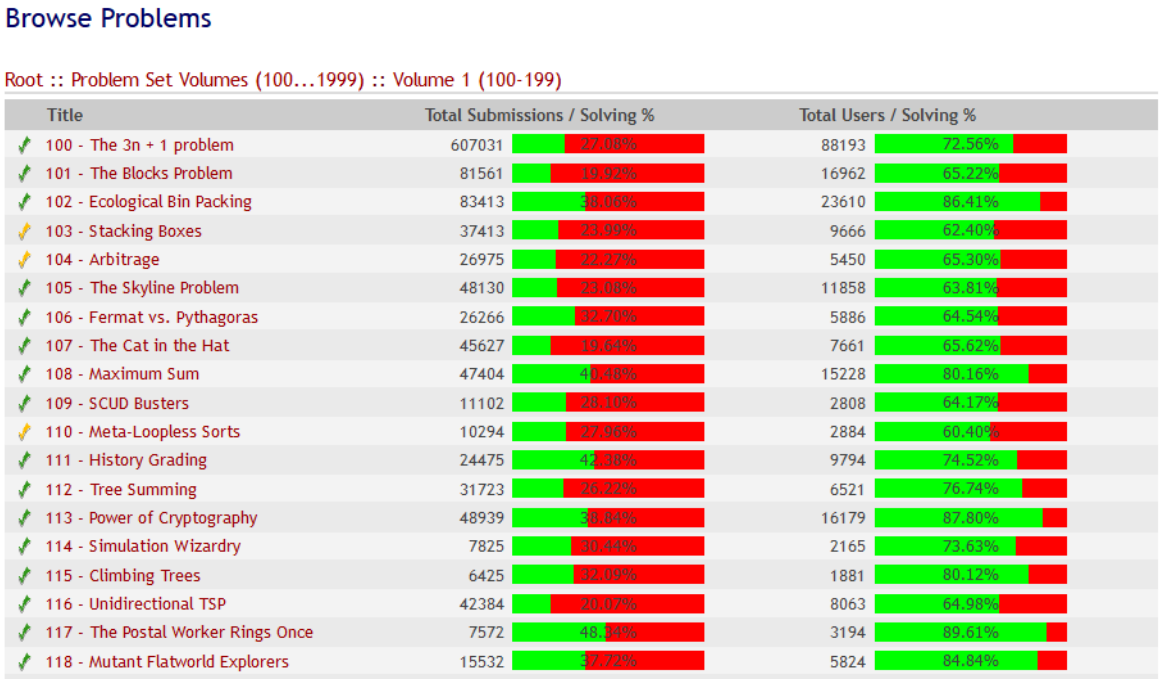


Figura 2.11 UVa Online Judge permite descargar sus problemas en formato PDF (1).



Browse Problems

Root :: Problem Set Volumes (100...1999) :: Volume 1 (100-199)

Title	Total Submissions / Solving %	Total Users / Solving %
100 - The 3n + 1 problem	607031 27.08%	88193 72.56%
101 - The Blocks Problem	81561 19.92%	16962 65.22%
102 - Ecological Bin Packing	83413 38.06%	23610 86.41%
103 - Stacking Boxes	37413 23.99%	9666 62.40%
104 - Arbitrage	26975 22.27%	5450 65.30%
105 - The Skyline Problem	48130 23.08%	11858 63.81%
106 - Fermat vs. Pythagoras	26266 32.70%	5886 64.54%
107 - The Cat in the Hat	45627 19.64%	7661 65.62%
108 - Maximum Sum	47404 40.48%	15228 80.16%
109 - SCUD Busters	11102 28.10%	2808 64.17%
110 - Meta-Loopless Sorts	10294 27.96%	2884 60.40%
111 - History Grading	24475 42.38%	9794 74.52%
112 - Tree Summing	31723 26.22%	6521 76.74%
113 - Power of Cryptography	48939 38.84%	16179 87.80%
114 - Simulation Wizardry	7825 30.44%	2165 73.63%
115 - Climbing Trees	6425 32.09%	1881 80.12%
116 - Unidirectional TSP	42384 20.07%	8063 64.98%
117 - The Postal Worker Rings Once	7572 48.34%	3194 89.61%
118 - Mutant Flatworld Explorers	15532 37.72%	5824 84.84%

Figura 2.12 Estadísticas del problema 100 de Uva Online Judge. Se muestra la relación entre la cantidad de veces que cada problema ha intentado ser resuelto y el número de soluciones con valoración positiva.

Sphere Online Judge

http://www.spoj.com/

Descripción

El usuario deberá ingresar al sistema, elegir un problema, analizarlo e implementar una solución para esperar sus resultados.

Este sistema sobresale debido la gran cantidad de problemas descritos en lenguaje español. Soporta una amplia variedad de lenguajes de programación de propósito general. Además, permite a usuarios pertenecientes al rol *Problem Setter* diseñar problemas de una manera muy sencilla.

Una característica que es importante destacar es que, cuando un *Problem Setter* se encuentra editando un problema puede activar una casilla de verificación “Include new languages” lo cual sugiere la flexibilidad en sus mecanismos de evaluación.

Ventajas

- Ofrece una interfaz de usuario sencilla y minimalista.
- Permite la gestión de eventos en modalidades tipo concurso donde uno o más usuarios podrán poner a prueba sus capacidades bajo parámetros de tiempo y número de problemas.
- Ofrece soporte para gran cantidad de lenguajes de programación.

- Posee documentación detallada en donde se describen las funcionalidades ofrecidas a los roles existentes.
- Ofrece una gran cantidad de problemas descritos en lenguaje español.

Desventajas

- No representa un buen sistema de retroalimentación para programadores novatos. Aunque es importante mencionar que la redacción de sus problemas tiende a ser clara y concisa.
- No ofrece ninguna función que auxilie a instructores en sus labores de evaluación.

Figura 2.13 Página de inicio de Sphere Online Judge.

Figura 2.14 Los eventos permiten a sus participantes competir entre ellos para medir sus capacidades.

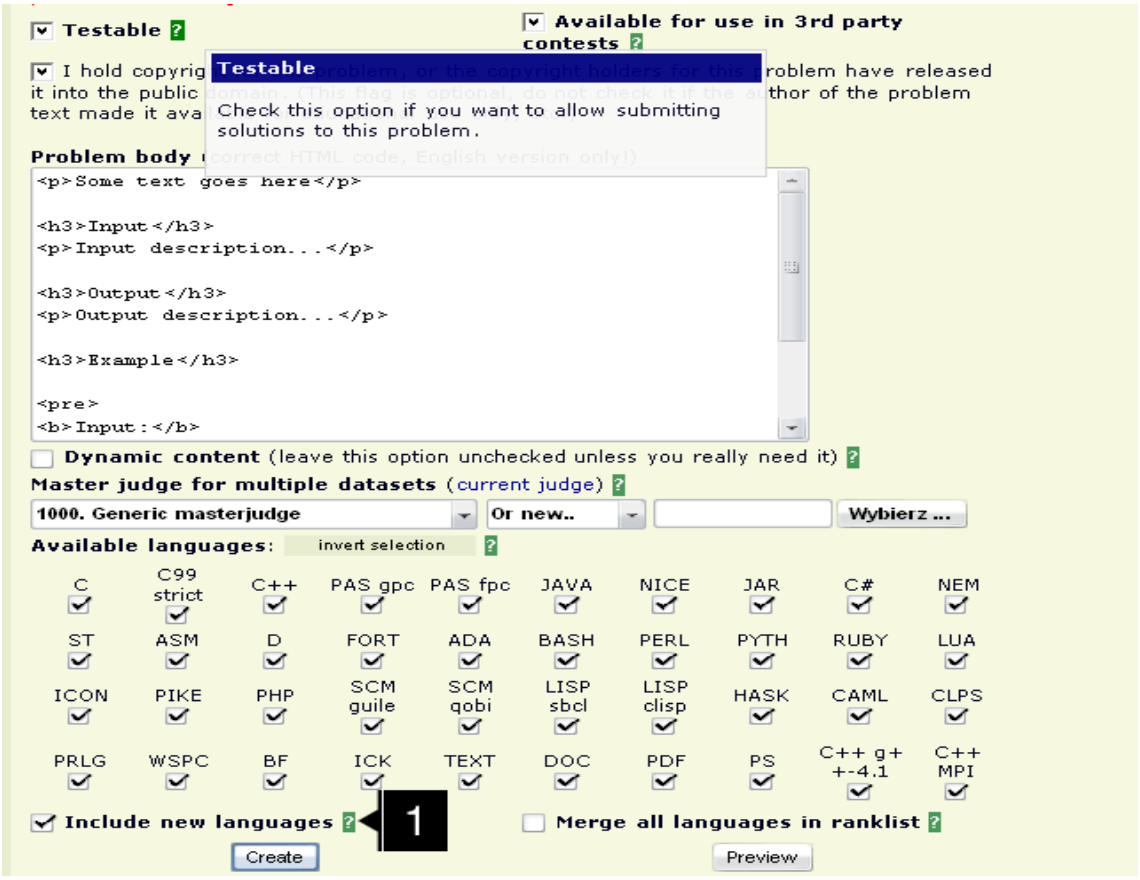


Figura 2.15 En la edición de problemas es posible elegir la opción *Include new languages*. Esto sugiere extensibilidad del sistema en sus mecanismos de evaluación (1).

Existen muchos otros sistemas cuyas mecánicas de participación son idénticas a aquellas que han sido descritas hasta este punto, difiriendo en características sin valores relevantes para el estudio del campo del arte. El siguiente listado presenta el resto de los sistemas que fueron analizados:

Peking University Online Judge
http://poj.org/

Zhenjian University Online Judge
http://acm.zju.edu.cn/

Teddy Online Judge
https://www.teddyonlinejudge.net/

Timus Online Judge
https://acm.timus.ru

BNU Online Judge
http://www.bnuoj.com/v3/

CodeForces
http://codeforces.com/

Light Online Judge
http://lightoj.com/login_main.php

Otro tipo de sistemas

Existen otras plataformas con metodologías de evaluación y aprendizaje diferentes a aquellas que han sido descritas con anterioridad.

TopCoder
http://community.topcoder.com/tc

Descripción

Este sistema, al menos por sus mecánicas de evaluación, podría ser catalogado como un sistema de tipo ACM-ICPC, sin embargo posee características que merecen especial atención.

TopCoder es una plataforma en línea formada por una inmensa comunidad de expertos en tecnologías de la información, caracterizada por organizar competencias de algoritmia y *ethical hacking* a nivel internacional.

Destacada por canalizar a los mejores desarrolladores y expertos en seguridad con empresas de renombre internacional como Intel, Facebook, PayPal o incluso organizaciones gubernamentales como la NSA de los Estados Unidos de América. Además, ofrece a sus usuarios la posibilidad de obtener remuneración económica de acuerdo su desempeño.

Para poder participar será necesario registrarse en el sitio para acceder a un amplio tutorial relacionado al funcionamiento del sistema. Comprendidas las formas de participación, el usuario elegirá la modalidad de práctica que más le interese: algoritmia o seguridad. La primera modalidad necesitará del *Java Runtime Environment* para poder acceder a las 'salas de práctica'.

Ya dentro de alguna sala, el usuario podrá formar parte de algunas competencias intentando resolver los problemas que las componen. Los problemas están clasificados por puntos y dificultad, por lo que es recomendable comenzar con problemas sencillos particularmente si no se posee noción alguna con respecto a este tipo de competencias.

Ventajas

- Los problemas poseen una ponderación basada en su dificultad.
- Ofrece soporte para diversos lenguajes de programación.
- Ofrece una gran cantidad de recursos de aprendizaje.
- Permite a sus usuarios vincularse con empresas de renombre internacional y obtener remuneraciones económicas.

Desventajas

- Requiere tener instalado el *Java Runtime Environment*
- La interfaz de usuario es poco intuitiva, con un esquema de colores poco agradable.
- Se requiere disponer de fuertes conocimientos en programación y resolución de problemas.
- No ofrece ninguna funcionalidad de soporte para instructores en sus labores de evaluación.

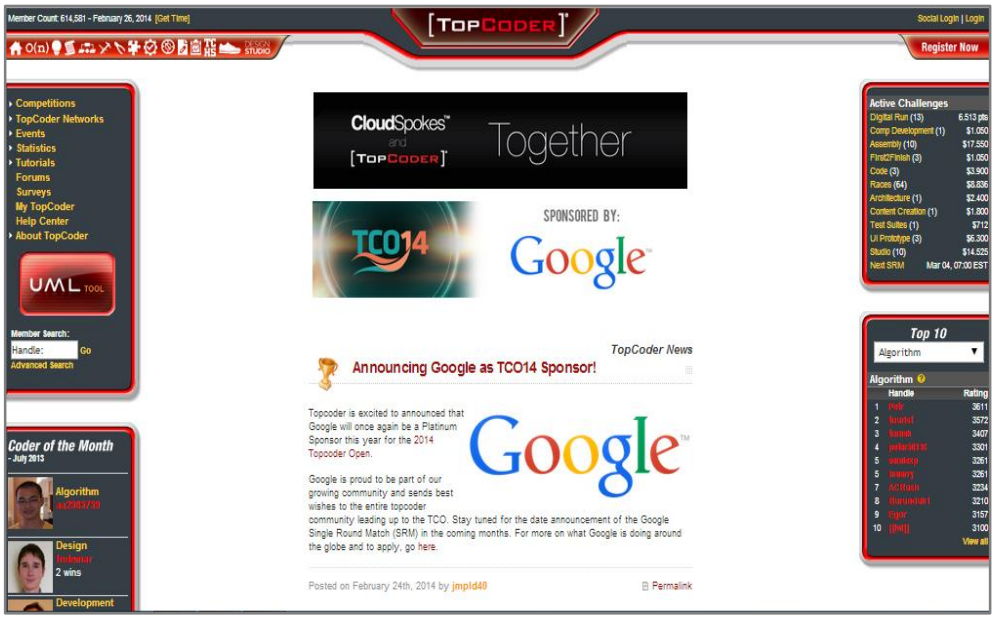


Figura 2.16 Página de inicio de TopCoder.

Figura 2.17 TopCoder es inicialmente poco intuitivo.

Codecademy

http://www.codecademy.com/

Descripción

Codecademy es un excelente sitio de aprendizaje en línea que sirve como tutor virtual para aquellas personas que estén interesadas en aprender de manera dinámica e interactiva, lenguajes de programación orientados al desarrollo web, lenguajes de marcado y algunos lenguajes de propósito general. A diferencia de la mayoría sistemas de evaluación mencionados con anterioridad, la dinámica de aprendizaje se desarrolla completamente en el sistema mediante el uso de una consola interactiva que ayudará, entre otras cosas a entender la sintaxis del lenguaje de programación en el que se esté interesado.

Ventajas

- Interfaz de usuario simple y fácil de usar.
- Dinamismo en su metodología de enseñanza mediante cursos interactivos, retos y reconocimientos en función del desempeño fomentando la participación constante de sus usuarios.
- Completa administración de perfiles y gestión de sesiones.
- Incorpora dinamismo en su metodología de evaluación al ofrecer una consola que permite al usuario practicar directamente en la aplicación web. Gracias a esto, el usuario aprende de manera interactiva los detalles relacionados a la sintaxis del lenguaje que intenta aprender.

Desventajas

- Soporte para un número limitado de lenguajes de programación.
- Algunos ejercicios son evaluados de manera errónea.

Figura 2.18 Página de inicio de CodeAcademy.

Figura 2.19 Codeacademy permite a sus usuarios realizar proyectos para reforzar sus conocimientos.

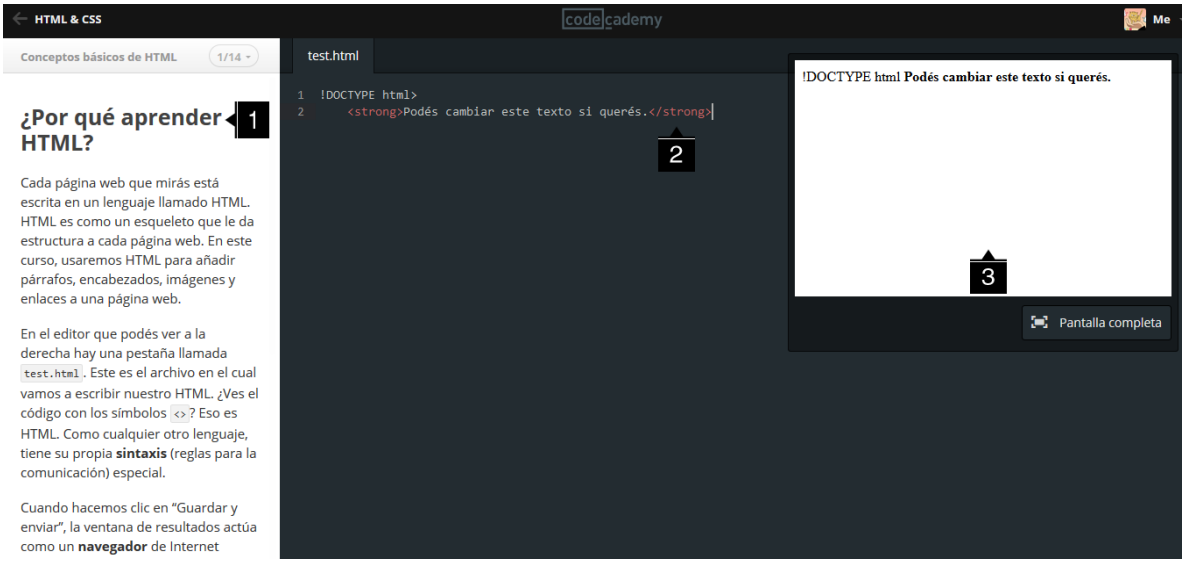


Figura 2.20 En la figura se presenta el mecanismo de interacción entre un usuario y el sistema. A izquierda se localizan las instrucciones que el usuario deberá seguir para dar solución a los retos planteados (1). Al centro se localiza un editor de texto que irá orientando al usuario mientras este vaya codificando (2). A la derecha de la pantalla de muestra la consola de salida del sistema (3) donde se verán reflejados los resultados del código fuente una vez que este ha sido interpretado.

Todos estos sistemas aportan un gran valor a las problemáticas competentes a este trabajo, así que elegir sus fortalezas y combinarlas en beneficio del enriquecimiento de las funcionalidades del sistema emergente es una labor de gran importancia.

Los sistemas tipo ACM-ICPC (incluyendo a TopCoder) sugieren una metodología sencilla, compuesta por una lógica de participación que resulta fácil de implementar y de entender para usuarios con conocimientos básicos en informática. Sus criterios de evaluación, ofrecen los valores cualitativos buscados en términos de rendimiento, así que se optará por adaptar un modelo de retroalimentación similar.

Finalmente, CodeAcademy destaca por las funcionalidades presentadas a sus usuarios, cuidando al máximo tanto sus detalles estéticos como técnicos así que se buscará identificar claramente qué características pueden resultar enriquecedoras.

Capítulo 3 . Análisis de requisitos

3.1 Observación

La metodología de evaluación existente no difiere mucho de cualquier otra metodología en niveles medio o medio superior de cualquier otra asignatura. Está conformada por la implementación de exámenes escritos y prácticas de laboratorio.

Por lo general, un examen está compuesto en las dos secciones:

Evaluación de conceptos. Se evalúan los fundamentos teóricos del estudiante mediante preguntas abiertas, de opción múltiple o de relación.

Resolución de ejercicios. Se pretende evaluar el desempeño de un estudiante quien deberá implementar de manera escrita el código fuente de un programa. Comúnmente, se pide al estudiante que realice 'corridas de escritorio' para verificar la funcionalidad de su implementación.

Aunque son sencillos implementar, este tipo de enfoques presentan inconvenientes importantes que se deben señalar:

- **Deficiencias en sus resultados.** Los exámenes escritos no garantizan en absoluto que el código escrito por el estudiante funcione de acuerdo a los requisitos del instructor. Son en realidad una vaga aproximación de lo que "se supone" el programa debe realizar.
- **Eficiencia limitada.** Verificar que el código fuente del estudiante sea aceptable, revisando la sintáxis línea por línea y comprobando sus resultados mediante corridas de escritorio resulta ineficiente a largo plazo, sin mencionar lo agobiante del proceso.

Para intentar extender los alcances del enfoque descrito anteriormente, se somete al estudiante a realizar ejercicios en laboratorios de cómputo donde deberán verificar que sus implementaciones funcionen correctamente. Aunque este enfoque ofrece mejores resultados también presenta algunos inconvenientes:

- **Autosuficiencia no garantizada**. No ofrece ninguna garantía de que el estudiante haya implementado por sí mismo la solución para el problema en contexto. Es muy común ver que los estudiantes hagan uso de buscadores en Internet para poder encontrar una solución ya codificadas o incluso que compartan entre ellos sus propias soluciones.
- **Dependencia sobre el elemento humano**. El instructor deberá evaluar manualmente el programa y corroborar sus resultados uno a uno lo cual tiende a ser una tarea bastante.
- **Incompatibilidad**. En ocasiones, la incompatibilidad de entre sistemas operativos o diferentes versiones de compiladores provocan conflictos en el momento de generar un programa a partir del código fuente. Esto obliga al instructor a mantener diferentes versiones

de un compilador o tener que editar el código fuente buscando corregir los detalles que han generado errores.

En el Apéndice A, adjunto a este documento, se presentan los resultados de una encuesta realizada a estudiantes que permite conocer su perspectiva sobre las metodologías de evaluación usadas generalmente en materias de programación estructurada.

3.2 Entrevistas

El proceso de desarrollo de software se inicia a través de una entrevista con el principal rol involucrado; el instructor, con la finalidad de delimitar los alcances del proyecto mediante una plática abierta.

Las preguntas realizadas al instructor se clasifican en las categorías descritas a continuación.

Categoría	Descripción
Preguntas abiertas	Son preguntas que permiten al analista involucrarse a fondo en contexto del problema. Ofrecen al usuario la oportunidad de expresarse libremente. Su importancia radica en eliminar ideas preconcebidas por parte del analista con los procesos existentes.
Preguntas de control	Este tipo del preguntas permiten acotar las dimensiones de problema. Son preguntas que requieran respuestas concretas
Pregunta de confirmación	Una vez acotado el problema, es necesario confirmar que las necesidades del cliente han sido plasmadas de manera adecuada en los requisitos

Tabla 3.1 Descripción de las categorías de las preguntas.

Algunas de las preguntas realizadas en la plática mencionada anteriormente se presentan en la siguiente tabla.

