Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Cuadernos activos para la enseñanza de la programación

Tesis de Licenciatura en Ciencias de la Computación

Gisela Belén Confalonieri

Director: Dr. Rodrigo Castro
Codirector: Dr. Fernando Shapachnik
Buenos Aires, 2021

CUADERNOS ACTIVOS PARA LA ENSEÑANZA DE LA PROGRAMACIÓN

En el estado de la práctica actual, el material producido y/o utilizado por docentes para dar clases de programación suele constar principalmente de información expuesta estáticamente, acompañada por el uso de herramientas para la ejercitación de los conceptos. Ante la necesidad de una herramienta dinámica e interactiva que presente de forma integrada el material generado por el docente y las actividades intervenidas por los estudiantes, surgen los Cuadernos Activos como documentos holísticos en los que los contenidos progresan evolucionando en el tiempo acorde a decisiones tomadas por el docente respecto a cada estudiante.

En esta tesis implementamos una clase de Cuadernos Activos basada en los llamados Cuadernos Computacionales, los cuales permiten combinar secciones de texto, código, y resultados de ejecuciones en un mismo documento dinámico.

Trabajando con Jupyter Notebook y aprovechando la posibilidad de extender su funcionamiento, presentamos un mecanismo para analizar las entradas ingresadas por un estudiante, evaluarlas de manera tanto individual como contextual, y ofrecer feedback automatizado en forma de mensajes y/o contenidos subsiguientes que sean función de la trayectoria previa del Cuaderno Computacional.

Finalmente desarrollamos un prototipo que integra la aplicación educativa Pilas Bloques con la especificación Learning Tools Interoperability (LTI), mostrando una forma de incrustar una aplicación externa en el Cuaderno Computacional, que además permite el intercambio sistematizado de calificaciones de ejercitaciones entre ambas herramientas.

Palabras claves: Cuadernos Activos, enseñar programación, Jupyter Notebook, Pilas Bloques, LTI.

AGRADECIMIENTOS

Este trabajo es producto de un gran esfuerzo personal y de un excelente y amoroso acompañamiento de personas maravillosas. Agradezco a mis directores por su paciencia, enseñanzas, y apoyo infinitos. A Jorge por darme aliento desde el primer día de CBC. A compañeres y amigues que fui encontrando en el camino por el aguante, las horas de estudio, y los mates con pastafrola. A mi hija por comprender cuando mamá tiene que estudiar, y a quienes me apoyan constantemente en los cuidados. A las Mujeres del DC por la escucha y ese empujoncito cuando hizo falta. Al Departamento de Computación por ser mi segunda casa durante tantos años, y a la Universidad de Buenos Aires por darme la oportunidad de soñar con un título de grado.

Dedicado a Mei

Índice general

1.. Introducción 1
1.1. Propuesta Cuadernos Activos (marco de la tesis) 1
1.2. Estado del arte 1
1.2.1. Cuadernos Computacionales 2
1.2.2. Sistemas de Gestión de Aprendizaje (LMS) 6
1.2.3. Estándares y especificaciones para contenido digital 6
1.3. Enfoque y alcance de este trabajo 12

2.. Arquitectura base 15
2.1. Jupyter Notebook 15
2.1.1. Extensiones frontend y backend 15
2.1.2. JupyterHub 16
2.2. Sistema de corrección e indicador de próximo ejercicio 17
2.2.1. Implementación 18
2.3. Propuesta de diseño: aprovechando las extensiones de Jupyter Notebook . . 18

3.. Pruebas de concepto simples: secuencias de ejercicios 21
3.1. Corrección sin seguimiento de actividad 21
3.2. Secuencias de ejercicios con seguimiento de actividad 22

4.. Pruebas de concepto avanzadas: integración con aplicaciones 27
4.1. Motivación: Integración con *Pilas Bloques* 28
4.2. Elección de LTI 29
4.3. LTI 1.0 (Basic LTI) 29
4.3.1. Idea 29
4.3.2. Implementación 31
4.3.3. Resultado de la POC 31
4.4. LTI 1.1 32
4.4.1. Idea principal 33
4.4.2. Implementación 35
4.4.3. Resultado de la POC 36