Categoría	Pregunta	Objetivo
Pregunta abierta	Descríbame con sus propias palabras y manera precisa las actividades que realiza para aplicar un examen a un estudiante en sus materias	Permitir al usuario del sistema introducir al analista al contexto de la problemática actual
Pregunta de control	En concreto ¿cuáles de las actividades que me mencionaste en la pregunta te gustaría poder automatizar?	Permitir delimitar los alcances del proyecto conociendo cuáles son las necesidades del usuario. El analista deberá valorar estas necesidades y evaluar si es o no factible integrarlas dentro del proyecto.
Pregunta de control	¿Cuáles son los criterios que usa para evaluar los programas de tus estudiantes?	Permitir delimitar los alcances del proyecto conociendo cuáles son las necesidades del usuario. El analista deberá valorar estas necesidades y evaluar si es o no factible integrarlas dentro del proyecto
Pregunta de control	¿Existe algún otro criterio que te gustaría poder incluir y que en el actual mecanismo de evaluación no posee?	Permitir delimitar los alcances del proyecto conociendo cuáles son las necesidades del usuario. El analista deberá valorar estas necesidades y evaluar si es o no factible integrarlas dentro del proyecto.
Pregunta de control	¿Cuáles son los criterios que usas para permitir a un estudiantes aplicar examen o práctica?	Permitir delimitar los alcances del proyecto conociendo cuáles son las necesidades del usuario. El analista deberá valorar estas necesidades y evaluar si es o no factible integrarlas dentro del proyecto.

Tabla 3.2 Primeras preguntas

3.3 Diagrama de actividades

Los procesos de evaluación actuales son representados en los siguientes diagramas de actividades.

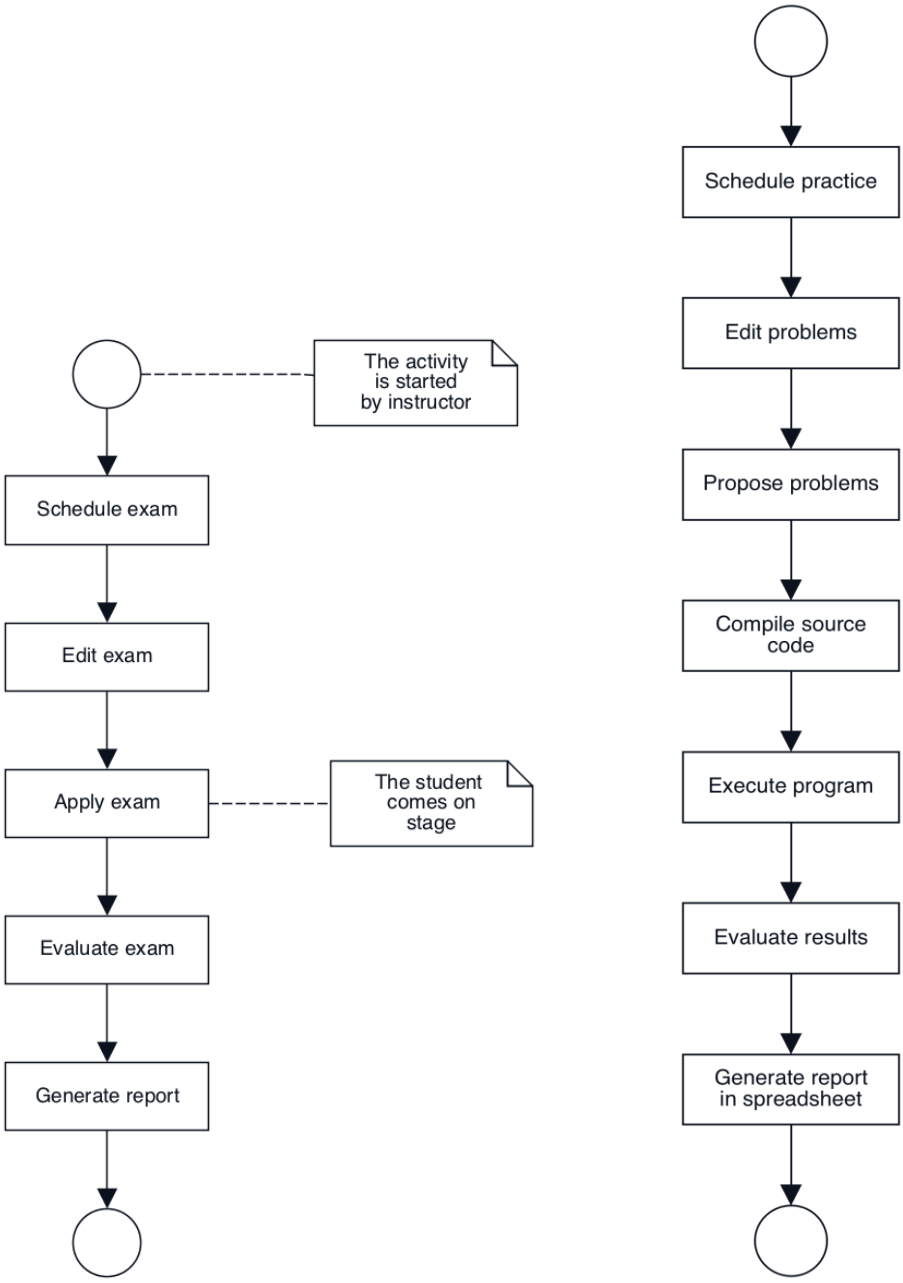


Figura 3.1 Proceso de evaluación existente.

3.4 Especificaciones del sistema

De acuerdo al análisis de los procesos de evaluación existentes y las entrevistas realizadas a los roles competentes se definen los siguientes listados con las especificaciones para cada rol.

Especificaciones para el rol instructor

- Se desea automatizar el proceso de evaluación del desempeño de los estudiantes frente a problemáticas que representen situaciones reales basándose en el rendimiento de sus programas.
- La resolución de problemas se llevará a cabo en función de dos tipos de eventos: prácticas y exámenes. Cada tipo de evento varía en función del periodo de duración del evento y número de problemas.
- Se deber permitir obtener información detallada respecto al desempeño de los estudiantes por alumno y por evento.
- Se debe tratar de determinar si el estudiante ha hecho uso de un programa ya implementado en internet.
- Se debe tratar de determinar si dos estudiantes han hecho uso del mismo código fuente para resolver un problema en particular.

Especificaciones para el rol estudiante

- El sistema debe ofrecer una dinámica sencilla donde el estudiante solo tenga que preocuparse por codificar su solución sin atender detalles de implementación ajenos a la resolución de problemas.
- El sistema debe ofrecer al alumno puntos clave que le ayuden a entender cómo mejorar el desempeño de su implementación.
- Se debe ofrecer de manera recurrente contenido que refuerce los conocimientos que este haya adquirido en clases, internet, libros u otros medios de información.

3.5 Identificación de las tareas e historias de usuario.

Mediante las labores de observación y en función de las entrevistas hechas a los roles involucrados es posible definir los escenarios en los que estos se desenvuelven.

Historias de usuario

El objetivo de las entrevistas consiste en tratar de delimitar los alcances del proyecto y realizar las especificaciones de los roles pertinentes. Una eficaz manera de documentar estos requisitos es mediante el uso de historias de usuario descritas en el Capítulo 1. Las siguientes tablas representan un modelo esquemático estas historias.

Número	Nombre
1	Evaluar código publicado
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Me gustaría que el sistema me permita evaluar el código fuente de mis estudiantes validando sus resultados con valores predefinidos por mí, ofreciéndoles retroalimentación de manera automática. Se deberá compilar y generar ejecutables sin necesidad de mi intervención	
Criterios de aceptación	
> Que el código fuente publicado por mis estudiantes sea compilable y una vez compilado ejecutado sea finito > Que el código fuente publicado no sea mayor a 2 MB > Que la retroalimentación generada indique tiempo consumido, calificación, memoria consumida y número de respuestas correctas	

Tabla 3.3 Evaluar código fuente de los estudiantes

Número	Nombre
2	Gestión de problemas
Prioridad	**Observaciones**
Alta	
Descripción	
Quiero poder publicar problemas y ejercicios en el sistema. Publicaré la descripción del problema a resolver, los criterios que deseo evaluar, el número de respuestas parciales que considero para que la respuesta sea la que yo espero	
Criterios de aceptación	
> Un problema deberá de manera obligatoria contener un título,descripción, cuerpo y un conjunto de casos de prueba > El cuerpo permitirá en su estructura cualquier contenido HTML > Para que un problema pueda formar parte de un evento su estructura deberá estar completa	

Tabla 3.4 Gestión de problemas

Número	**Nombre**
3	Edición de problemas
Prioridad	**Observaciones**
Alta	
Descripción	
El sistema deberá permitir actualizar un la estructura de un problema una vez que este ha sido creado	
Criterios de aceptación	
> Un problema solo podrá ser editado si el evento al que pertenece no ha iniciado > Un problema solo podrá ser editado si el problema no ha sido cancelado	

Tabla 3.5 Gestión de problemas

Número	**Nombre**
4	Eliminación de problemas
Prioridad	**Dependencias**
Alta	
Descripción	
Quiero poder eliminar problemas de un evento cuando este no ha iniciado. Si el evento ya inicio quiero cambiar su estado para evitar confundir a mis estudiantes durante un evento	
Criterios de aceptación	
> Un problema solo podrá ser eliminado si el evento al que pertenece no ha iniciado	

Tabla 3.6 Eliminación de problemas

Número	**Nombre**
5	Gestión de eventos
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Requiero gestionar eventos de tipo examen y práctica que agrupen los problemas que deseo generar	
Criterios de aceptación	
> Un evento deberá tener una fecha inicial y una fecha final > Un examen no puede tener más de cinco problemas > Una práctica puede tener cualquier número de problemas > Para que un estudiante pueda participar en un evento debe confirmar su participación	

Tabla 3.7 Gestión de eventos

Número	Nombre
6	Edición de eventos
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Deseo poder modificar la información de los eventos que genere una vez que los he creado siempre y cuando estos se encuentren en proceso	
Criterios de aceptación	
> Si el evento aún no ha comenzado debo poder modificar sus fechas iniciales > Si el evento se encuentra en línea pero no ha iniciado y todos los problemas que le corresponden han sido eliminados entonces el evento cambiará su estado a off-line > Si el evento han iniciado este no podrá ser editado	

Tabla 3.8 Gestión de eventos

Número	Nombre
7	Cancelación de eventos
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Se requiere poder cancelar un evento cuando sea necesario y notificar a los estudiantes que ya han sido invitados con respecto a esta operación	
Criterios de aceptación	
> Un evento podrá ser cancelado siempre y cuando este no se encuentre en proceso > Si un evento se cancela se deberá notificar a los participantes	

Tabla 3.9 Gestión de eventos

Número	Nombre
8	Consultar resultados
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Requiero poder consultar los resultados de mis estudiantes que han participado en un evento mediante un reporte o resume	
Criterios de aceptación	
> La consulta de resultados podrá darse por evento cuando este haya terminado > La consulta de resultados también podrá darse por estudiante > El seguimiento de los resultados deberá darse en tiempo real	

Tabla 3.10 Gestión de eventos

Número	Nombre
9	Gestión de grupos
Prioridad	**Observaciones**
Media	Gestión involucra las operaciones crear, eliminar, actualizar...
Descripción	
Me gustaría poder generar grupos a los cuáles mis estudiantes podrán unirse o incluso yo podré agregar. Estos grupos me permitirán organizar mis eventos y publicaciones	
Criterios de aceptación	
> Un estudiante solo podrá unirse a un grupo mediante una clave o número de registro	

Tabla 3.11 Gestión de eventos

Número	Nombre
10	Publicación de contenido
Prioridad	**Observaciones**
Baja	Ninguna
Descripción	
Quiero poder compartir contenido multimedia con mis grupos. Esto será a través de una mecánica similar a los blogs convencionales	
Criterios de aceptación	
> El contenido solo será visible para los grupos que yo indique	

Tabla 3.12 Gestión de eventos

Número	Nombre
11	Verificar autenticidad de los programas
Prioridad	**Observaciones**
Baja	Ninguna
Descripción	
Quiero poder determinar sin dos o más códigos fuentes publicados durante un evento fueron codificados por la misma persona pero publicados en diferentes perfiles. Al finalizar el evento el sistema deberá advertirme sobre códigos "sospechosos" e indicarme cuáles fueron los estudiantes y sus respectivas soluciones que lanzaron la advertencia	
Criterios de aceptación	
> El evento debe haber concluido	

Tabla 3.13 Gestión de eventos

Número	Nombre
12	Generar reporte
Prioridad	**Observaciones**
Alta	Ninguna
Descripción	
Se requiere permitir a los estudiantes consultar los resultados de su participación en un evento cuando este haya concluido	
Criterios de aceptación	
> El evento debe haber concluido	

Tabla 3.14 Gestión de eventos

De las historias de usuario se desprende la siguiente pila de requisitos. Esta herramienta de carácter organizativo ofrece una manera estimar los costos de desarrollo a partir de la prioridad de cada historia de usuario.

Pila de requisitos	
Historia de usuario	**Prioridad**
1	Alta
2	Alta
3	Alta
4	Alta
5	Alta
6	Alta
7	Alta
8	Alta
9	Media
10	Baja
11	Baja
12	Alta

Tabla 3.15 La pila de requisitos.

Tareas

Las historias de usuario definidas en el apartado anterior permiten establecer aquellas tareas que representan una funcionalidad del sistema.

Discriminante de tipo para las tareas

La discriminante de tipo permite indicar al desarrollador aquellas tereas que necesitas ser implementadas. Las tareas se encuentran clasificadas de acuerdo al rol que la desencadena y son descritas en tablas posteriores.

Nombre	Descripción	Abreviatura
Automatic	Deberá ser iniciada por un usuario pero completarse de manera automática	A
Interactive	Deberá permitir la interacción entre el usuario y el sistema	I
Manual	Se encuentra fuera del dominio del sistema pero es necesaria para poder iniciar algún proceso o escenario	M

Tabla 3.16 Discriminantes de tipo

Tareas de propósito general

Estás tareas ofrecen las mismas funciones para cualquier rol que las desencadena.

Registrarse en el sistema					
#	Nombre	Predecesor	Descripción	Justificación	Tipo
1	Completar formulario de registro	-	El usuario accede a la página y elige la acción Registrarme.	El usuario deberá hacerse con una identidad dentro del sistema	I
2	Enviar notificación por correo electrónico	1	El sistema deberá enviar un correo electrónico solicitando al usuario confirmar su cuenta de usuario en el sistema.	Para que un usuario pueda acceder al sistema deberá poder verificar su identidad mediante correo electrónico	A
3	Confirmar registro en el sistema	2	El usuario tendrá que acceder a la cuenta de correo electrónico que proporcionó y confirmar su registro en el sistema	Confirmar su identidad a través de un vínculo su participación dentro del sistema	M

Tabla 3.17 Registrarse en el sistema

Ingresar al sistema					
#	Nombre	Predecesor	Descripción	Justificación	Tipo
1	Completar formulario de registro	-	El usuario accede al sistema y elige la acción Ingresar	El usuario deberá ingresar sus credenciales para poder acceder al sistema	I
2	Mostrar panel de tareas y actividad reciente	1	Una vez que el instructor se ha autenticado correctamente el sistema deberá mostrar contenido e información de interés como notificaciones, resultados de eventos pasados, entre otras cosas	El sistema deberá mostrará contenido de interés al usuario en función de su rol	A

Tabla 3.18 Ingresar al sistema

Recuperar contraseña					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Completar formulario de registro	-	El usuario accede al sistema y elige la acción Recuperar contraseña.	Se debe elegir la operación adecuada para la acción que se desea realizar	I
2	Enviar notificación por correo electrónico	1	El sistema deberá enviar un correo electrónico solicitando al usuario confirmar la recuperación de su contraseña mediante un vínculo	Enviar un correo electrónico servirá para verificar que sólo el dueño de la cuenta de usuario podrá recuperar	A
3	Confirmar registro en el sistema	2	El usuario tendrá que acceder a la cuenta de correo electrónico que proporcionó y confirmar la acción solicitada haciendo click en el vínculo contenido dentro del cuerpo del mensaje de correo electrónico		M
4	Redirigir a la vista "Recuperar contraseña"	3	El correo deberá redirigir al usuario una pantalla del sistema en dónde tendrá que proporcionar nuevamente su contraseña	Es necesario redirigir al usuario a la vista adecuada	A
5	Elegir una nueva contraseña	4	El usuario deberá elegir una nueva contraseña y guardar los cambios en su perfil	Se requiere actualizar la información existente para evitar que la cuenta de usuario quede comprometida	I

Tabla 3.19 Recuperar contraseña

Actualizar información					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir tarea solicitada	-	El usuario accede al sistema, se autentica y elige la tarea actualizar información	El usuario deberá elegir la tarea adecuada antes actualizar su información	I
2	Completar formulario	1	El usuario deberá completar el formulario que aparecerá en pantalla	Modificar la información existente es la parte fundamental de esta tarea	I
3	Confirmar actualización	2	El usuario tendrá que confirmar la actualización de la información.	Sólo se permitirá actualizar la información cuando el usuario verifique que sus datos son correctos	I
4	Actualización de la información	3	El sistema deberá actualizar la información en la base de datos y redirigir al usuario la página principal	La información actualizada deberá almacenarse de forma permanente o hasta que el usuario decida actualizarla nuevamente	A

Tabla 3.20 Actualizar información

Tareas vinculadas al rol instructor

Las siguientes tablas describen las tareas iniciadas por el instructor.

Generar evento					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Generar evento	-	El instructor genera un evento eligiendo la opción de su panel de tareas	El instructor deberá elegir la tarea adecuada	I
2	Elegir tipo de evento	1	El instructor elige el tipo de evento. Estás puede ser dos modalidades: práctica o examen	Elegir el tipo de evento permitirá al sistema construir la mecánica adecuada para cada tipo de evento	I
3	Elegir fecha y hora de inicio y culminación.	2	El instructor elegirá la fecha inicial y/o final del evento dependiendo de la modalidad	Es necesario establecer la duración del evento	I
4	Redactar problemas.	3	Una vez establecida la duración del evento el instructor podrá comenzar a redactar problemas. El proceso de redacción se describe en una tabla independiente	El instructor deberá agregar problemas al evento para que este pueda iniciar	I
5	Enviar invitaciones	4	Una vez que el instructor se encuentra satisfecho con la estructura del evento tendrá la oportunidad de notificar a sus estudiantes una invitaciónpara que participen.	Se notificará a los estudiantes a través de diferentes mecanismos sobre el evento recientemente creado	I
6	Notificar estudiantes	5	El sistema generará una notificación al estudiante proporcionándole un código de acceso al evento. Este código será único e intransferible	Se deberá notificar a los estudiantes vía correo electrónico sobre el evento recientemente creado.	A
7	Publicar evento	6	El instructor deberá confirmar la publicación del evento antes de cambiar su estado a "En línea"	El evento no puede iniciar si su estado no se lo permite	I

Tabla 3.21 Generar evento

Cancelar evento					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir evento	-	El instructor elige el evento que desea cancelar.	Se deberá indicar el evento que se desea cancelar	I
2	Elegir tarea requerida	1	El instructor selecciona la tarea "Cancelar evento" que desencadena la acción solicitada	Se deberá indicar la tarea que se desea realizar	I
3	Solicitar confirmación	2	El sistema responde al instructor pidiendo confirmación para completar la acción solicitada	Las acciones críticas requieren confirmación del usuario	I
4	Confirmar acción solicitada	3	El instructor deberá confirmar la acción solicitada.	Las acciones críticas requieren confirmación del usuario	I
5	Guardar cambios	4	Si la acción es confirmada por el instructor, el sistema elimina el evento seleccionado	Los cambios generados debe ser persistentes y verse reflejados en el sistema	A

Tabla 3.22 Cancelar evento

Actualizar evento					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir evento	-	El instructor elige el evento que desea editar.	Se debe elegir el evento que se desea editar	I
2	Elegir tarea requerida	1	El instructor selecciona la tarea "Actualizar evento" que desencadena la acción solicitada	Se debe elegir la tarea adecuada para actualizar la información del evento	I
3	Completar formulario	2	El instructor completa el formulario que se muestra en pantalla y selecciona la operación "Guardar"	Actualizar la información existente es la parte fundamental de esta tarea	I
4	Solicitar confirmación	3	El sistema responde al instructor solicitando confirmación para completar la acción solicitada.	Las operaciones críticas requieren confirmación por parte del usuario	I
5	Confirmar acción solicitada	4	El instructor deberá confirmar la acción solicitada	Las operaciones críticas requieren confirmación por parte del usuario	I
6	Guardar cambios	5	Si la acción es confirmada por el instructor, el sistema actualiza el evento seleccionado	Los cambios generados debe ser persistentes y verse reflejados en el sistema	A

Tabla 3.23 Actualizar evento

Buscar evento					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir tarea requerida	-	El instructor selecciona la tarea "Buscar evento" que desencadena la acción solicitada	El instructor deberá elegir la tarea adecuada	I
2	Ingresar texto de búsqueda	1	El instructor completa el formulario de búsqueda que se muestra en la vista	Se deberá ingresar un criterio de búsqueda para delimitar la cantidad de información que se desea obtener	I
3	Buscar eventos	2	El sistema busca todos los eventos que coincidan con el criterio de búsqueda	Es la parte fundamental de la tarea	A
4	Mostrar resultados	3	Se actualizar la vista mostrando los resultados obtenidos de la consulta	Es necesario mostrar los resultados obtenidos que coincidan con el criterio de búsqueda	A

Tabla 3.24 Buscar evento

Generar problema					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Generar problema	-	El instructor genera un problema eligiendo la opción en el panel de tareas	Se requiere elegir la tarea a adecuada	I
2	Completar formulario	1	El instructor deberá completar un formulario con los datos correspondientes al problema que ha diseñado	Ingresar la información apropiada es requisito para verificar la integridad del problema	I
3	Guardar cambios	2	El instructor podrá guardar el problema una vez que se encuentre satisfecho con su estructura. El estado del problema es "Creado" cuando es guardado	Los cambios generados debe ser persistentes y verse reflejados en el sistema	I

Tabla 3.25 Generar problema

Eliminar problema					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir problema	-	El instructor elige el problema que desea eliminar	Se debe indicar el problema a eliminar	I
2	Elegir tarea requerida	1	El instructor selecciona la tarea "Eliminar problema" que desencadena la acción solicitada	Se debe indicar el tipo de acción que se desea realizar sobre el problema seleccionado	I
3	Solicitar confirmación	2	El sistema responde al instructor solicitando confirmación para completar la acción solicitada	Las operaciones críticas requieren confirmación por parte del usuario	I
4	Confirmar acción solicitada	3	El instructor deberá confirmar la acción solicitada	Las operaciones críticas requieren confirmación por parte del usuario	I
5	Guardar cambios	4	Si la acción es confirmada por el instructor, el sistema actualiza el evento seleccionado	Los cambios generados debe ser persistentes y verse reflejados en el sistema	A

Tabla 3.26 Eliminar problema

Actualizar problema					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir problema	-	El instructor elige el problema que desea actualizar	Se debe indicar el problema a actualizar	I
2	Elegir tarea requerida	1	El instructor selecciona la tarea "Actualizar problema" que desencadena la acción solicitada	Se debe indicar la acción que se desea realizar sobre el problema seleccionado	I
3	Completar formulario	2	El instructor completa el formulario que se muestra en pantalla y selecciona la operación "Guardar"	Permitir al instructor actualizar la información existente es la parte fundamental de la tarea	I
4	Solicitar confirmación	3	El sistema responde al instructor pidiendo confirmación para confirmar la acción solicitada	Las operaciones críticas requieren confirmación por parte del usuario	I
5	Confirmar acción solicitada	4	El instructor deberá confirmar la acción solicitada	Las operaciones críticas requieren confirmación por parte del usuario	I
6	Guardar cambios	5	Si la acción es confirmada por el instructor, el sistema actualiza el evento seleccionado	Los cambios generados debe ser persistentes y verse reflejados en el sistema	A

Tabla 3.27 Actualizar problema

Responder aclaración					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Seleccionar tarea requerida	-	El instructor selecciona la tarea "Atender aclaración"	El instructor deberá elegir la tarea adecuada	I
2	Responder aclaración	1	El instructor responde la aclaración solicitada	Es la parte fundamental de la tarea	I
3	Notificar usuarios	2	El sistema notifica a todos los usuarios que la aclaración correspondiente tiene una respuesta	Es necesario notificar a todos los participantes del evento para mantener la claridad del evento	A

Tabla 3.28 Responder aclaración

Consultar informe					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Elegir evento	-	El usuario elige el evento del cuál desea conocer los resultados	Se necesita especificar el evento	I
2	Consultar informe generado	1	El usuario consulta el informe generado		M

Tabla 3.29 Consultar informe

Tareas vinculadas al rol estudiante

Aquellos procesos iniciados por el estudiante son descritos en las siguientes tablas.

Evaluar solución					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Acceder al evento	-	El estudiante accede al evento en el que desea participar	Se requiere indicar el evento en el que se desea participar	I
2	Elegir problema	1	El estudiante elige el problema que desea resolver	Se deberá indicar el problema que el estudiante desea resolver	I
3	Implementar solución	2	El estudiante implementará un programa en el lenguaje de programación de su preferencia		M
4	Cargar solución	3	El indicará cuál es la implementación que desea publicar	Publicar el código fuente permitirá al sistema obtener el recurso fundamental para evaluar el desempeño del estudiante	I
5	Publicar solución	4	El estudiante publicará su solución una vez que su implementación se encuentre lista para enviarse al servidor	Publicar el código fuente permitirá al sistema obtener el recurso fundamental para evaluar el desempeño del estudiante	I
6	Evaluar solución	5	El sistema evaluará el código fuente adjunto a la publicación hecha por el estudiante	La tarea más importante del sistema es medir el desempeño del estudiante basado en los resultados esperados por el instructor de acuerdo al problema	A
7	Generar resultados	6	El sistema generará retroalimentación para el estudiante	Es necesario ofrecer retroalimentación al estudiante en relación a la publicación hecha	A
8	Publicar resultados	7	El sistema comunicará los resultados de la evaluación al estudiante y al instructor		M

Tabla 3.30 Evaluar problema

Solicitar aclaración					
#	**Nombre**	**Predecesor**	**Descripción**	**Justificación**	**Tipo**
1	Acceder a la sección adecuada	-	El estudiante accede a la sección de aclaraciones	Se debe elegir la sección donde la tarea requerida se encuentre	I
2	Elegir tarea requerida	1	El estudiante elige la tarea "Solicitar aclaración" de la sección de aclaraciones.	Se requiere elegir la tarea adecuada.	A
3	Editar aclaración	2	El estudiante deberá redactar la aclaración que desea solicitar	El estudiante debe redactar su aclaración lo más claramente posible	I
4	Enviar aclaración	3	El estudiante deberá elegir la acción "Realizar aclaración" una vez que ha terminado de editarla.	El estudiante debe enviar su aclaración para poder que esta pueda ser atendida	I
5	Notificar a otros usuarios	4	El sistema deberá notificar a los otros usuarios (independientemen del rol) sobre la notificación que ha sido realizada	Todos los usuarios deben ser notificados. Las solicitudes de aclaración son públicas al igual que su respuesta	A

Tabla 3.31 Solicitar aclaración

3.6 Prototipo del sistema

El prototipo ofrece a los usuarios una perspectiva de la apariencia que tendrá el sistema. Es particularmente útil ya que les permite familiarizarse con la manera en que interactuarán con él. Inicialmente se plantea la posibilidad de presentar a los usuarios una vista dividida en tres secciones con diferentes propósitos.

Panel	Descripción
Panel superior o panel de acceso	Funciona como panel de acceso a la cuenta de usuario. Se proponen establecer en el funcionalidades relacionadas con el la identificación de la cuenta y el área de notificaciones
Panel izquierdo o panel de tareas	Se propone agrupar todas las tareas relacionadas con eventos, problemas, configuración de la cuenta, administración de contenido entre otras funciones
Panel central o panel de contenido	Este panel cambiará de manera dinámica en función de la tarea o actividad solicitada por los usuarios. Puede desplegar tablas, resultados de búsquedas, reportes, entre otras cosas

Tabla 3.32 Descripción de los paneles en que será dividida la vista inicial.

Las siguientes figuras presentan los modelos conceptuales de las vistas que serán presentadas a los usuarios.

Figura 3.2 Prototipo de la vista general.

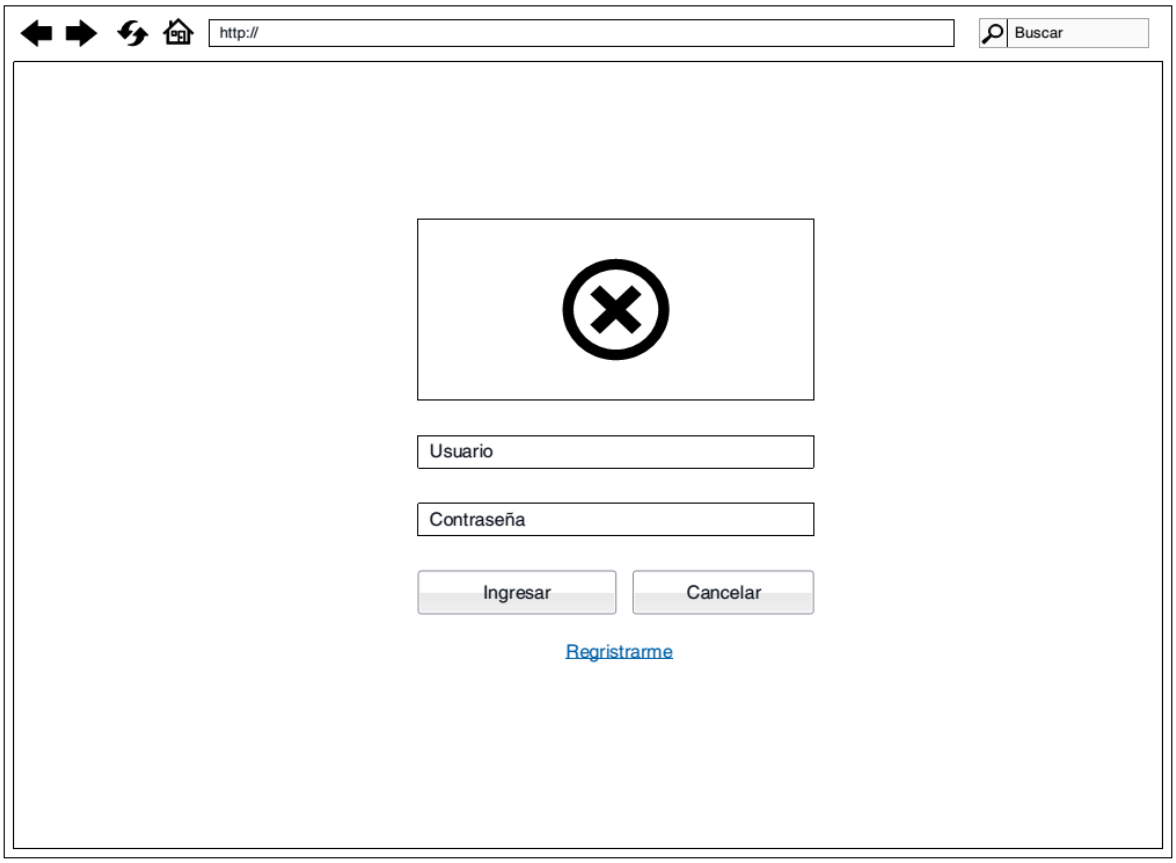


Figura 3.3 La vista de inicio de sesión.

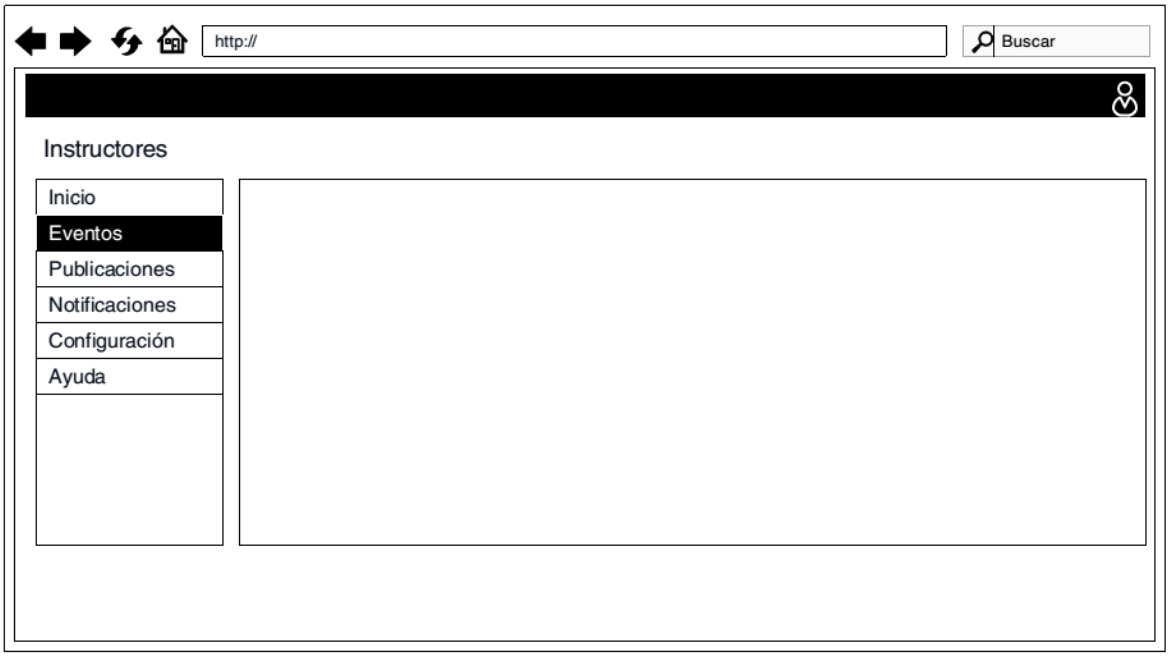


Figura 3.4 Vista general del sistema.

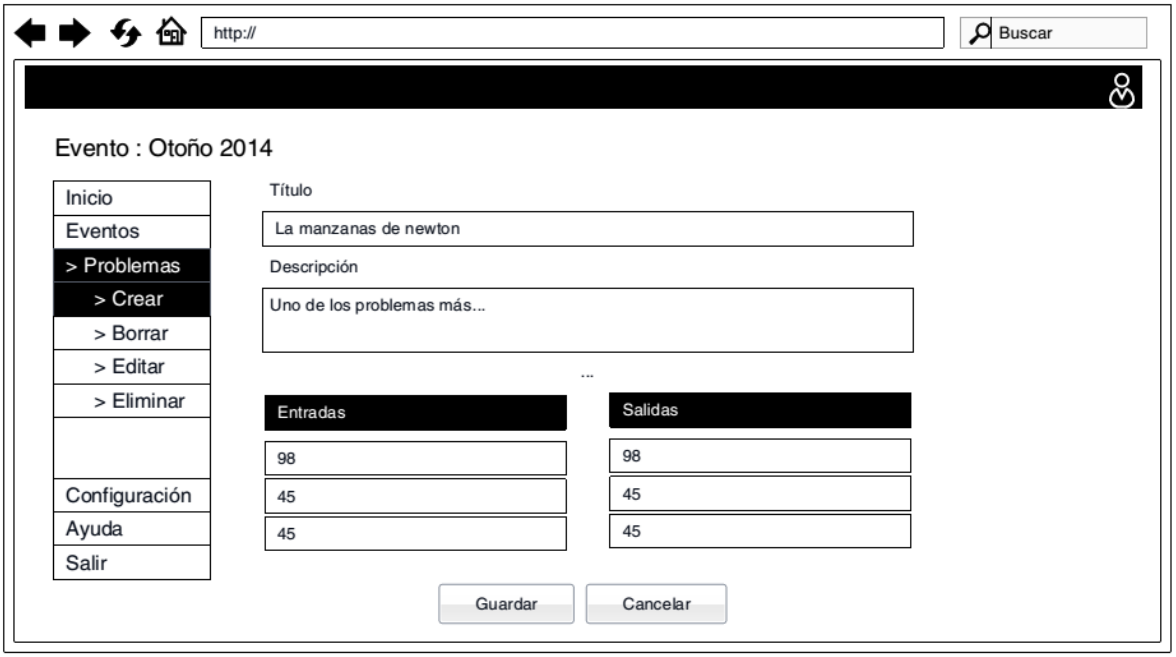


Figura 3.5 Edición de un problema.

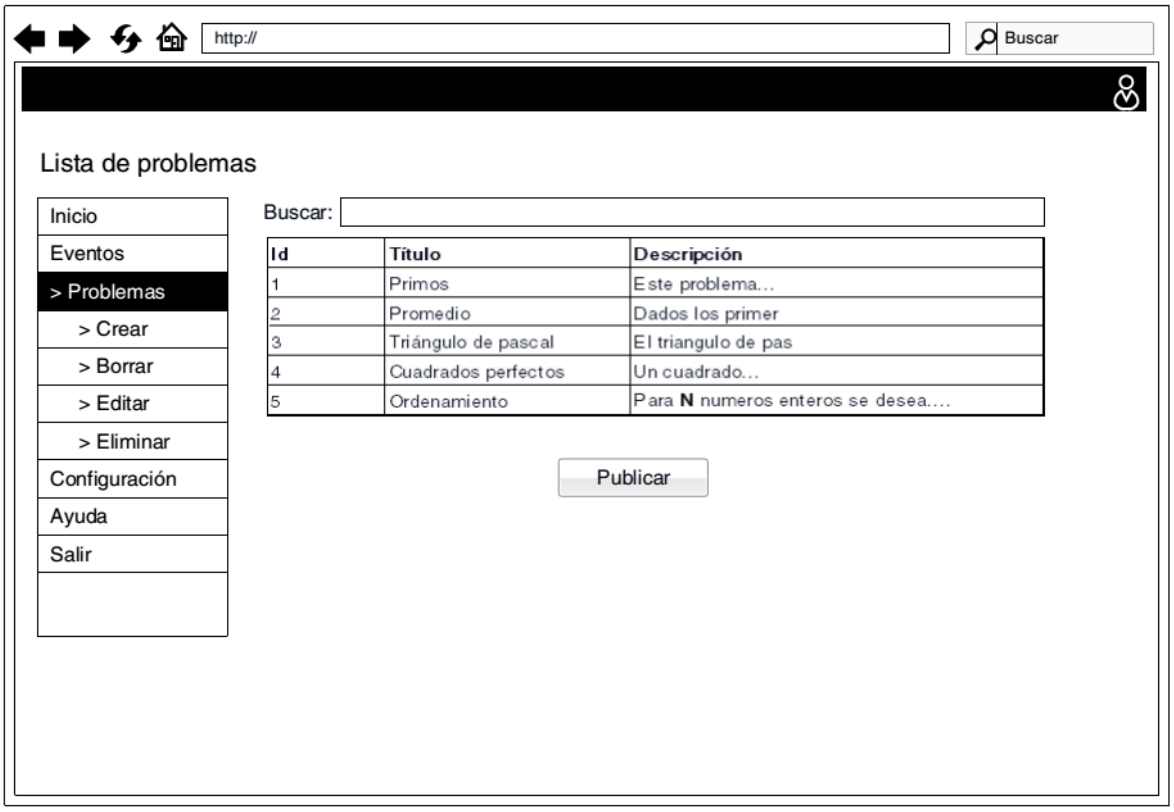


Figura 3.6 Vista de problemas en forma de lista.

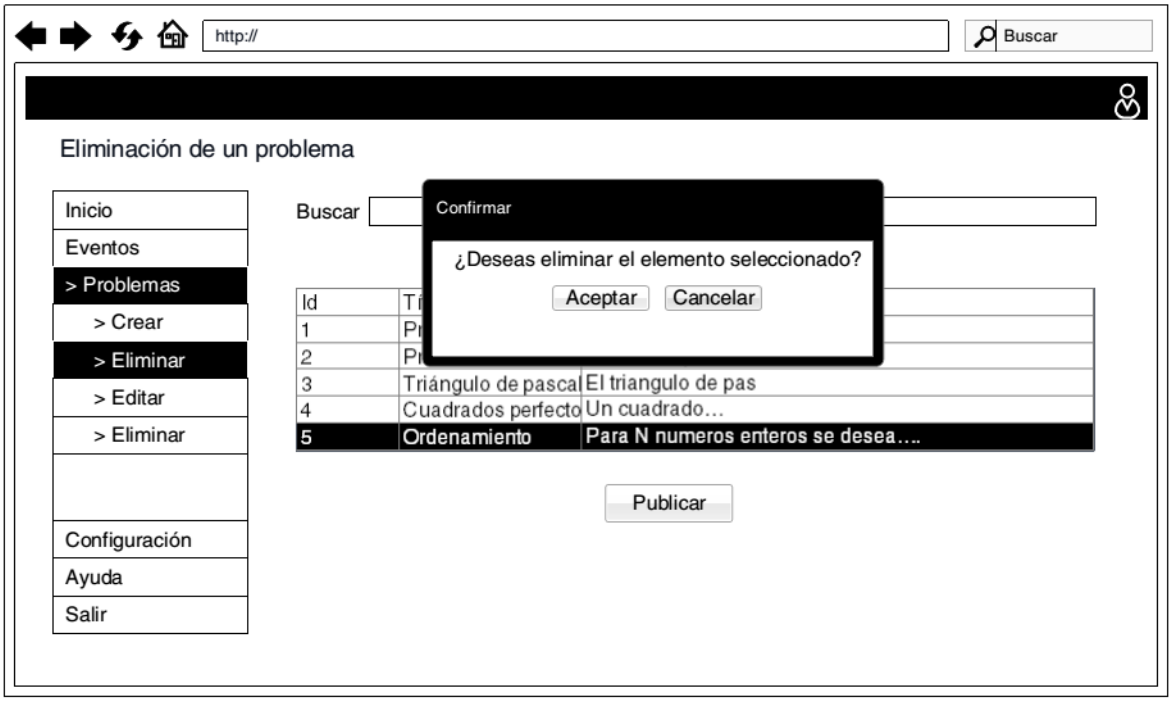


Figura 3.7 La eliminación y otras operaciones críticas requieren confirmación del usuario.

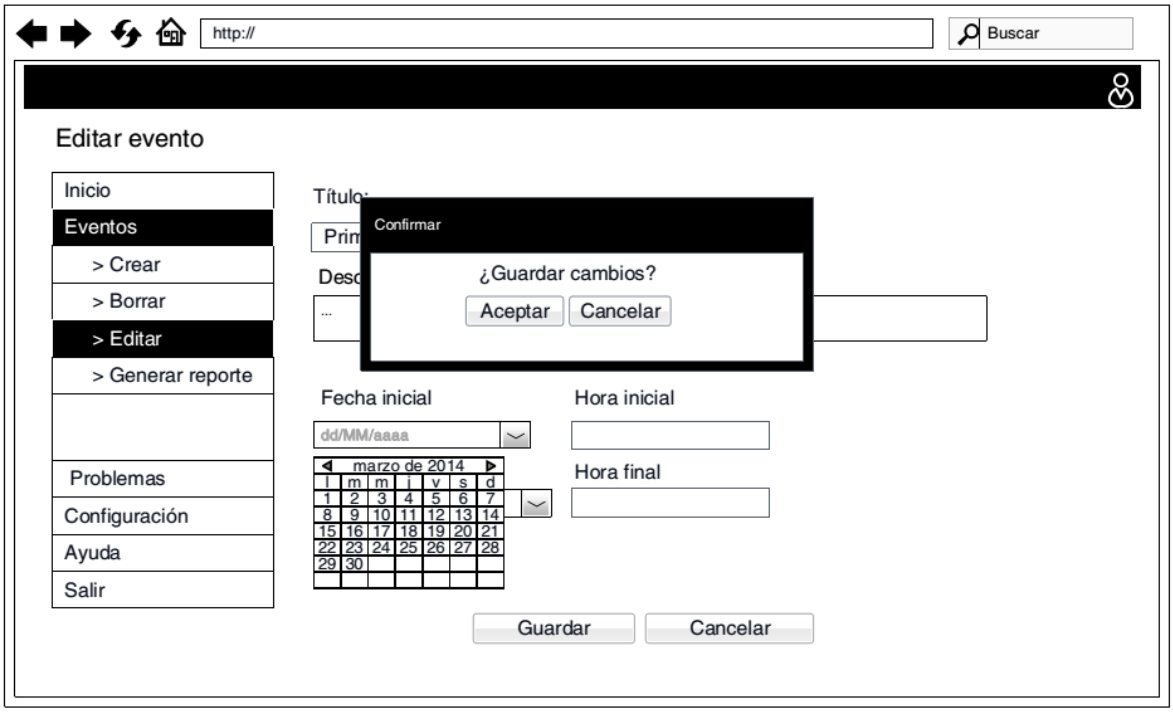


Figura 3.8 Calendarización de un evento.