5.. Conclusiones y trabajo futuro 39
5.1. Conclusiones 39
5.2. Trabajo Futuro 40

1. INTRODUCCIÓN

1.1. Propuesta Cuadernos Activos (marco de la tesis)

En la práctica actual, el material producido y/o utilizado por docentes a la hora de dar clases suele constar de información expuesta estáticamente (en apuntes, presentaciones, etc) que los estudiantes utilizan, ya sea pasivamente o crítica y activamente, y el uso de herramientas adicionales al texto explicativo para la ejercitación de los conceptos. Dentro de estas herramientas adicionales podemos mencionar, por ejemplo, páginas web interactivas y plataformas de aprendizaje electrónico o *e-learning*. Las páginas web interactivas permiten a los estudiantes realizar actividades interactivas en línea; dichas actividades son provistas por el sitio y los docentes deben seleccionar las que considere apropiadas e indicar al estudiantado cuáles resolver, en general, sin facilidades para realizar un seguimiento. Los sistemas de aprendizaje electrónico resuelven parte de estos inconvenientes en cuanto a selección, asignación y seguimiento de actividades, aunque puede requerir la utilización de distintas herramientas para producir actividades completas, por ejemplo, separando el material teórico de las actividades prácticas.

Dado este contexto, sería ideal contar con una herramienta dinámica e interactiva que cubra estas necesidades. Definimos entonces a los Cuadernos Activos (CAs): **un CA es un documento digital que permitiría presentar la información teórica y las actividades prácticas formando un todo; la estructura y los contenidos progresarían en el tiempo con base en decisiones que el docente toma respecto a cada estudiante, y los estudiantes lo intervendrían directamente con sus producciones.** Es decir, con los CAs se busca ofrecer la información y proveer espacios para ejercitación en el mismo documento, además de favorecer el trabajo colaborativo y la comunicación entre estudiantes y docentes. Esto es, a su vez, motivador para los estudiantes, ya que los involucra en el proceso de aprendizaje al ver los resultados de sus ejercicios sin tener que cambiar de vista constantemente. Con esto, llevado a la enseñanza de la programación, cada estudiante podría producir sus programas, ejecutarlos, observar los resultados y reportar sus análisis en la misma interfaz digital, pudiendo experimentar libremente en un entorno semiestructurado por el docente, por ejemplo, modificando parámetros para comprender temas como funciones matemáticas o algoritmos de ordenamiento. Además, si bien los docentes siguen apareciendo como los "transmisores de conocimiento", con la aproximación que proponemos, la interacción colaborativa docentes-estudiantes-CAs puede tender a horizontalizar el proceso de enseñanza-aprendizaje.

1.2. Estado del arte

En primera instancia, realizamos una revisión de las herramientas existentes al momento con un funcionamiento y/o intención alineados al proyecto.

Decidimos separar las herramientas halladas en tres grupos, a saber:

- **Cuadernos Computacionales:** son aquellas herramientas que se acercan al concepto de Cuaderno Activo en cuanto a la forma en que se muestra el contenido y se trabaja sobre el mismo.

- **Sistemas de gestión de aprendizaje:** este grupo está conformado por plataformas que se orientan a la administración de cursos en cuanto a asignación de contenidos y tareas, calificación y seguimiento.

- **Estándares y especificaciones para contenido digital:** aquí se presentan las especificaciones existentes y en desarrollo para la estandarización de contenido digital en cuanto a interoperabilidad, seguimiento de actividad, y analíticas de datos.

1.2.1. Cuadernos Computacionales

Un *cuaderno computacional* es una colección de bloques o *celdas* ordenadas, donde cada celda encapsula texto, código y/o resultados de los cómputos. Generalmente, estos entornos imitan a los cuadernos de papel, e incluyen texto, cálculos y visualizaciones (Figura 1.1). A menudo, un cuaderno es una interfaz para un motor de cómputo aparte; el cuaderno no ejecuta por sí mismo los cómputos definidos en las celdas de código, sino que los envía a un *kernel* que computa y devuelve los resultados para ser mostrados en la interfaz [1].

Estos cuadernos plantean un nuevo modo de enseñar y aprender. Se han explorado diversos usos de estos cuadernos computacionales en el aula, como utilizar cuadernos (en lugar de presentaciones) con un proyector digital para enriquecer lecciones y discusiones. A su vez, utilizar cuadernos como lecturas para fuera del aula encaja muy bien con la noción de *aula invertida* [2].