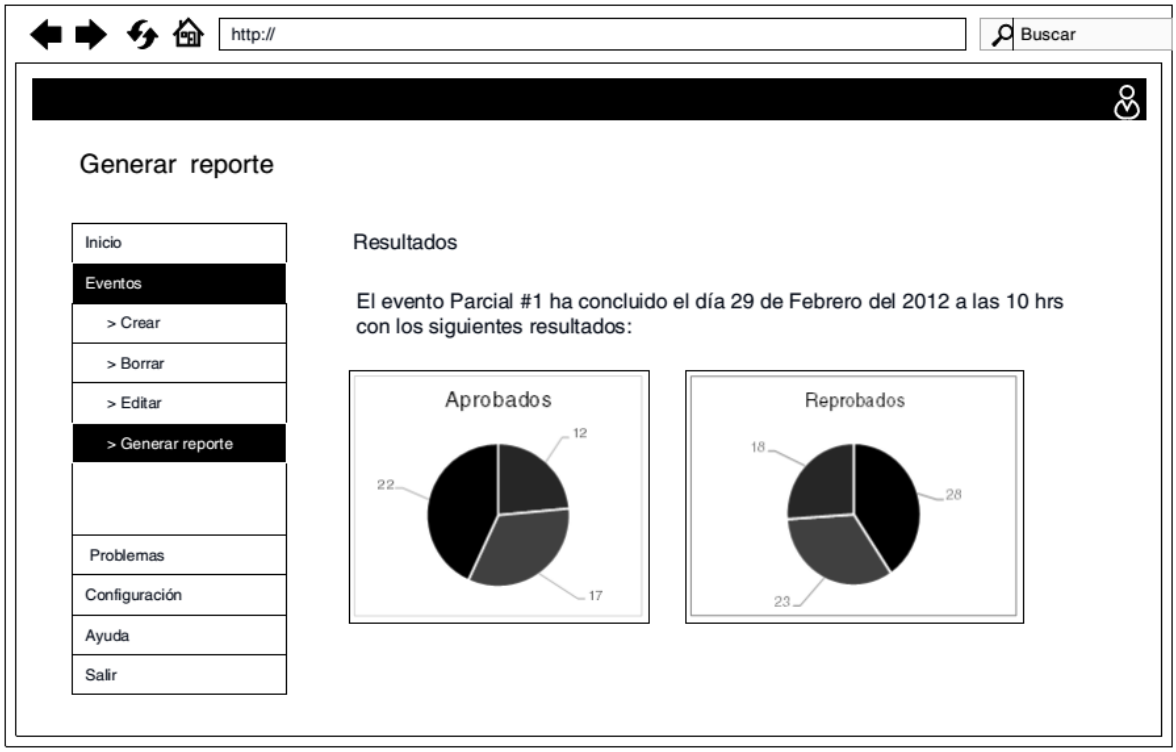


Figura 3.9 Prototipos de los reportes que serán generados.

Capítulo 4 . Implementación

El propósito de este capítulo consiste en presentar los detalles que ha ayudado a cumplir con el objetivo general.

4.1 Cronograma de actividades

El siguiente esquema representa las etapas que han involucrado el desarrollo del proyecto.

#	Fase	Semana	Duración aproximada
1	Planeación	1	5 días
2	Levantamiento de la infraestructura física	2	5 días
3	Implementación	3	60 días
4	Documentación	15	5 días

Tabla 4.1 Fases de desarrollo del proyecto.

4.2 Metodología de desarrollo

Para determinar la metodología de desarrollo, fue necesario identificar los factores más influyentes sobre las fases que constituyentes del proceso de producción del proyecto, siendo el tiempo el factor más determinante. De esta manera se requirió aplicar los principios de desarrollo ágil de aplicaciones sin adoptar una metodología de manera explícita.

4.3 Técnica de programación

Una metodología de desarrollo ágil requiere técnicas de programación que permitan reducir los costes de desarrollo y producción frente a cambios en las especificaciones de los usuarios. Además se debe garantizar la calidad del producto, maximizando su simplicidad sin que esto obstaculice su crecimiento. Sin dudarlo, representó un escenario ideal para aplicar técnicas de programación dirigida por pruebas.

4.4 Arquitectura física

Las siguientes características representan las especificaciones técnicas para el equipo de cómputo donde se ha desarrollado el sistema.

Hardware

- Procesador Intel Core 2 Duo a 2,93 GHz
- Memoria RAM de 2 GB de capacidad

- Disco Duro con capacidad de 120 GB
- Periféricos de entrada y salida

Software

- Visual Studio 2013
- Framework de desarrollo .NET versión 4.5
- IIS 8.1
- Navegadores WEB Internet Explorer 9, Mozilla Firefox, Google Chrome y Safari

4.5 Arquitectura lógica

Una arquitectura distribuida en capas asigna las responsabilidades más significativas de la aplicación a módulos independientes enriqueciendo la calidad de su composición. El diseño está conformado por las capas de dominio, infraestructura, operación y presentación.

Capa de dominio

Es responsable de representar las clases que componen el dominio de la aplicación. Está implementada mediante una librería de clases escrita en lenguaje C# que expone las clases que subyacen bajo el dominio del problema así como sus estados correspondientes y reglas de ámbito.

Capa de infraestructura de acceso a datos

Esta capa proporciona los mecanismos necesarios para la gestión y administración de datos a partir de clases constituyentes de la capa de dominio. Está diseñada bajo el enfoque *code first* e implementada mediante las tecnologías que integran Entity Framework 6.

Capa de operación

Es responsable de gestionar los procesos internos de la aplicación. Define y expone las solicitudes hechas al sistema coordinando su ejecución y resolución. Es una librería de clases escrita en C#, compuesta por dos módulos: evaluación y servicios.

Módulo de servicios

Está compuesto por un conjunto de componentes interrelacionados que implementarán alguna de las siguientes interfaces: *Request, Response o Response Handler*.

Request

Representa una petición a la aplicación de alguna de sus operaciones internas.

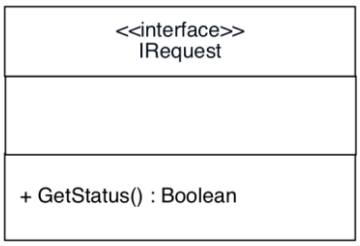


Figura 4.1 Interface *IRequest*

Response

Representa la resolución dada a una operación solicitada.

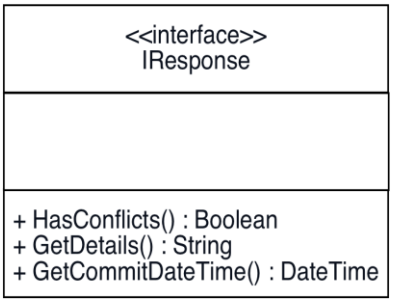


Figura 4.2 Interface *IResponse*.

Request handler

Encapsula las operaciones que componen está capa. Su principal función es vincular las solicitudes enviadas al sistema devolviendo un solo resultado.

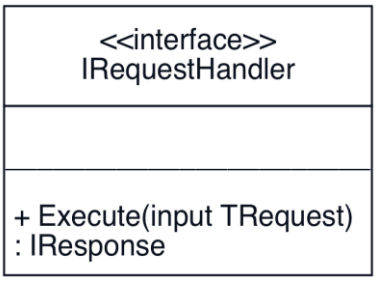


Figura 4.3 Interface *Request Handler*.

Módulo de evaluación

Es un módulo de la capa de operación cuya finalidad es coordinar la ejecución los componentes necesarios que determinan la retroalimentación que debe darse al estudiante, como resultado del desempeño del código fuente que ha implementado.

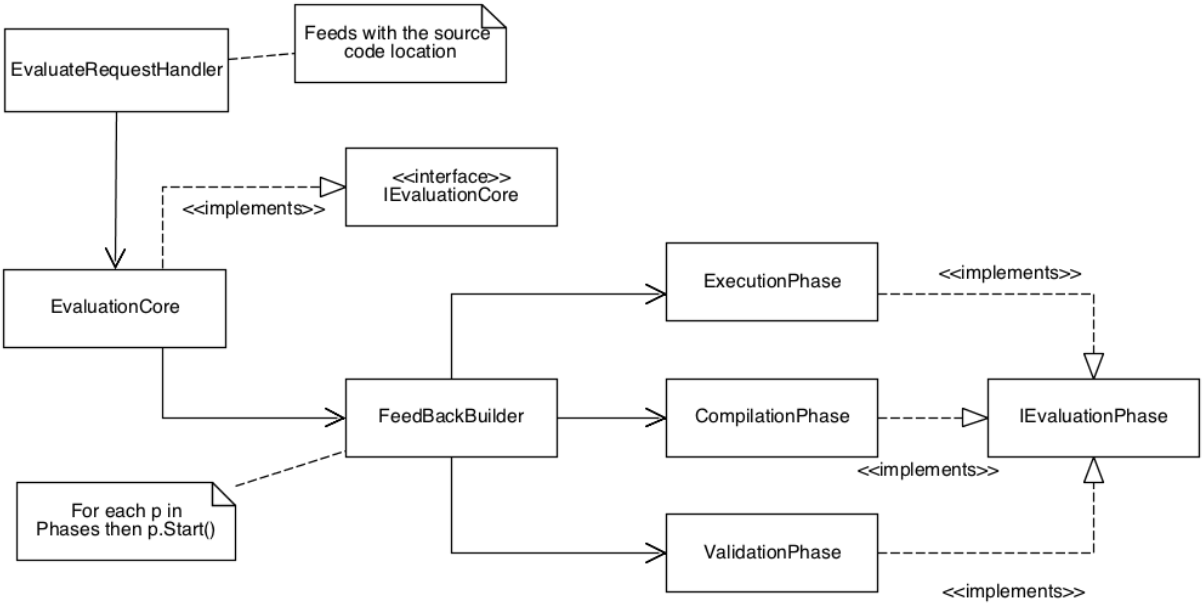


Figura 4.4 Módulo de evaluación

Este módulo implementa una metodología de evaluación adoptada y adaptada de los sistemas tipo ACM-ICPC descritos en el estado del arte de este documento a partir de los siguientes criterios:

1. Se valida la integridad del código fuente buscando en su estructura palabras exclusivas para el uso del sistema.
2. Se verifica que el código fuente publicado sea compilable.
3. Si el proceso de compilación es posible se genera un programa ejecutable a partir del código fuente.
4. Se ejecuta el programa generado.
5. Se determina la valoración de la solución basada en la comparación de los resultados generados por el programa y los resultados esperados por el instructor ante la problemática planteada.
6. Se establece el tiempo y memoria consumidos por el programa, desde que inició su ejecución hasta que verifica el último caso de prueba.

ICollaborator

Un colaborador es una entidad que realiza una o más tareas específicas de múltiples propósitos. Su estructura está descrita en la interface *ICollaborator* presentado en la siguiente figura:

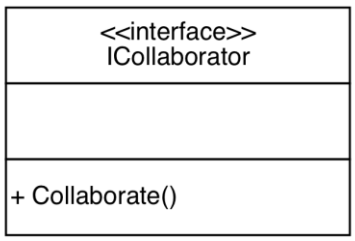


Figura 4.5 Interface ICollaborator.

Un colaborador se comunica con los componentes que lo invocan mediante el patrón de diseño *Observer*.

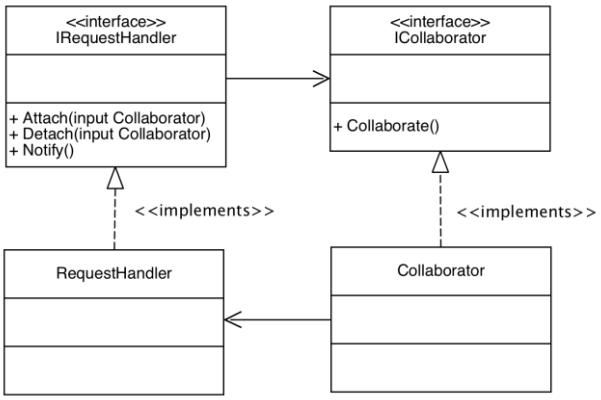


Figura 4.6 Comunicación entre un colaborador concreto y un manejador de solicitud.

IEvaluationCore

Es una interface que describe el esquema fundamental para crear clases concretas encargadas de coordinar las fases que integran la metodología de evaluación. Una clase que implemente esta interface, se encargará de administrar el código fuente determinando su desempeño en función de los criterios de evaluación establecidos por el problema que tenga en contexto.

Esta interface, está diseñada bajo las características descritas por los patrones de diseño *State* y genera sus resultados aplicando los fundamentos del patrón de diseño *Builder.*

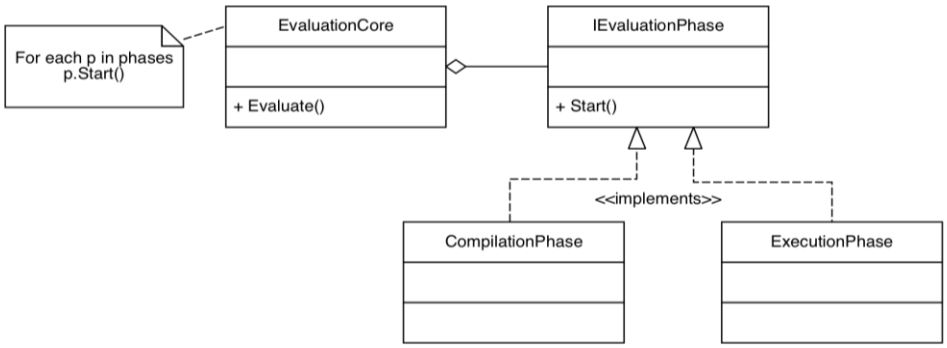


Figura 4.7 Patrón de diseño *State.*

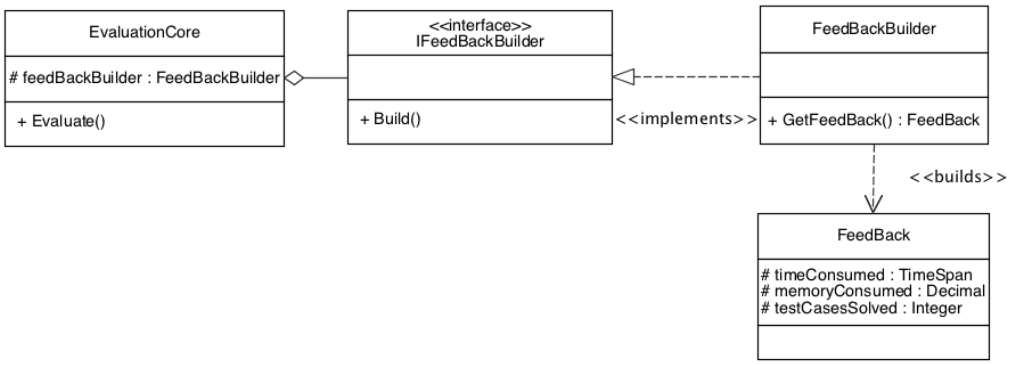


Figura 4.8 Patrón de diseño *Builder* usado para generar los resultados del programa ejecutado.

IEvaluationPhase

Cada fase constituyente de la nueva metodología de evaluación deberá implementar la interface *IEvaluatorPhase*. La implementación de clases concretas a partir de interfaces permitiría agregar nuevas fases a la metodología propuesta.

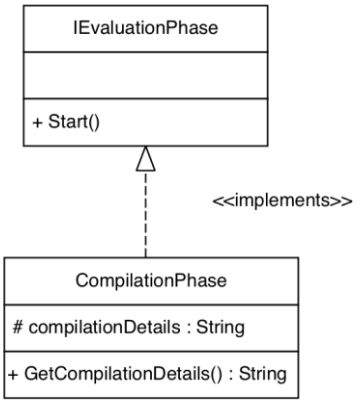


Figura 4.9 Interface IEvaluationPhase

Capa de presentación

Es responsable de interactuar directamente con los usuarios del sistema ofreciendo los servicios que componen la capa de operación. Ha sido implementada mediante una aplicación web ASP.NET MVC 5.

Aquellas vistas cuyas funcionalidades integren tecnologías de empuje implementan en su estructura interna el patrón de diseño MVVM descrito en el Capítulo 1.

Figura 4.10 Arquitectura lógica del sistema

4.6 Especificaciones de uso del sistema

El siguiente apartado tiene por objetivo describir la mecánica de uso del sistema.

Presentación del sistema

Siguiendo los prototipos que se establecieron en el Capítulo 3, la pantalla de inicio del sistema de evaluación se divide en tres paneles como muestra la siguiente figura.

Figura 4.11 El sistema gestiona la actividad relacionada a la cuenta.

El panel de tareas

El panel de administración se encuentra localizado en la parte izquierda de la pantalla. Su función consiste en desplegar las tareas que un usuario dispone dependiendo del rol al que pertenece.

Acción		Descripción
	Noticias	Muestra información relacionada con eventos, publicaciones, comentarios mediante una línea de tiempo
	Eventos	Agrupa las tareas relacionadas la gestión de eventos
	Estudiantes	Presenta al instructor toda la comunidad de estudiantes registrados en el sistema. Podrá invitar a uno o más estudiantes a formar parte de sus grupos o eventos
	Grupos	Presenta las operaciones relacionadas a la administración de grupos
	Publicaciones	Presenta operaciones relacionadas a la administración de contenido como artículos y vínculos
	Reportes	Presenta operaciones relacionadas a la gestión de reportes
	Ayuda	Muestra tutoriales multimedia con contenido relacionado a las especificaciones de uso del sistema

Tabla 4.2 Panel de tareas para el rol instructor.

Acción		Descripción
	Noticias	Muestra información relacionada con eventos, publicaciones, comentarios en mediante una línea de tiempo
	Eventos	Agrupa las tareas relacionadas participación en eventos del estudiante
	Comunidad	Muestra operaciones relacionadas a los grupos existentes en toda la comunidad. Al seleccionar uno el estudiante podrá ver su actividad
	Grupos	Presenta operaciones relacionadas a los grupos en los que el estudiante tiene alguna participación
	Perfil	Permite gestionar la información relaciona a la cuenta de usuario del estudiante
	Soluciones	Presenta el historial de soluciones que el estudiante ha publicado en los eventos en que ha participado
	Reportes	Permite al estudiante generar reportes de los eventos en los que ha participado
	Ayuda	Muestra tutoriales multimedia con contenido relacionado a las especificaciones de uso del sistema

Tabla 4.3 Panel de tareas para el rol Estudiante

Ayuda

La ayuda es accesible desde el panel de tareas y es proporcionada por medio de videos cortos con una duración menor a tres minutos aprovechando el poder multimedia del HTML 5.

Calendarización de eventos

La nueva metodología de evaluación se inicia mediante la calendarización de un evento en cualquiera de sus modalidades.

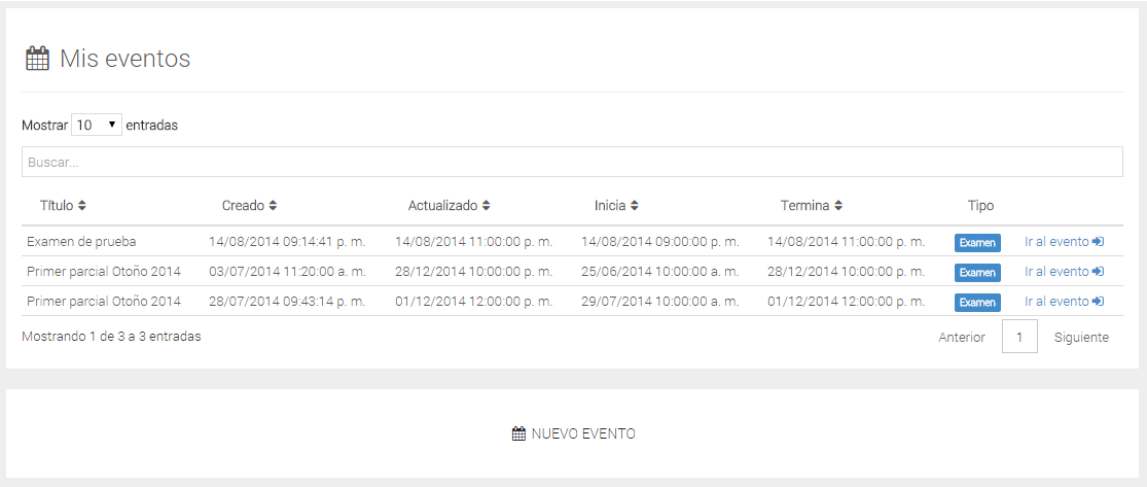


Figura 4.12 Eventos un instructor.

Crear un evento

Para crear un nuevo evento el instructor solicitará al sistema la vista correspondiente haciendo click en el botón “NUEVO EVENTO” debajo del listado de eventos.

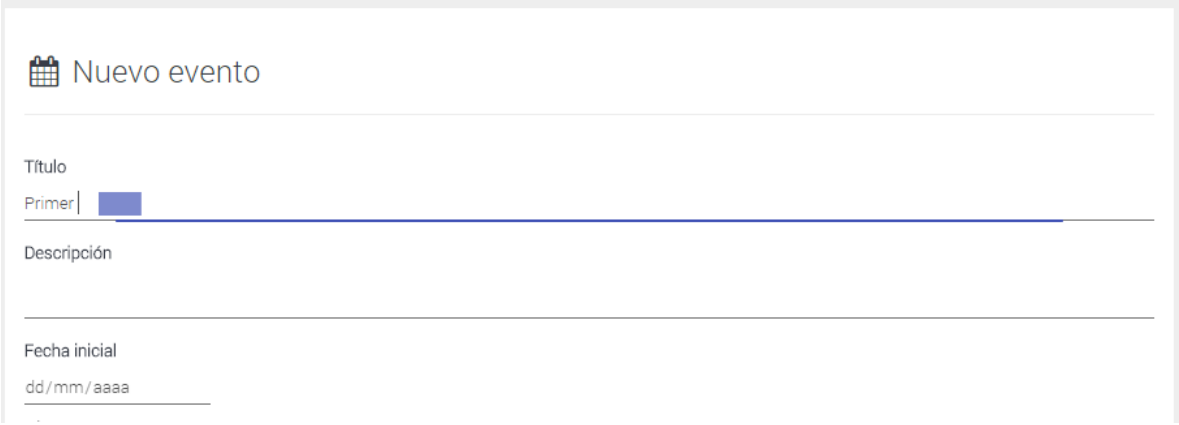


Figura 4.13 Creando un nuevo evento.

Figura 4.14 Evento calendarizado.

Creado el evento, será posible acceder sus detalles. La siguiente figura presenta los detalles un evento. Esta vista se mantendrá actualizada en todo momento gracias a las tecnologías de empuje.

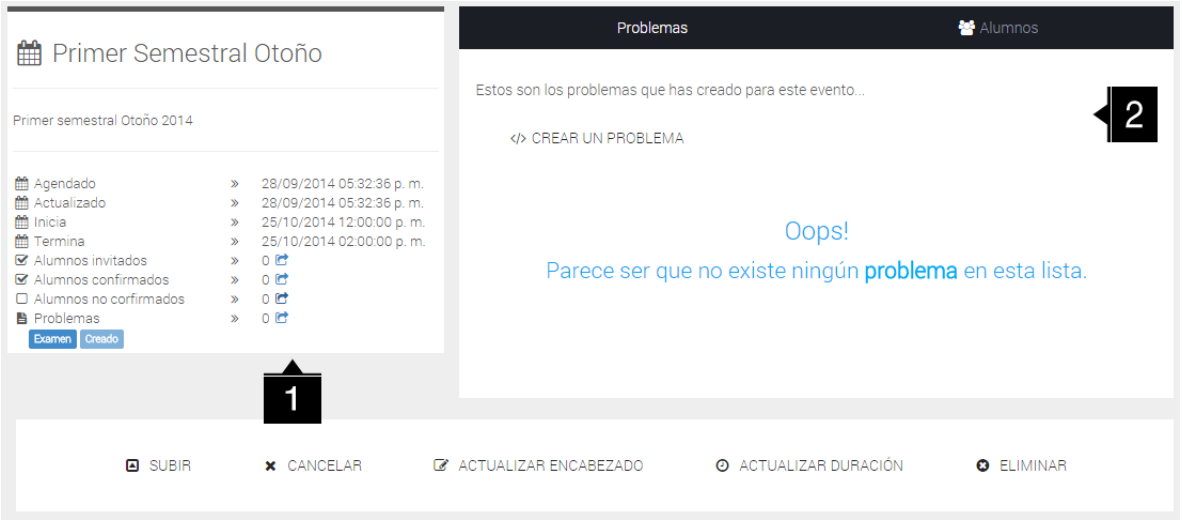


Figura 4.15 Detalles del evento.

Un evento requiere que estar compuesto por al menos un problema. Para editar un problema el instructor deberá iniciar las operaciones de edición de problemas en los detalles del evento.

Editar un problema

Encabezado

En encabezado periten dar una identidad al problema y establecer sus criterios de evaluación.

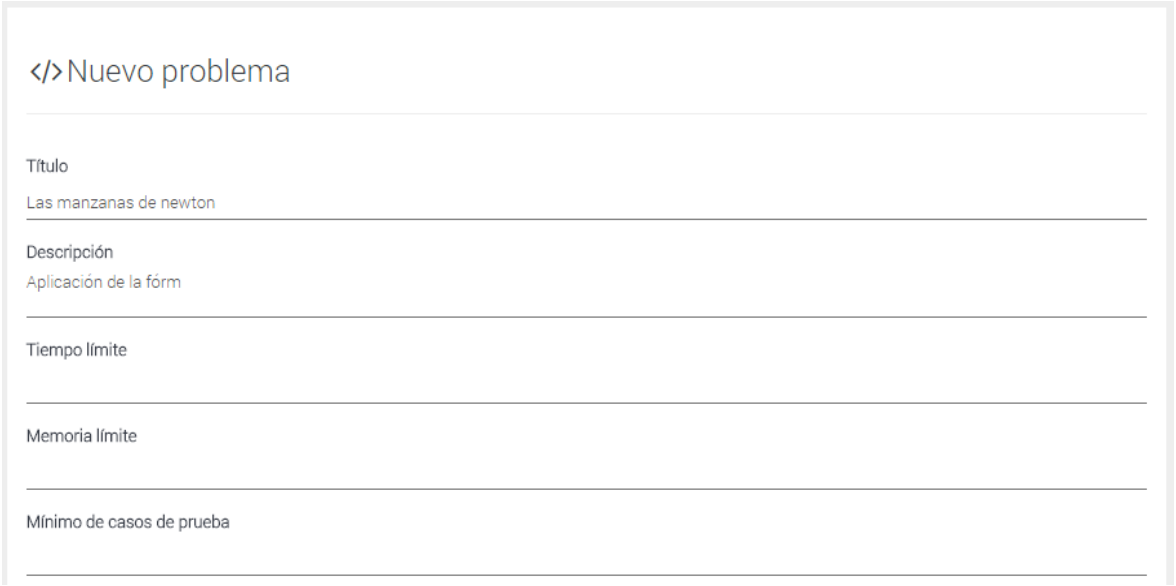


Figura 4.16 Encabezado de un problema

Cuerpo y casos de prueba

El cuerpo describe las especificaciones que el estudiante debe seguir para obtener los resultados que el instructor espera como parte de su evaluación.

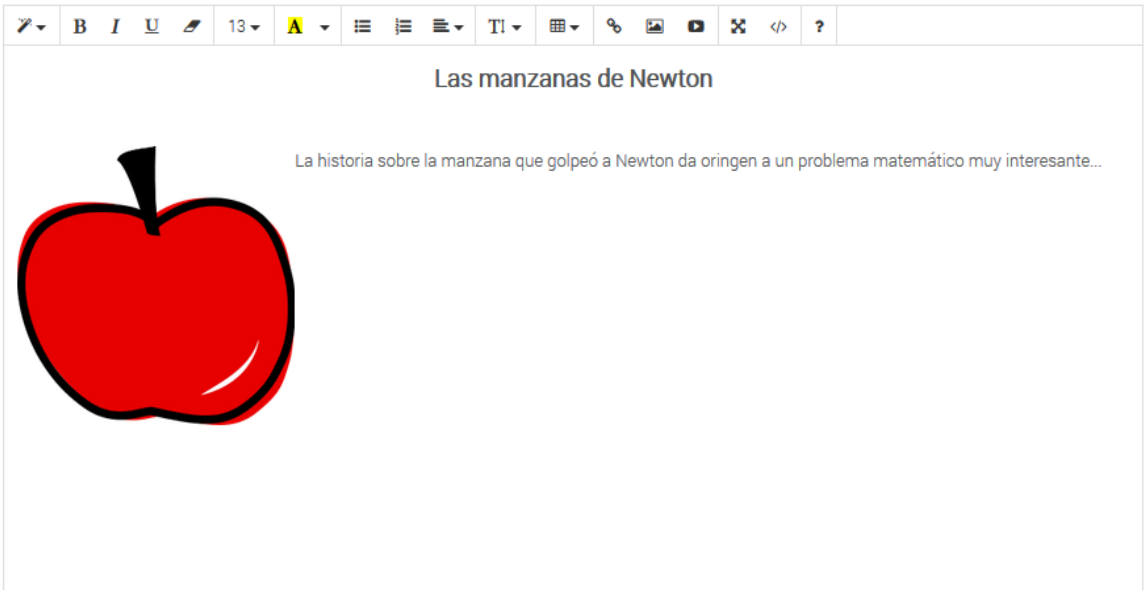


Figura 4.17 El cuerpo del problema hace uso Summernote, un complemento en JavaScript que ofrece funciones para la edición de contenido multimedia.

Los casos de prueba son un conjunto de duplas **clave-valor** que constituyen la respuesta adecuada para el problema de acuerdo al instructor.

Figura 4.18 Los casos de prueba se alimentan uno a uno mediante una tabla dinámica o publicando un archivo separado por caracteres especiales.

Si el problema es creado de manera exitosa será posible visualizarlo desde la perspectiva de los estudiantes.

Figura 4.19 La vista al público permite al instructor ver el problema desde la perspectiva que tendría cualquier alumno.

El estado de un problema determina propiedades de acceso de acuerdo a las necesidades del instructor.

Color	Estado	Descripción
	Creado	Representa el estado inicial del problema. Permite al instructor editar sus propiedades
	Listo	Habilita la publicación de soluciones mientras el evento al que pertenece se encuentre en proceso
	Cancelado	Imposibilita la publicación de soluciones si el instructor determina que el problema no es integro

Tabla 4.1 Estados de un problema

Otro de los requisitos necesarios para poder publicar un evento es que se encuentre vinculado al menos a un estudiante. Estas tareas de vinculación están disponibles en la vista "Detalles del evento" donde se ofrece la posibilidad de invitar un estudiante o grupo de ellos.

Figura 4.20 La pestaña Alumnos de la ficha de vinculación ofrece la posibilidad de vincular al evento con los grupos del instructor.

Al seleccionar alguno de los botones descritos en la figura anterior, se mostrará un dialogo con los estudiantes o grupos disponibles para participar en el evento.

Figura 4.21 Grupos disponibles.

Publicar un evento

Una vez cumplidos los prerrequisitos descritos anteriormente será posible hacer público el evento permitiendo que inicie y culmine de acuerdo a las fechas designadas por el instructor. Esta

operación debe realizarse con mucho cuidado ya que los estudiantes invitados serán notificados a respecto por lo que se solicitará confirmación por parte del instructor.

Restricciones de acceso a un evento

El estado del evento plantea un conjunto de restricciones de acceso a sus detalles y los problemas que lo componen.

Color	Estado	Descripción
	Creado	Representa el estado inicial de todo evento. El instructor podrá modificar sus atributos además de agregar, eliminar y editar todo tipo de problemas
	En línea	El evento se encuentra en espera de iniciar y culminar en las fechas indicadas por el instructor
	Iniciado	Indica que el evento se encuentra en progreso. Aquellos estudiantes que ha sido invitados podrán participar en él
	Cancelado	El evento no permitirá a ningún estudiante participar en él y se dará por culminado independientemente de sus fechas inicial y final
	Finalizado	El evento no aceptará la publicación de nuevas soluciones y permitirá consultar resultados generales y generar reportes

Tabla 4.1 Estados de un evento

Evento en progreso

Iniciado el evento cada estudiante deberá elegir un problema, analizarlo, proponer una solución y finalmente publicarla en espera de sus resultados.

Figura 4.22 La figura muestra el título del evento en la parte superior (1). Debajo del título se muestra un contador que permanecerá activo hasta que el evento culmine (2). También es posible visualizar los ejercicios que integran el evento.

Publicación de una solución

La metodología de evaluación propuesta al instructor se encuentra ahora implementada y solo es necesario que el estudiante publique sus soluciones ante los ejercicios planteados. Para esto, elegirá la operación “QUIERO RESOLVER ESTE PROBLEMA”.

Figura 4.23 Operaciones disponibles para un problema.

Una ventana emergente aparecerá frente al estudiante permitiéndole elegir desde su equipo de cómputo el código fuente que desea publicar.

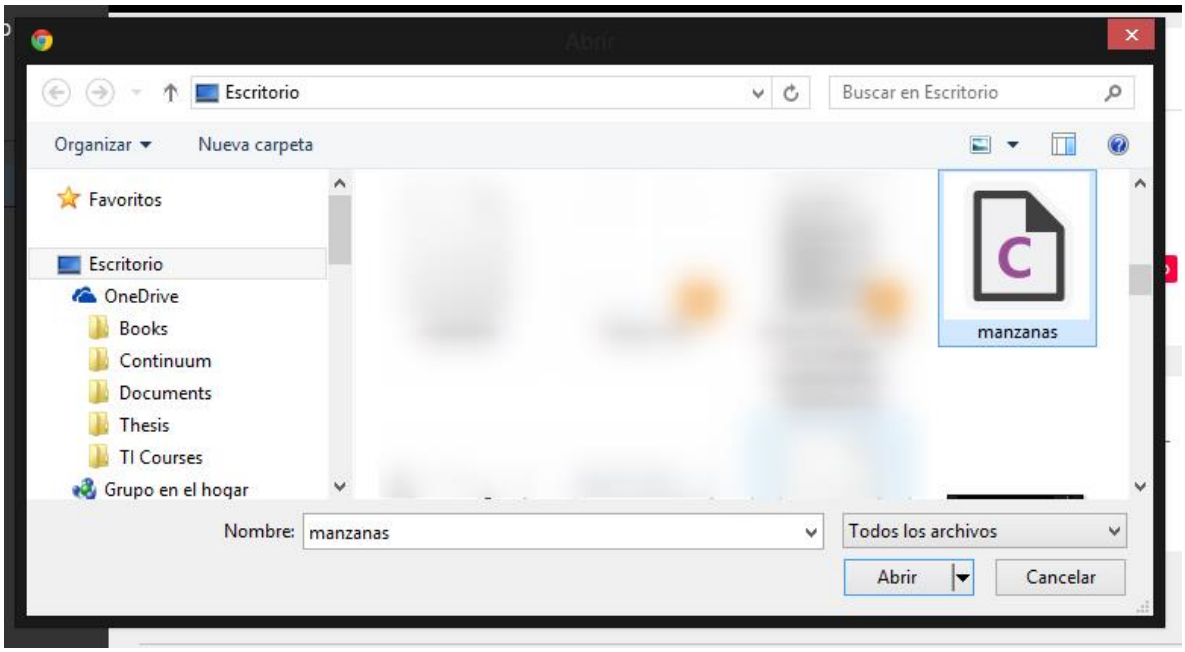


Figura 4.24 Ventana de carga de código fuente.

Figura 4.25 Código fuente cargado.

Al publicar la solución, el sistema realizará los procesos necesarios para evaluar el código fuente y ofrecerá retroalimentación al estudiante en función de los resultados que ha obtenido.

Figura 4.26 Resultado de la evaluación.

Ahora el estudiante podrá ver en el listado de problemas los problemas resueltos.

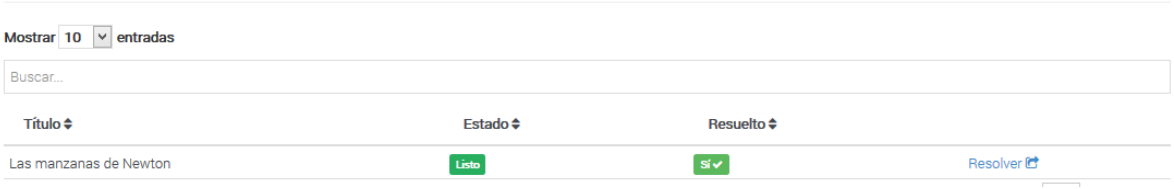


Figura 4.27 Un estudiante podrá intentar resolver un problema tantas veces como lo considere necesario.

Por otra parte, el instructor podrá ver en los detalles del problema las soluciones publicadas por los estudiantes que están participando en el evento.

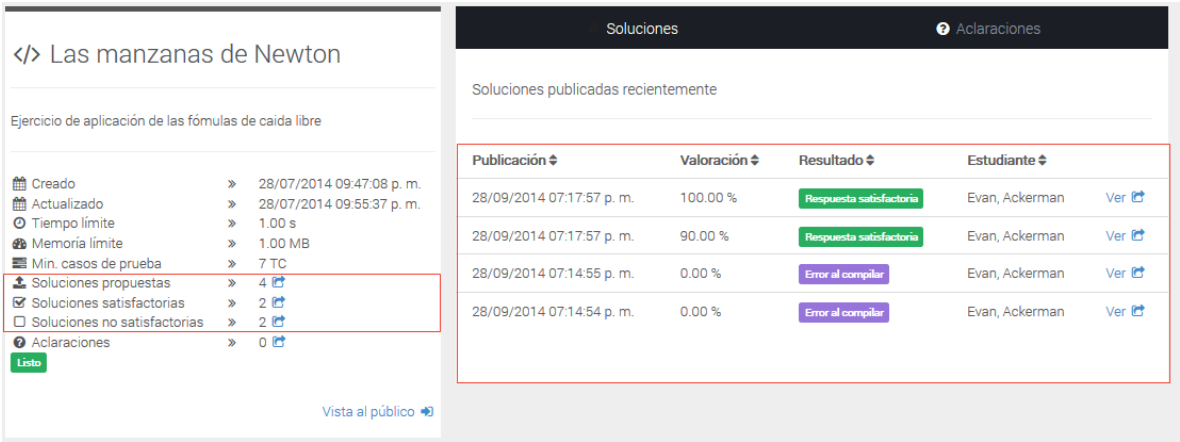


Figura 4.28 El instructor será informado en todo momento sobre los detalles del problema.

Si el instructor selecciona una solución podrá visualizar sus detalles correspondientes así como el código fuente adjunto a la publicación.

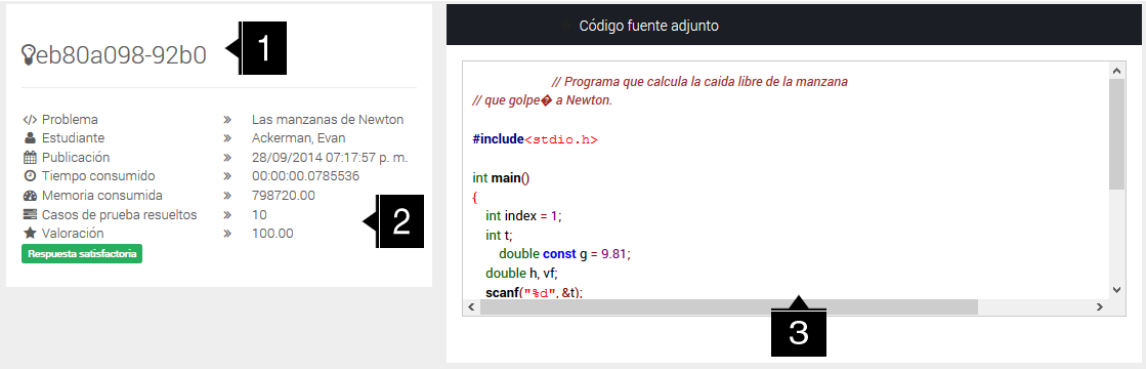


Figura 4.29 La figura presenta detalles sobre una solución publicada: el identificador de la solución (1), los resultados generados por el proceso de retroalimentación (2) y el código fuente adjunto a la publicación (3).

Retroalimentación otorgada a una solución

Los resultados que pueda tener una solución son descritos en la siguiente tabla.

Color	Resultado	Descripción
	Solución satisfactoria	La implementación ha resuelto el mínimo número de casos de prueba con un consumo de tiempo y memoria óptimos
	Solución insuficiente	La implementación ha resuelto el mínimo número de casos de prueba pero el tiempo o memoria consumida no son óptimos
	Solución no satisfactoria	La implementación no ha resuelto el mínimo número de casos de prueba designados
	Error en la compilación	El código fuente presenta un error en su sintáxis por lo que el proceso de compilación ha fallado
	Palabra restringida	El código fuente hace uso de alguna función o libreria de uso exclusivo del sistema
	Operación inválida	Representa un error artimético o de desbordamiento de la implementación durante su ejecución

Tabla 4.2 Códigos de una retroalimentación.

Estados de un problema

Al igual que un evento, un problema posee estados que determinan las propiedades de acceso a su contenido.

Color	Estado	Descripción
	Creado	Representa el estado inicial del problema. Permite al instructor editar sus propiedades
	Listo	Habilita la publicación de soluciones mientras el evento al que pertenece se encuentre en proceso
	Cancelado	Imposibilita la publicación de soluciones si el instructor determina que el problema no es integro

Tabla 4.3 Estados de un problema.

Solicitar Aclaraciones

Una aclaración permite solicitar al instructor información adicional relacionada a la estructura un problema. Esta operación se encuentra disponible en la parte inferior de cada problema.

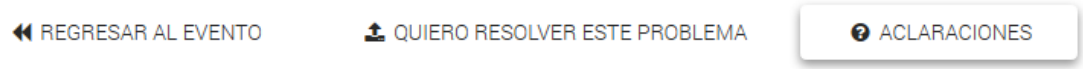


Figura 4.30 Posibles valores de una retroalimentación.

Al presionar el botón “ACLARACIONES” el estudiante visualizará el listado de aclaraciones. Para solicitar una nueva aclaración, será necesario presionar el botón “SOLICITAR ACLARACIÓN” y completar el dialogo que será mostrado.

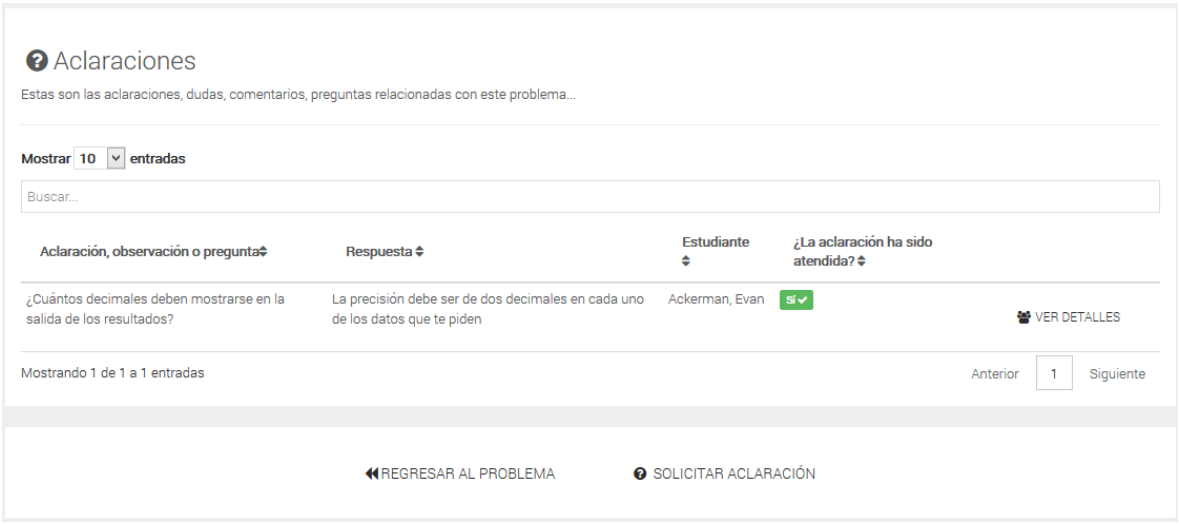


Figura 4.31 Listado de aclaraciones del problema “*Las manzanas de Newton*”.

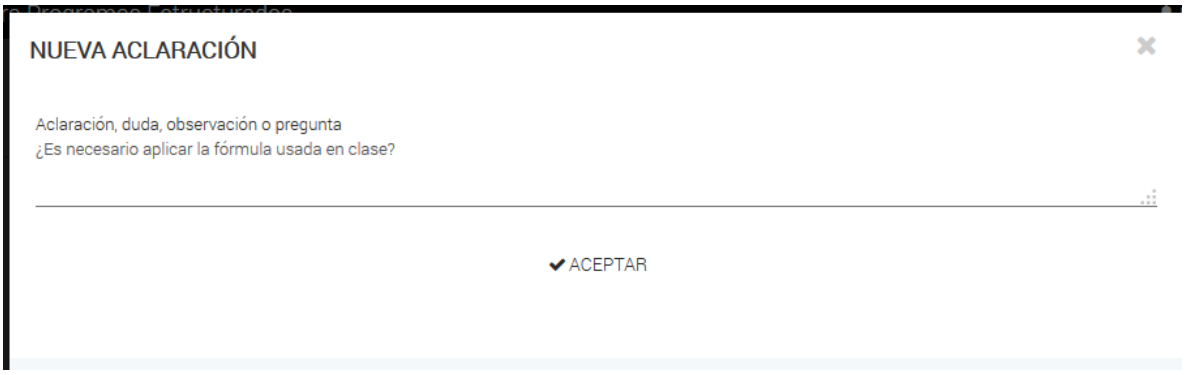


Figura 4.32 Nueva aclaración.

Después de completar el formulario, la aclaración aparecerá en el listado existente. El instructor verá la aclaración en los detalles del problema.

Figura 4.33 Aclaraciones del problema.

El instructor podrá ver la aclaración y resolverla de la misma forma en que fue solicitada.

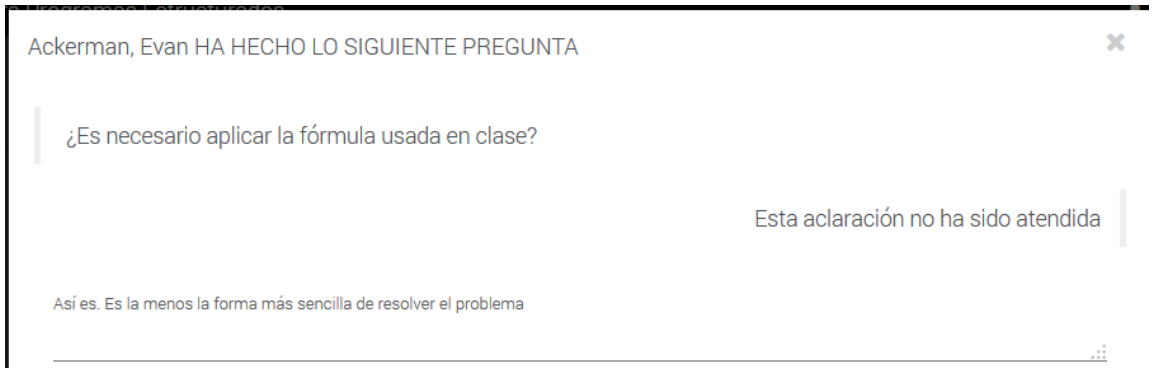


Figura 4.34 Una aclaración se responde de la misma manera con la que fue solicitada.

Resuelta la aclaración, el estudiante será notificado y verá la resolución dada por el instructor.

Reportes

Al finalizar el evento, será posible para cualquiera de los roles generar un breve reporte con datos estadísticos.

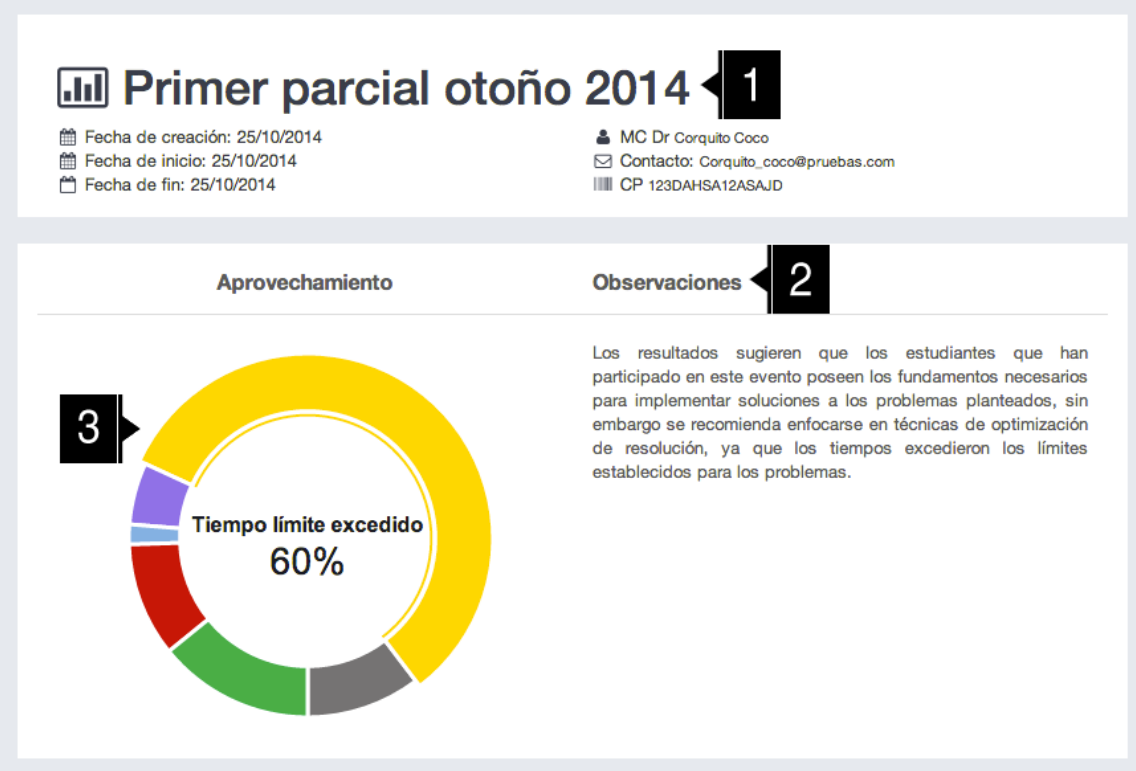


Figura 4.35 Se presentan detalles del evento que ha terminado, como su ficha técnica (1), observaciones (2) y gráficos de aprovechamiento (3).

Por otra parte, el estudiante podrá visualizar logros importantes en relación con su desempeño y en comparación con los del resto de sus compañeros.

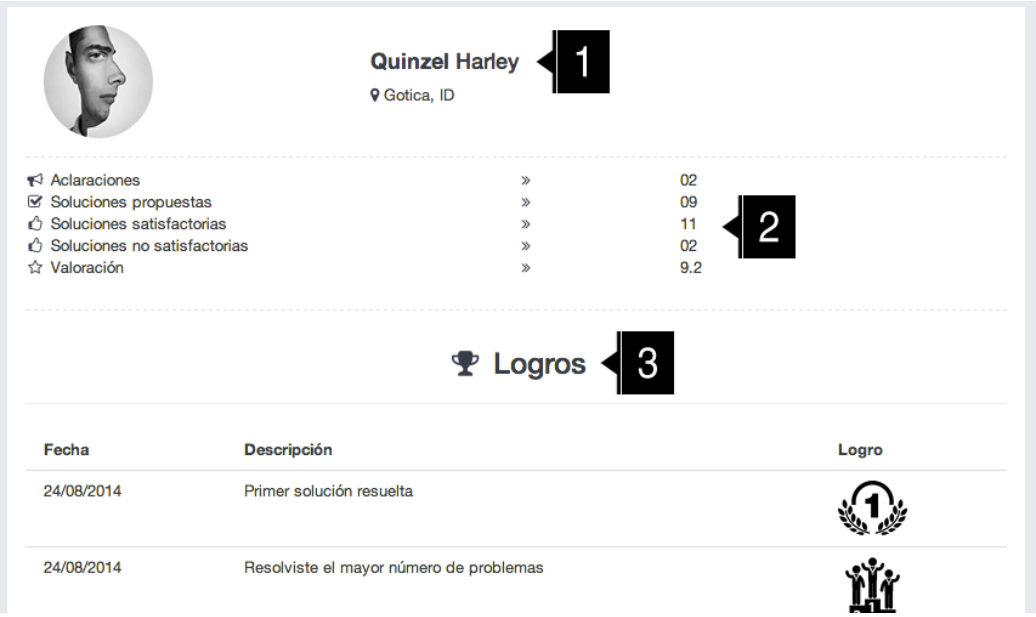


Figura 4.36 El estudiante podrá visualizar reportes como el que se muestra en la figura. Se muestra detalles como el perfil del estudiante (1), su desempeño (2) y sus logros (3).

Otras funcionalidades

Grupos

Para facilitar la administración de eventos y gestión de publicaciones el sistema permite al instructor generar grupos para clasificar a sus estudiantes.

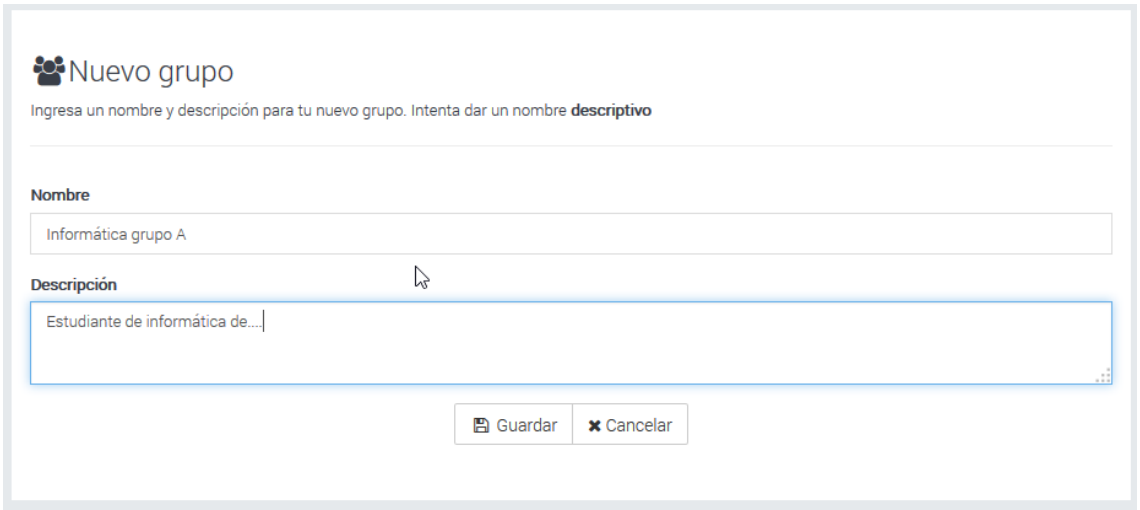


Figura 4.37 Los grupos delimitan el acceso al contenido y eventos del instructor.

Publicación de contenido

Una funcionalidad menor consiste en generar contenido multimedia para los grupos pertenecientes a un instructor. La mecanica de administración de este tipo de contenido puede ser comparada con la implenetación de un *blog* y fue desarrollada mediantes los mismos mecanimos usados para la edición de problemas.

4.7 Documentación ténica del sistema

Concluidas las fases que componen el ciclo de producción del proyecto es pertinente generar la documentación correspondiente, hecho que resulta muy importante ya que facilitará tareas de extensión del sistema en etapas posteriores permitiendo a los desarrolladores (quienes pueden estar familiarizados o no con él) conocer su composición interna.

Requisitos de Hardware y Software

Los requisitos definidos más adelante son los mínimos recomendados para que el sistema cumpla con sus funcionalidades sin que presente problemas de rendimiento y han sido establecidos considerandos los siguientes factores:

- Número de usuarios
- Número de usuarios simultáneos conectados a un mismo evento
- Estrés que puede sufrir el servidor en momentos críticos:
    - Número de usuarios conectados en los lapsos finales del evento.
    - Número de publicaciones realizadas en los lapsos finales del evento.
    - Tamaño en MB de las publicaciones realizadas.

Servidor

Software recomendado

- Sistema Operativo Windows 7 o superior
- IIS 8.1 instalado y configurado
- SQL Server 2008 R2 o superior para la administración de la base de datos

Hardware recomendado

- Memoria RAM de al menos 2GB
- Procesador Intel Core 2 Duo de 1,06 GHz a 3,33 GHz
- Disco duro de al menos 80 GB

Aunque las características de los equipos de cómputo de los usuarios que se conectarán al sistema se encuentran más allá de los alcances del sistema, se les recomendará el siguiente conjunto de especificaciones.

Cliente

Software recomendado

- Sistemas Operativos Windows, GNU Linux o MAC OS X.
- Navegador Web Internet Explorer, Google Chrome, Safari, Mozilla Firefox en sus versiones más recientes.
  - Cookies habilitadas
  - Javascript habilitado

Diagramas de secuencia

La serie de diagramas que se presentan a continuación representan los diagramas de secuencia constituyentes del sistema desarrollado.

Sección 1. Diagramas de propósito general

La siguiente serie de diagramas representa los diagramas de secuencia que son independiente del rol en contexto.

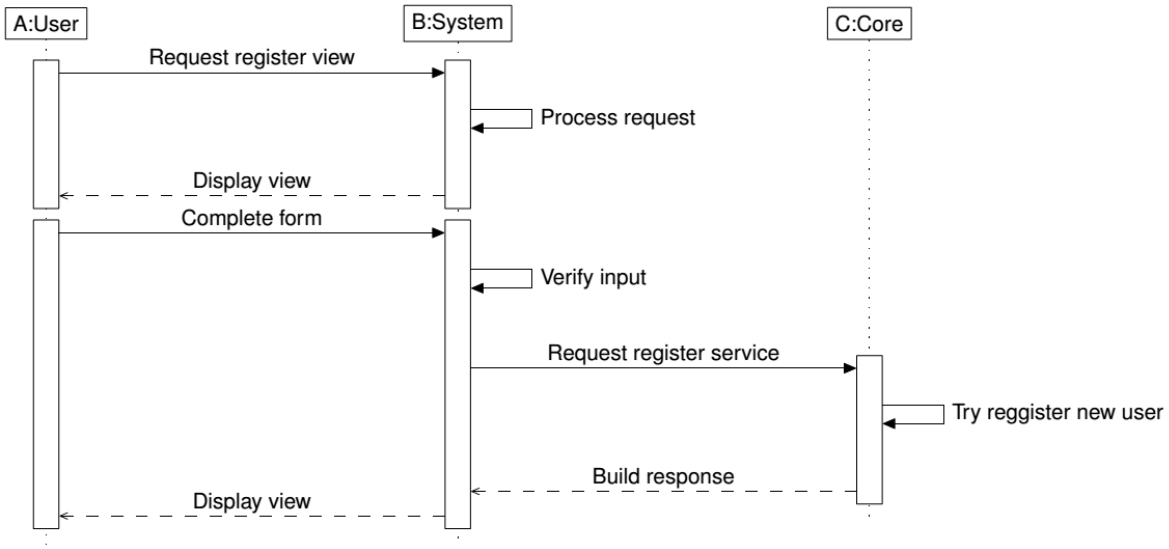


Diagrama 1. Registro de un nuevo usuario.

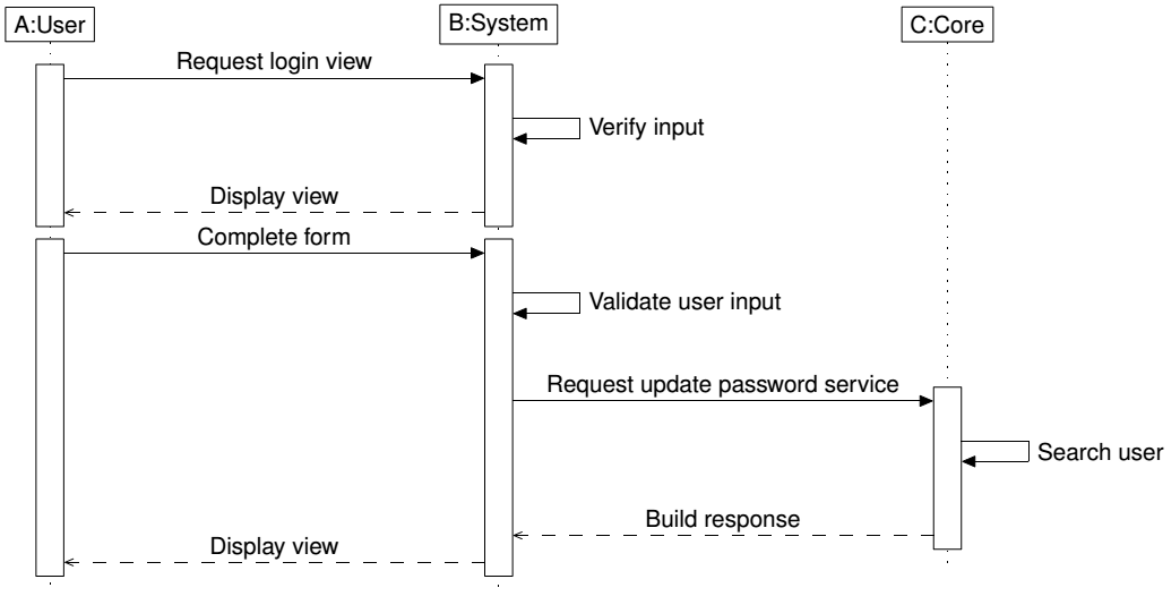


Diagrama 2. Acceder al sistema.

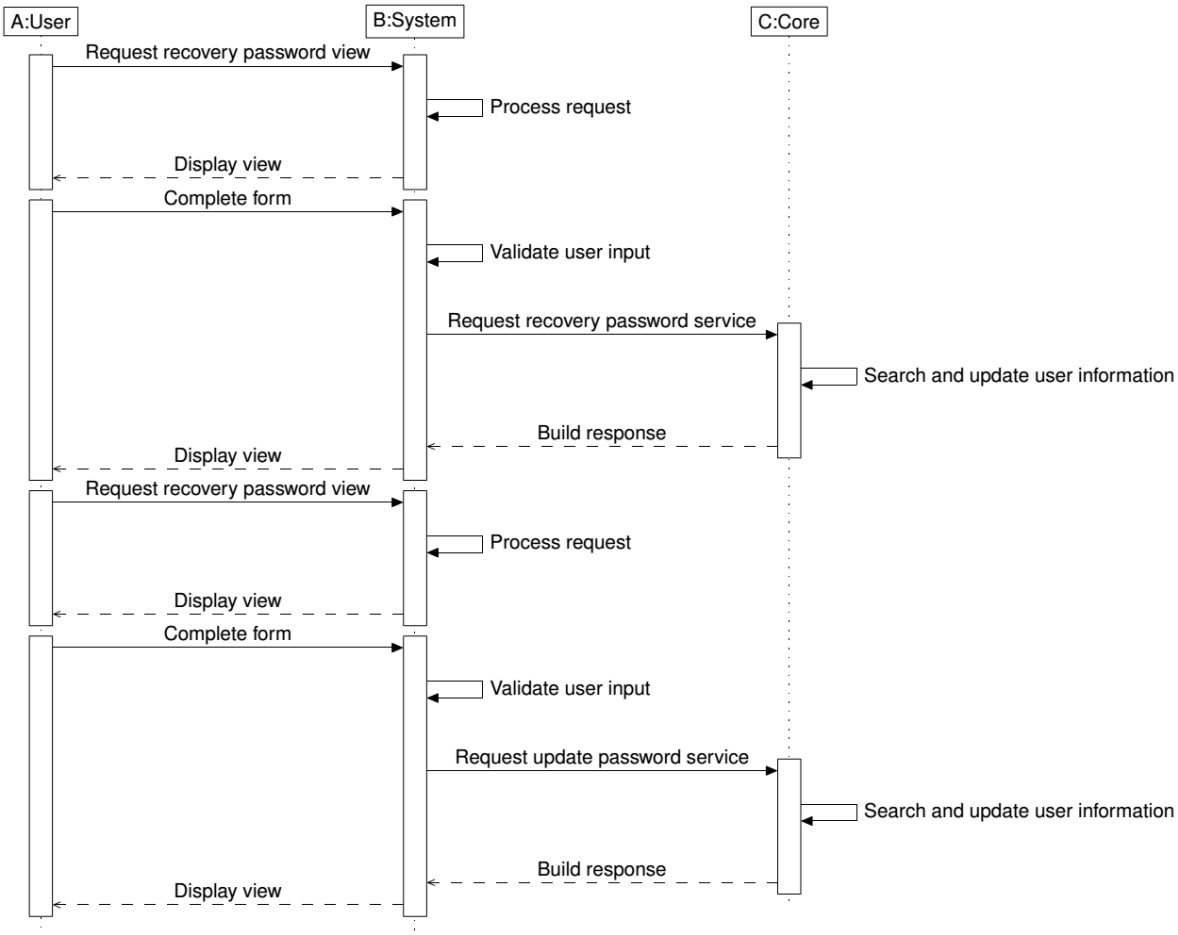


Diagrama 3. Recuperar contraseña.

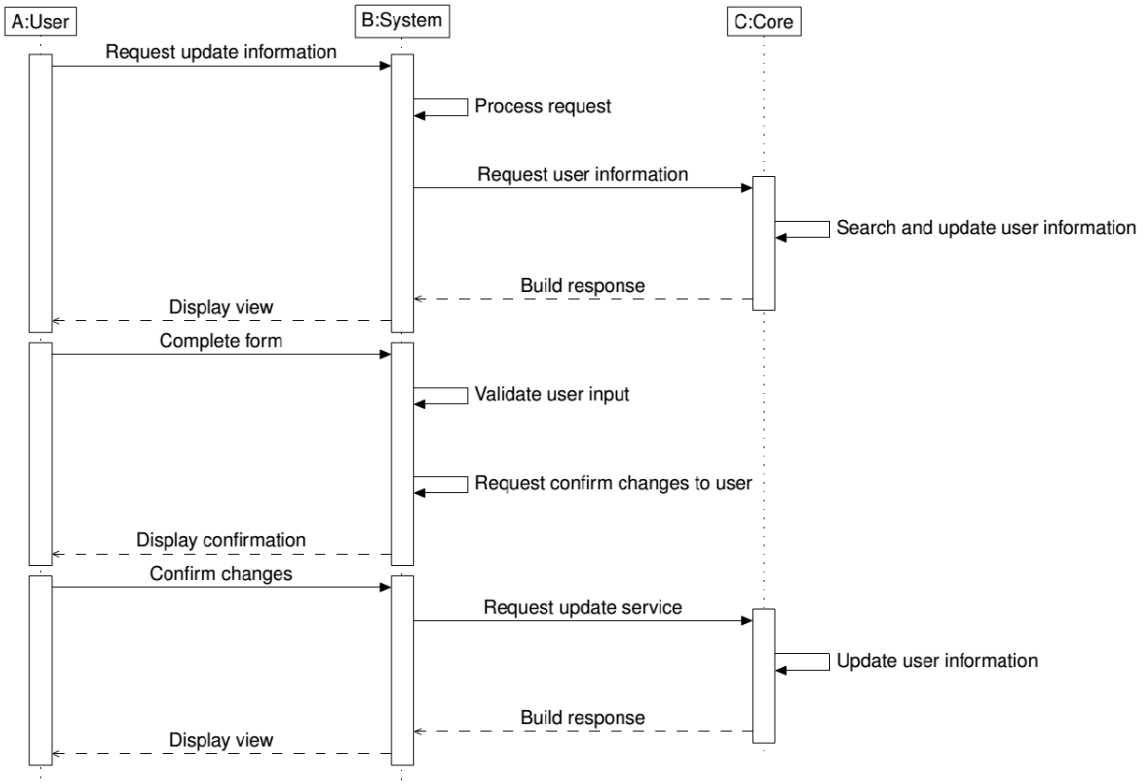


Diagrama 4. Actualizar información del sistema.

Sección 2. Diagramas de secuencia dirigidos por el instructor

La siguiente serie de diagramas representa los diagramas de secuencia que son iniciados por el rol instructor.

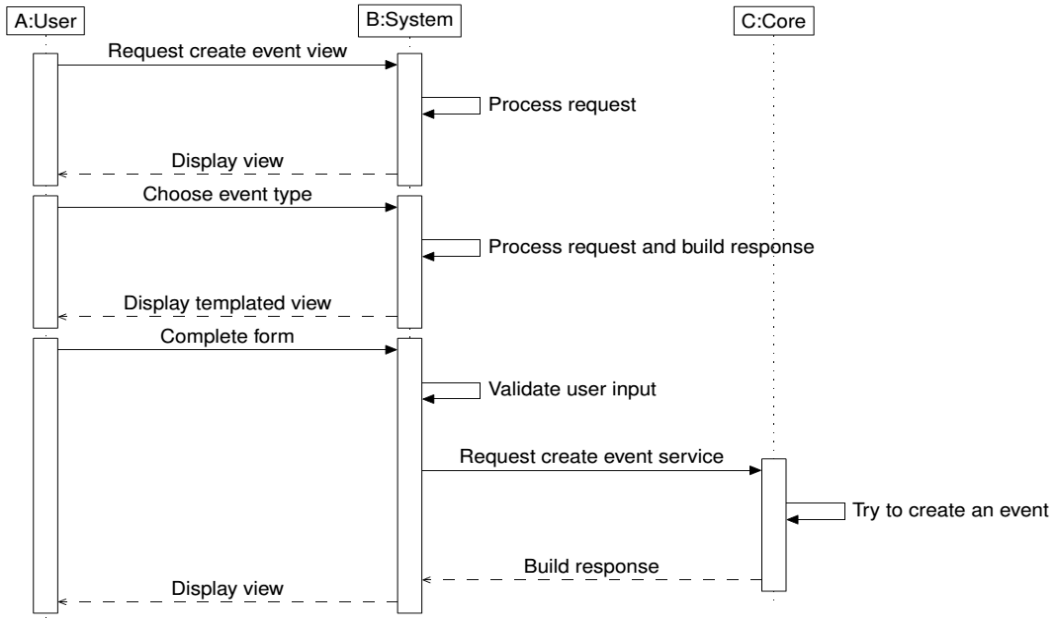


Diagrama 5. Calendarizar un evento.

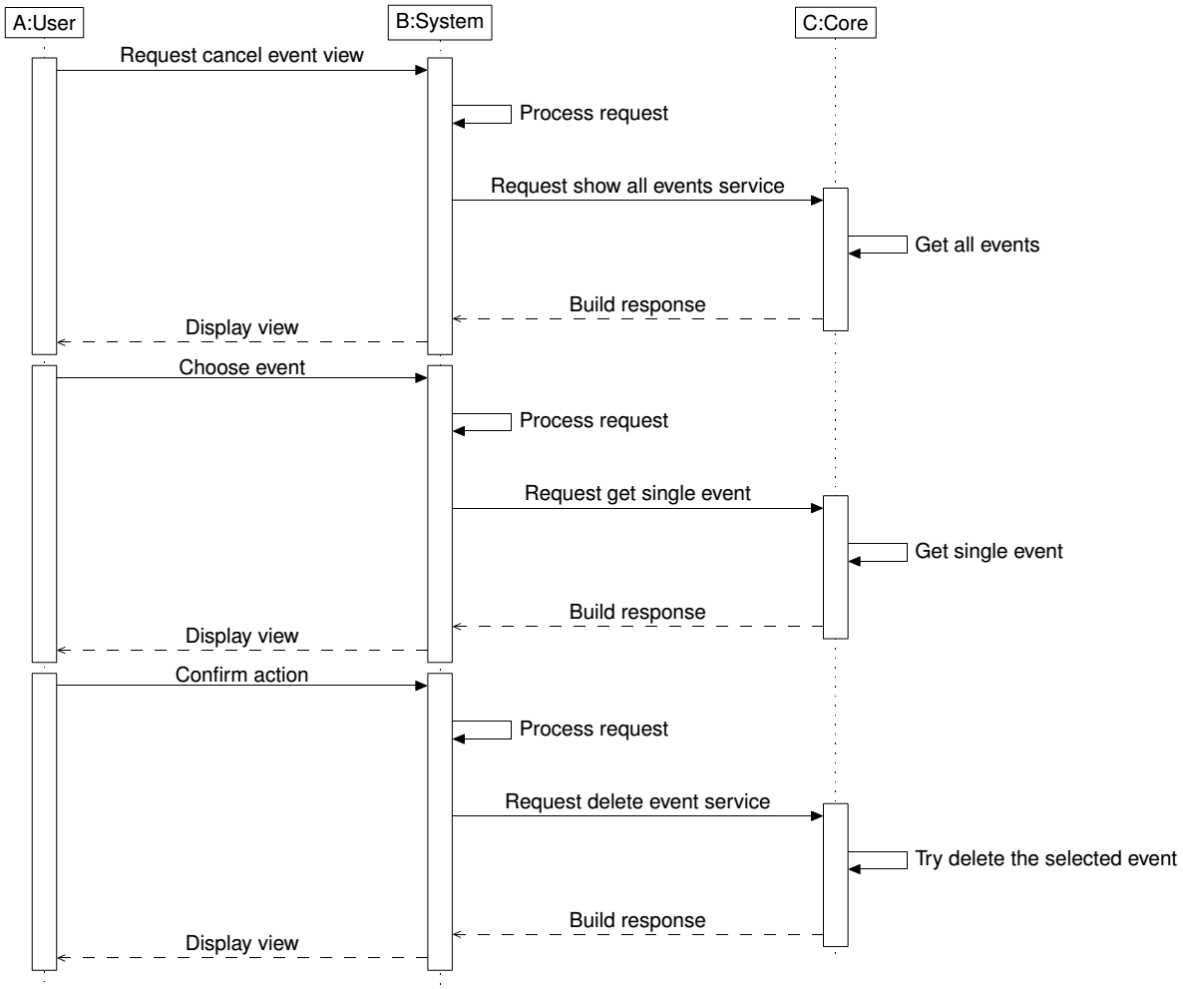


Diagrama 6. Cancelar un evento.

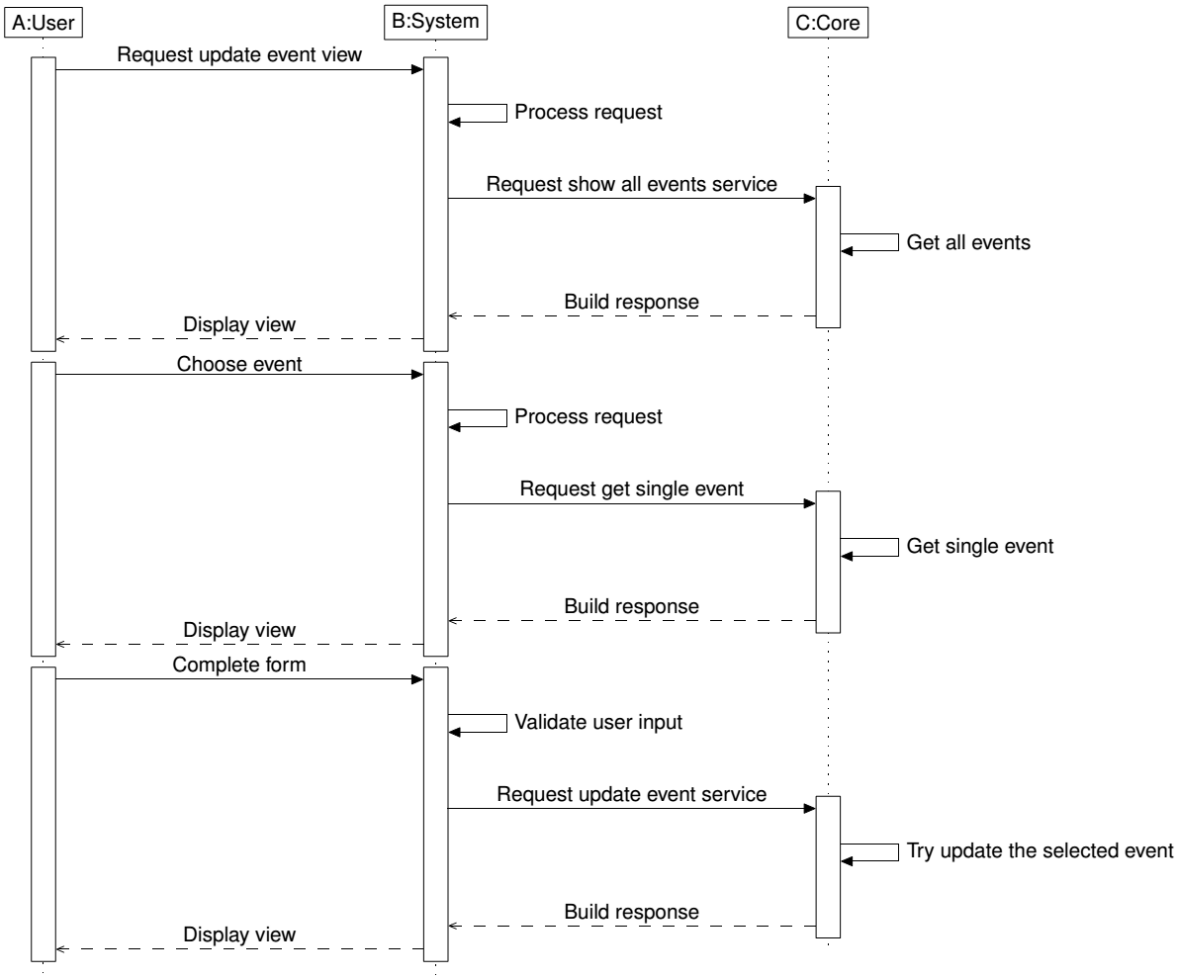


Diagrama 7. Actualizar información del evento.

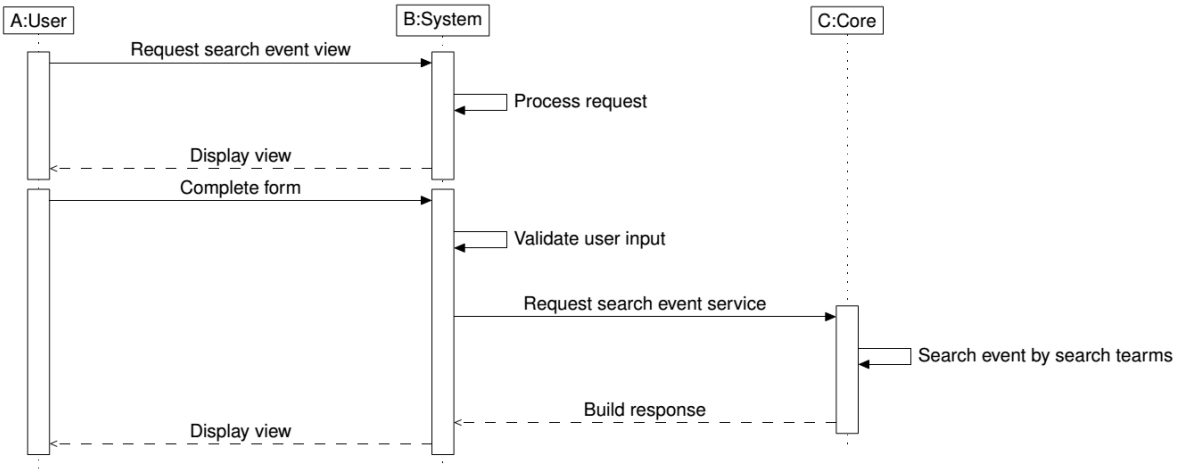


Diagrama 8. Buscar evento(s).

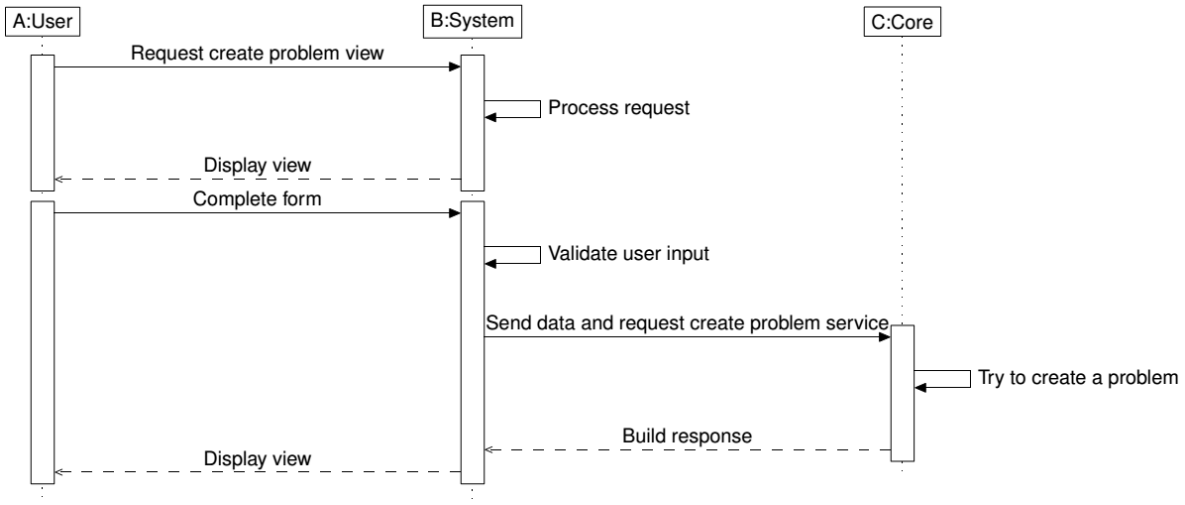


Diagrama 9. Crear problema

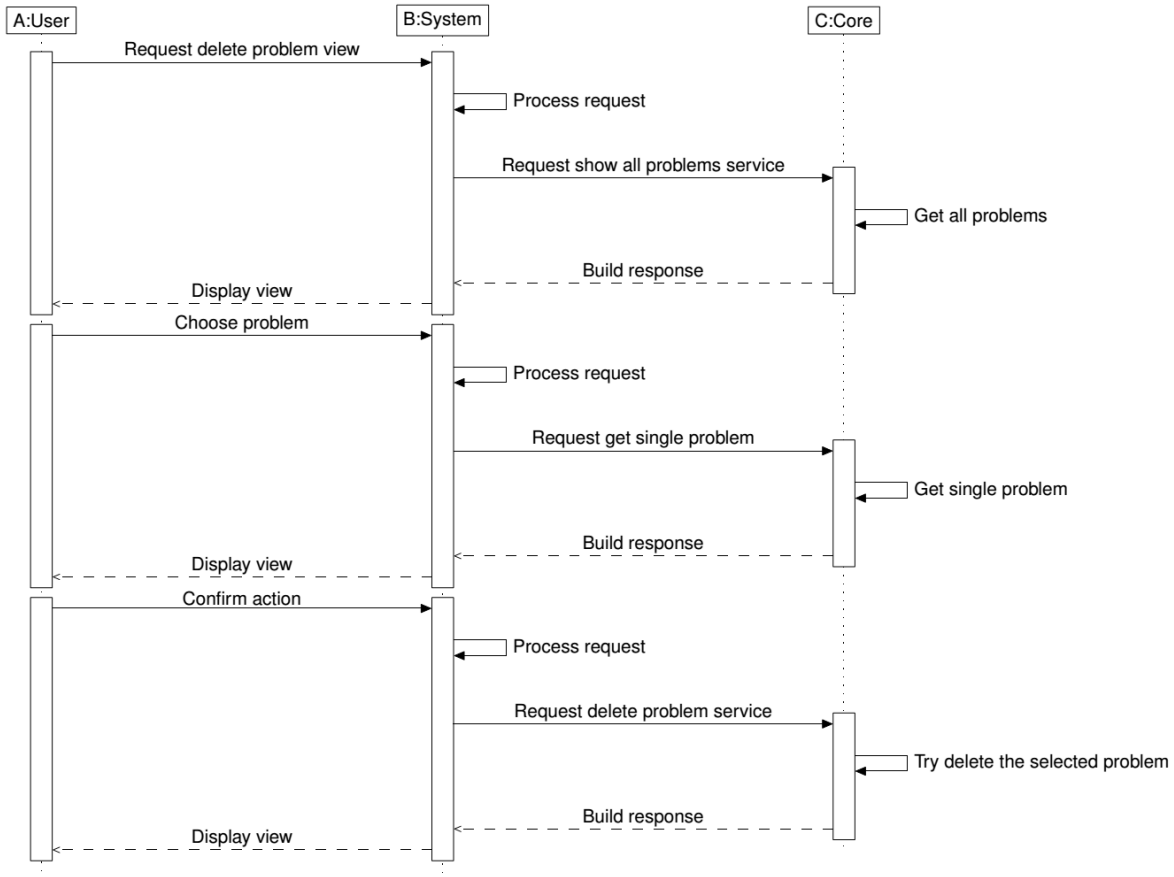


Diagrama 10. Eliminar o cancelar problema.

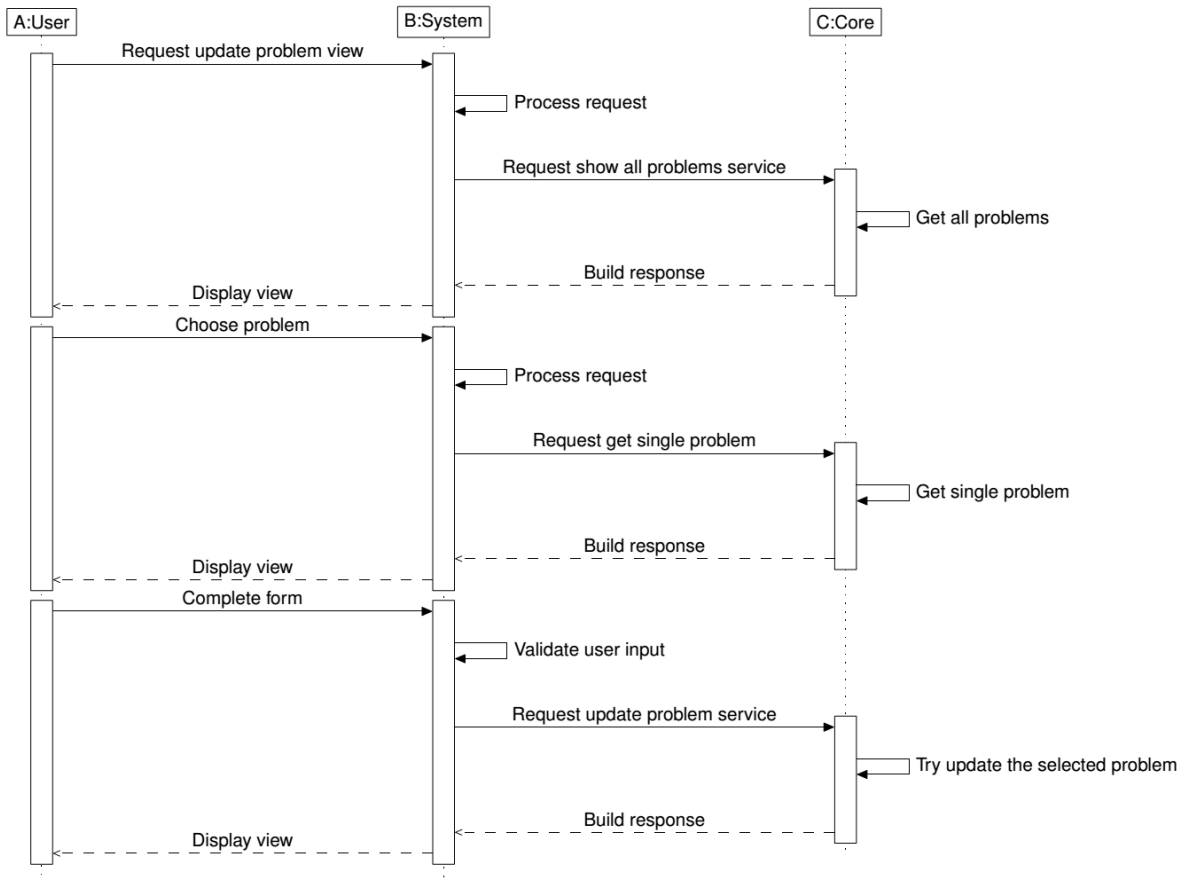


Diagrama 11. Actualizar problema.

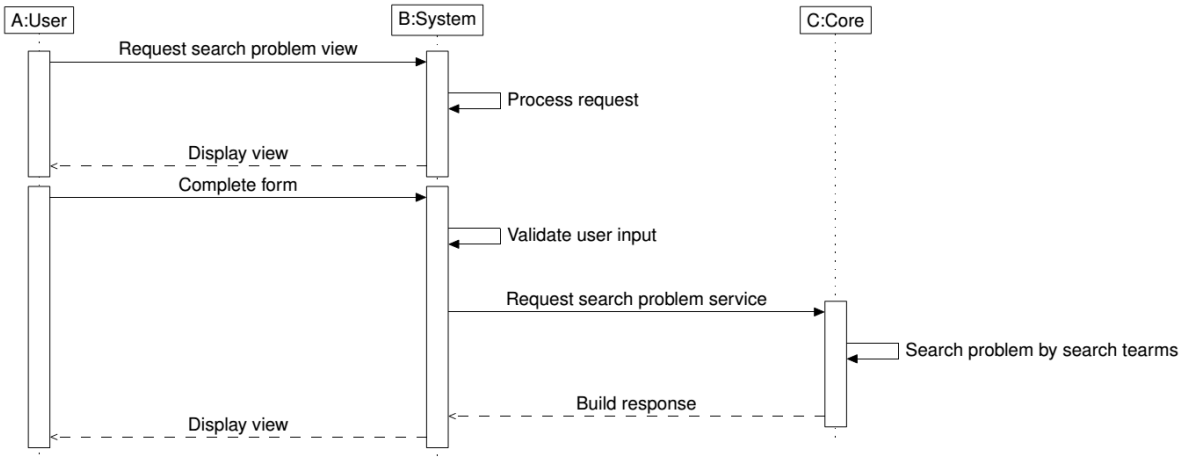


Diagrama 12. Buscar problema(s).

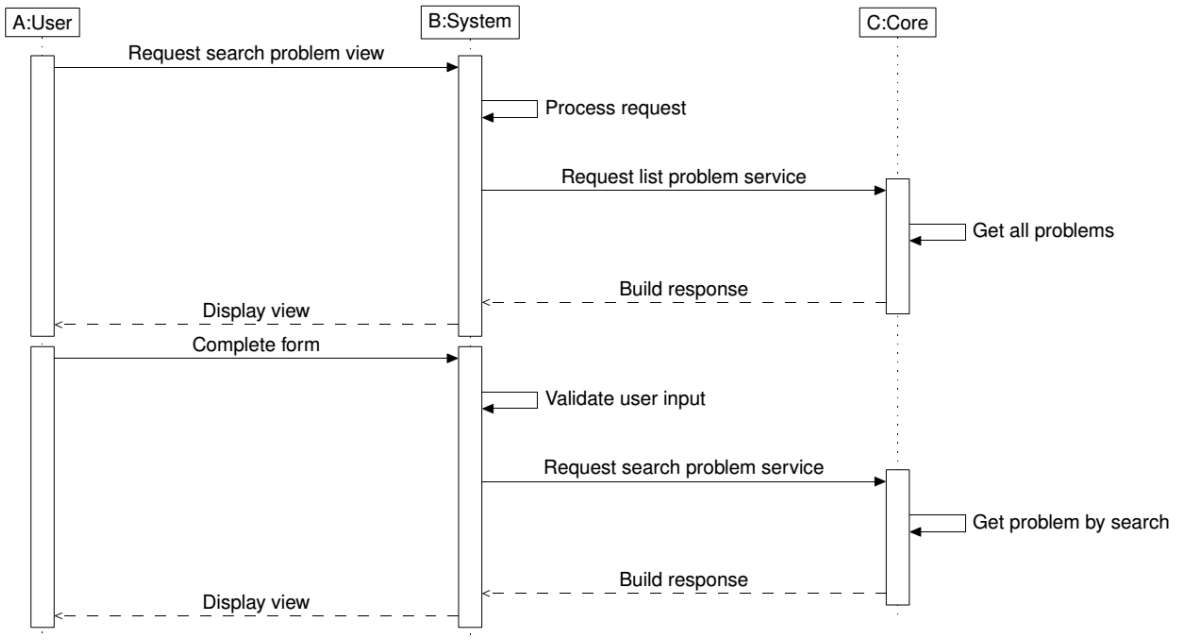


Diagrama 13. Mostrar detalles del problema.

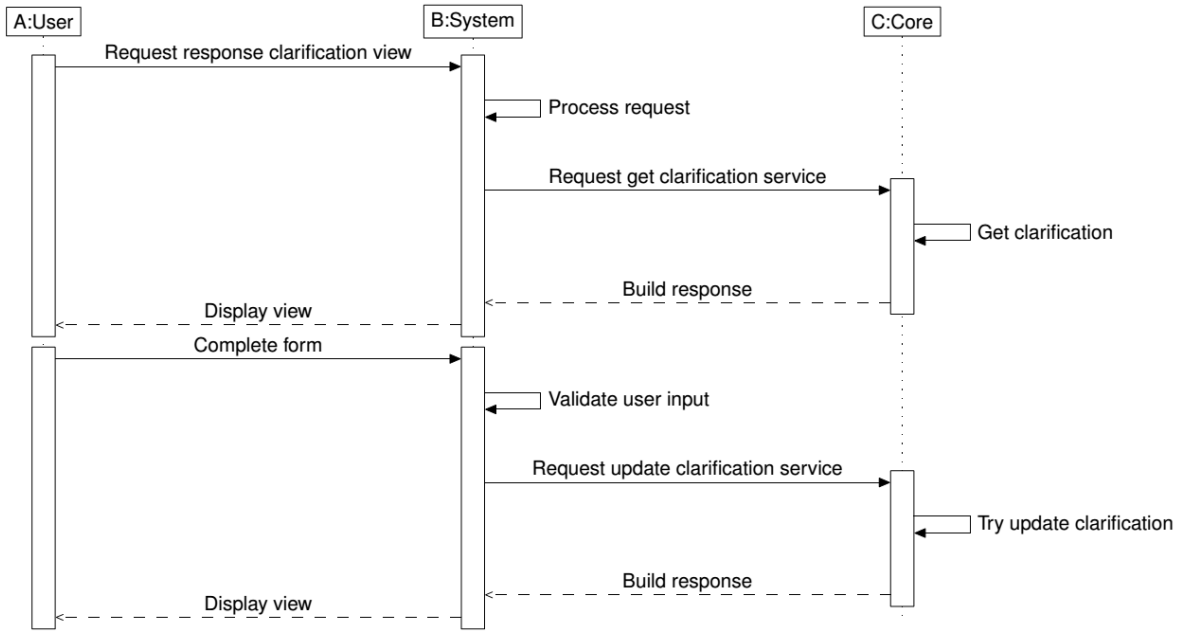


Diagrama 14. Responder aclaración.

Sección 3. Diagramas de secuencia dirigidos por el estudiante

La siguiente serie de diagramas representa los diagramas de secuencia que son iniciados por el rol estudiante.

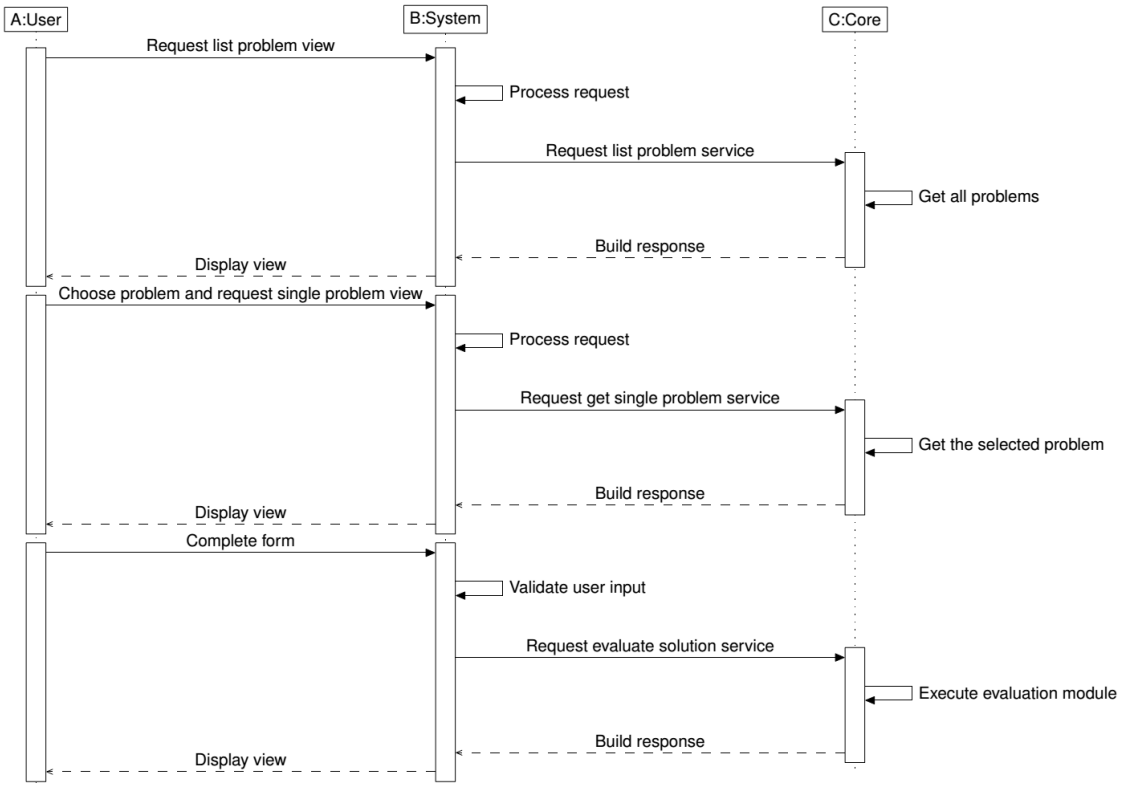


Diagrama 15. Publicar solución.

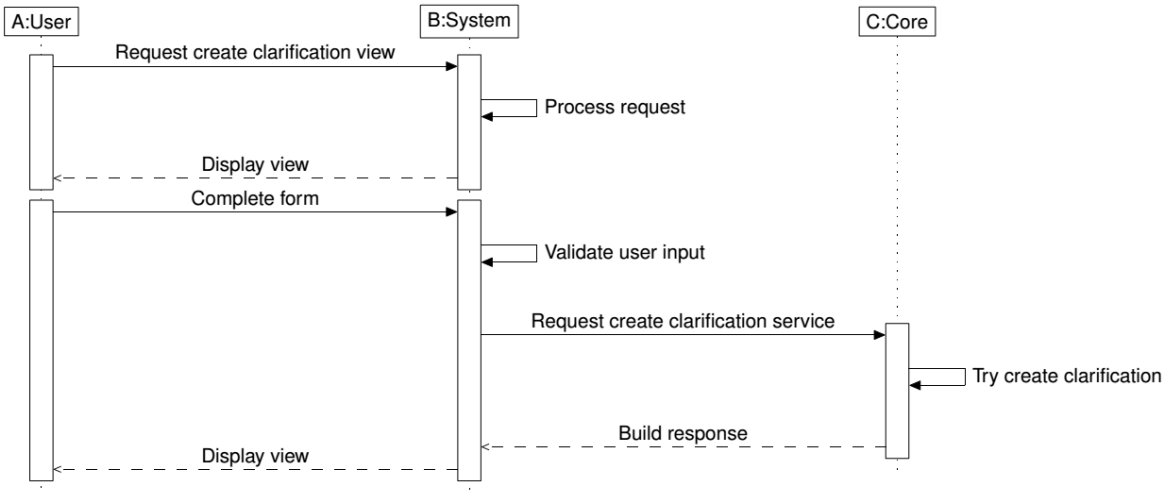


Diagrama 16. Solicitar aclaración.

El siguiente diagrama presenta las entidades que constituyen la capa de dominio.

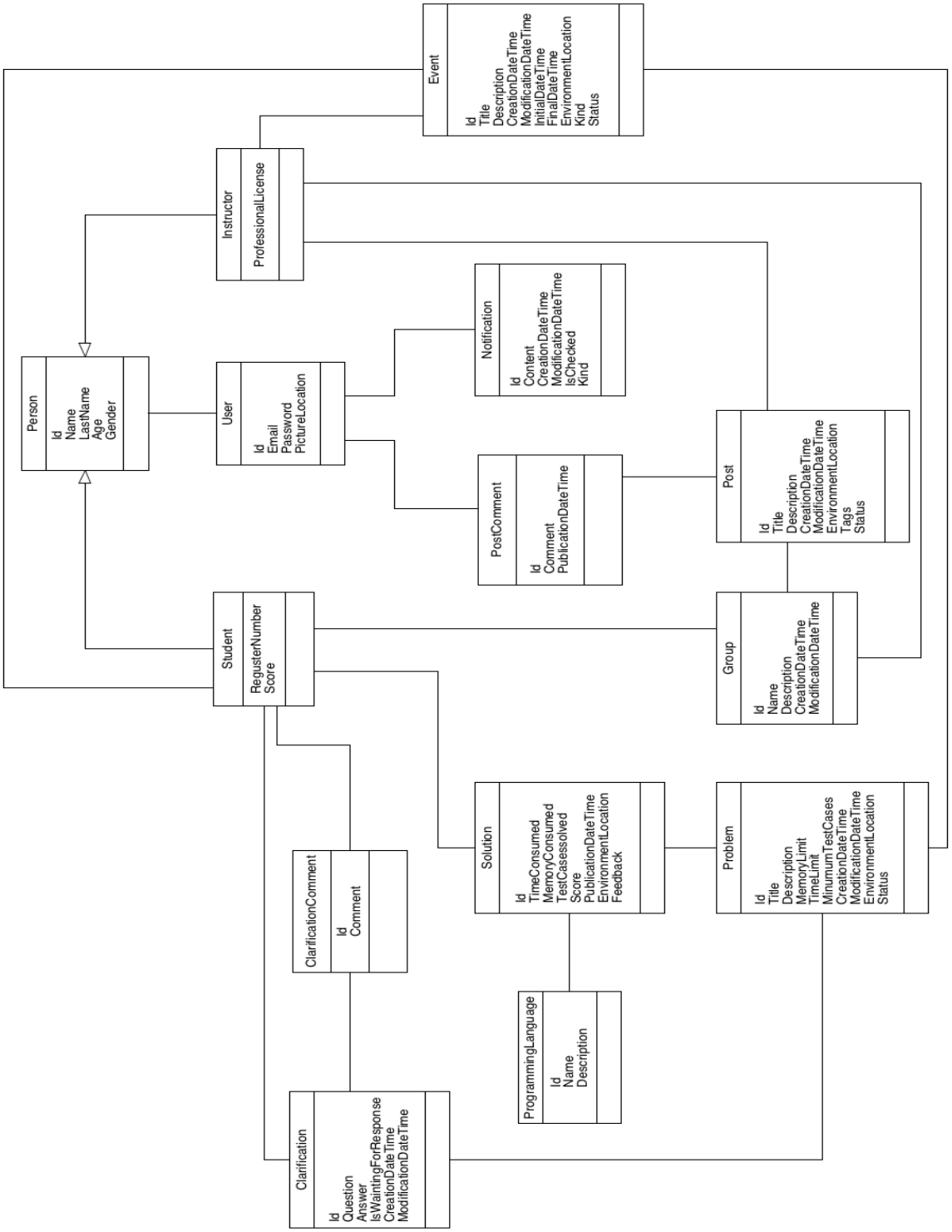


Diagrama 17. Diagrama de clases de la aplicación

El siguiente diagrama presenta las entidades que componen la base de datos del sistema.

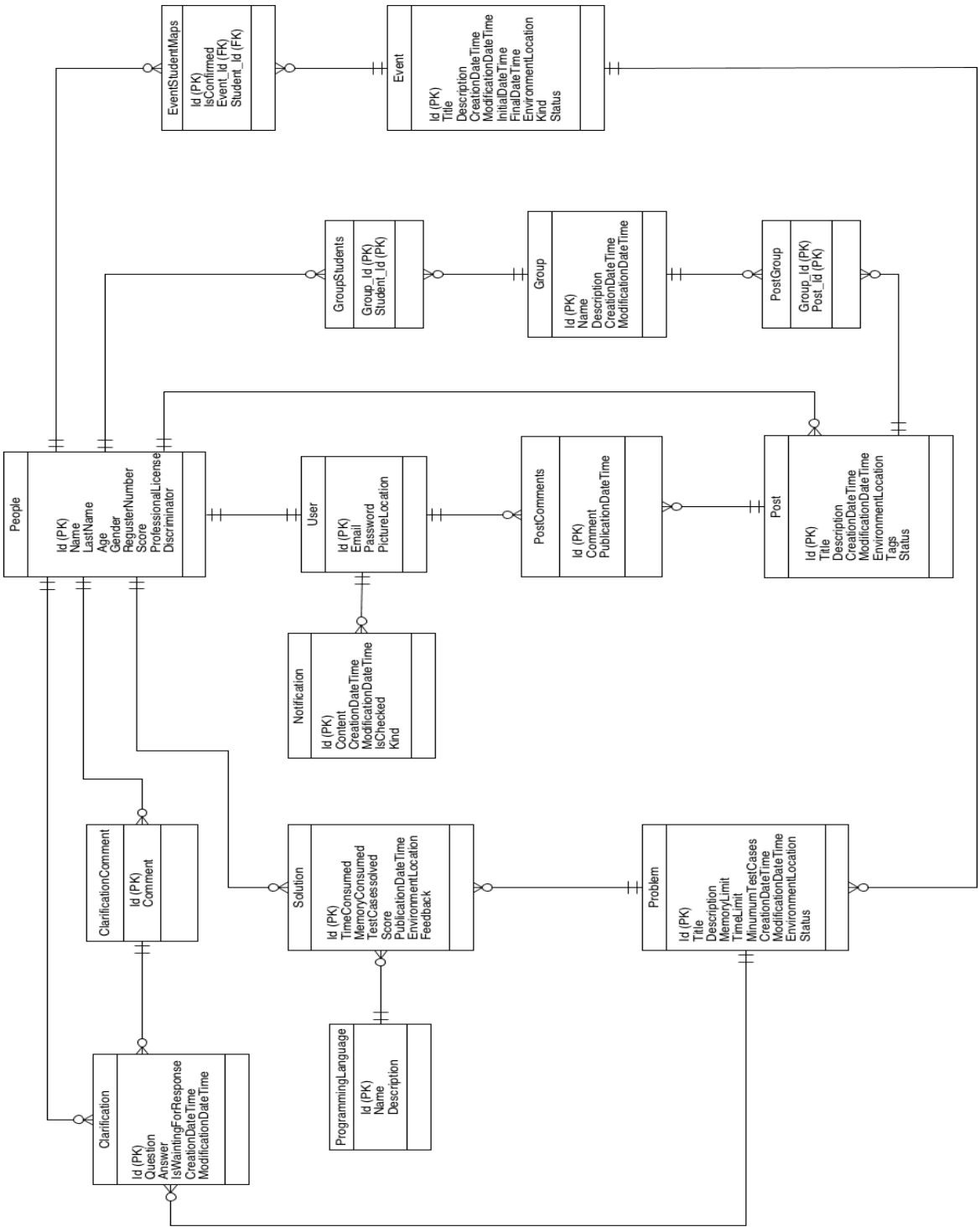


Diagrama 18. Diagrama de base de datos de la aplicación

Los principios de desarrollo ágil han permitido priorizar el valor del elemento realmente importante del ciclo de producción del proyecto: el software. Las metodologías tradicionales como RUP no son favorables para proyectos cuyo factor de riesgo determinante es el tiempo.

Haciendo uso de principios de desarrollo ágil, independientemente de la metodología usada es posible obtener un grado de flexibilidad que las metodologías tradicionales difícilmente permiten. Además fue posible generar un 'producto' con un diseño sencillo pero de gran calidad y con un desemepeño óptimo.

El desarrollo dirigido por pruebas permitió construir un sistema funcional y mantenible desde sus bases. Las pruebas unitarias son una competente herramienta de documentación y su mantenimiento ha sido tan importante como el de cualquier otro componente interno.

El enfoque *code first* permitió adaptar los mecanismos de gestión de datos a los cambios surgidos en los requisitos de los usuarios en contraste con los enfoque tradicionales mientras que la arquitectura de diseño modelo-vista-controlador fue lo suficientemente útil para concretar la comunicación entre los usuarios finales y el resto de la aplicación.

Conclusiones

Es importante entender que el avance de la tecnología y los medios de información deben tener un impacto es los métodos de aprendizaje actuales, ya que esto repercute en gran medida en la calidad de los profesionistas que se incorporarán en ambientes laborales, de investigación u otros sectores.

La metodología de evaluación implementada por este sistema tiene por objetivo presentar un cambio significativo respecto a aquellas que son aplicadas actualmente en materias de programación. Sin embargo, su efectividad se encuentra más allá del dominio del problema en el que se centra este trabajo y la eficiencia de sus resultados requieren de un estudio más detallado.

El análisis, diseño e implementación del sistema fue concretado mediante mecanismos flexibles intentando asegurar su calidad antes, durante y después de su producción representando un cambio exitoso con respecto a las metodologías y técnicas de desarrollo convencionales muy arraigadas en los niveles de formación profesionales.

Bibliografía

[1] W. S. Humphrey, Managing the Software Process (SEI), Addison Wesley Pub Co Inc, 1989.

[2] K. B. Acuña, «Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos [Edición electrónica gratuita],» 2009. [En línea]. Available: www.eumed.net/libros/2009c/584/.

[3] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland y D. Thomas, «Manifesto for Agile Software Development,» 2001. [En línea]. Available: http://agilemanifesto.org/.

[4] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, First ed., Addison-Wesley, 1994, pp. 2-4.

[5] J. Blankenship, M. Bussa y S. Millett, Pro Agile .NET Development with Scrum, Apress, 2011.

[6] IBM, [En línea]. Available: http://pic.dhe.ibm.com/infocenter/rsysarch/v11/index.jsp?topic=%2Fcom.ibm.sa.bpr.doc%2Ftopics%2Ft_ovwmdlprocs.html.

[7] J. Bender y J. McWherter, Test-Driven Development with C#, Wiley Publishing, Inc., 2011.

[8] «Software Testing Fundamentals,» [En línea]. Available: http://softwaretestingfundamentals.com/unit-testing/.

[9] C. De la Torre Llorente, U. Zorrilla Castro, M. A. Ramos Barroso y J. Calvarro Nelson, Guía de Arquitectura en N-Capas orientada al dominio con .NET 4.0, Krasis Press, 2010.

[10] A. Silberschatz, Fundamentos de Base de Datos, McGraw-Hill, 2002.

[11] J. Lerman y R. Miller, Programming Entity Framework: Code First, O'Reilly, 2012.

[12] A. Freeman, Pro ASP.NET MVC 5, Apress, 2013.

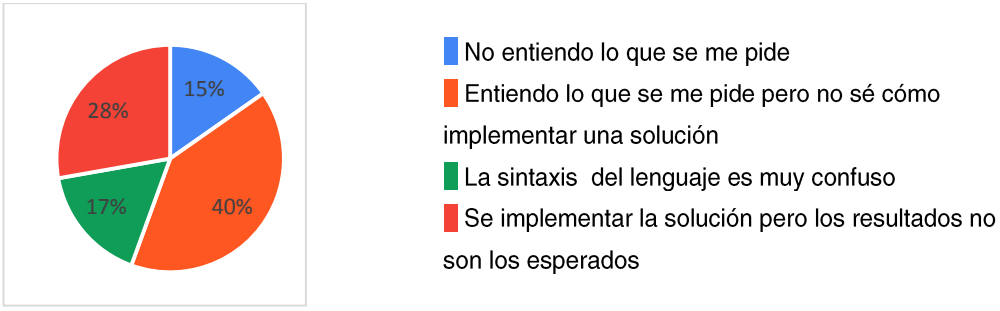
[13] G. Hall, Pro WPF and Silverlight MVVM, Apress, 2010.

[14] «Technische Universiteit Eindhoven,» [En línea]. Available: http://www.win.tue.nl/~wstomv/contests/acm-icpc-rules-summary.html.

Apéndice A

Una encuesta aplicada a estudiantes en los niveles medio y medio superior arrojo los resultados presentados en los siguientes gráficos.

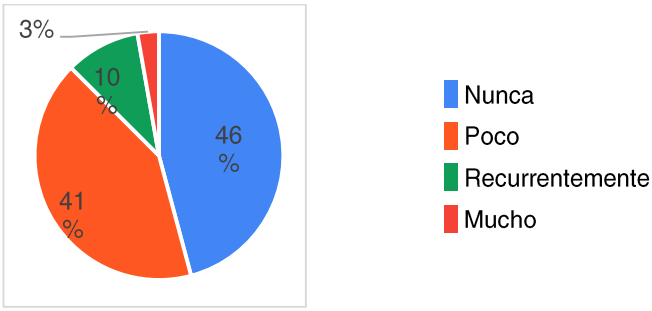
¿Cuáles son las principales dificultades que encuentras al intentar implementar un programa?



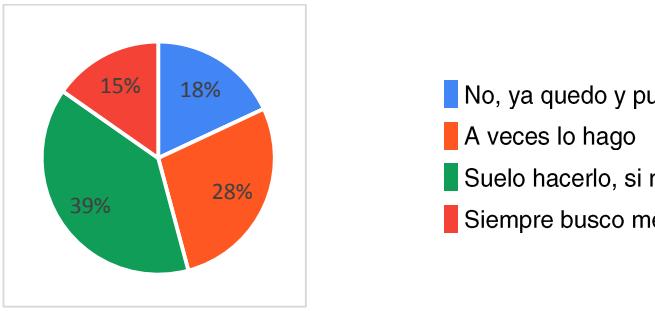
¿En qué medida te resulta útil realizar un examen escrito para reforzar tus conocimientos sobre programación?

¿En qué medida te resultan útiles exámenes prácticas de laboratorio para reforzar tus conocimientos sobre programación?

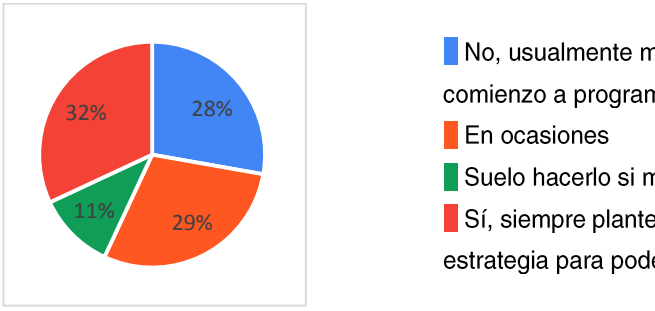
¿En los últimos meses qué tan recurrentemente has buscado en internet programas ya implementados para entregarlos como propios?



Una vez que logras implementar un programa ¿sueles volver analizar tu código fuente buscando mejorar tu implementación?



Antes de implementar un programa ¿Sigues alguna estrategia?



Antes de implementar un programa ¿Usas algún medio de información como libros o internet para basar en el tu implementación? (No estamos hablando de googlear la respuesta).