Por otro lado, requerir a los estudiantes trabajar con cuadernos computacionales es un método práctico para introducir la computación alfabetizada o *literate computing*. *Literate computing* es un paradigma de computación que entreteje código ejecutable con prosa, imágenes, vídeo, y otros medios digitales, y donde el código coexiste en el mismo espacio que su salida o *output* y los objetos y datos con los que opera. Recientemente ha tenido más aceptación particularmente en la comunidad de ciencia de datos con herramientas como los cuadernos computacionales (aunque cabe mencionar que estos cuadernos son sólo una expresión particular de *literate computing* [3]. El término *literate computing* deriva del *literate programming* presentado por Knuth en [4] pero mientras que el objetivo de Knuth es hacer una herramienta para documentar código "para programadores de sistemas, no para estudiantes de secundaria o aficionados", los entornos de *literate computing* persiguen la creación colaborativa de narrativas computacionales reproducibles que puedan ser usadas por un amplio rango de audiencias y contextos [5]. Así, los objetivos del *literate computing* se alinean muy bien con los objetivos de escritura de los estudiantes en un entorno educativo [6].

A continuación, se enumeran los cuadernos computacionales analizados.

- **Jupyter Notebook** [8]

  Se trata de una aplicación web de código abierto que permite crear y compartir documentos con código vivo, ecuaciones, visualizaciones (HTML, LaTeX, PNG, SVG, videos, etc.) y texto, formando una rica narrativa computacional. Actualmente soporta más de 100 lenguajes de programación [9], incluyendo Python, Julia, R y Scala, y es posible extenderlo a más lenguajes. Se accede a través de cualquier navegador web moderno, pudiendo correr sobre una aplicación local o en un servidor remoto. Los documentos pueden ser compartidos por mail, Dropbox [10], GitHub [11] y NbViewer [12]. Maneja autenticación y da lugar a implementar distintos métodos de

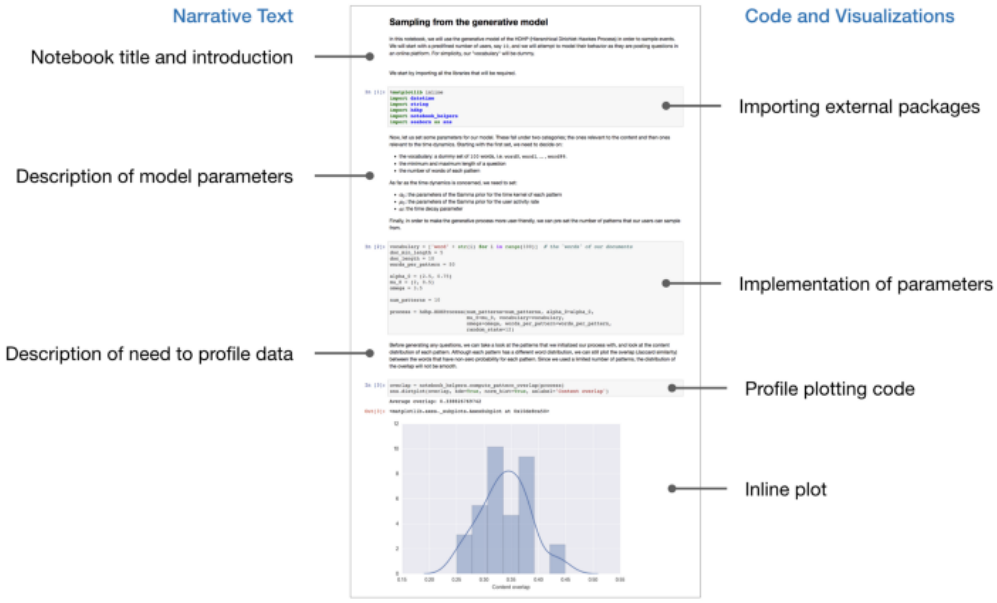


Fig. 1.1: Un ejemplo de cuaderno computacional combinando código de análisis de datos y visualizaciones con texto explicativo [7]

autenticación. Permite editar texto utilizando el lenguaje de marcado Markdown [13]; editar código desde el navegador con resaltado de sintaxis automático, indentación, y autocompletado; y permite ejecutar dicho código desde el navegador y visualizar directamente los resultados, pudiendo representar estos cómputos de diversas maneras, como tablas o gráficos. Además, se pueden utilizar bibliotecas específicas que permiten importar imágenes o vídeos, incluir enlaces a páginas web o incluso incrustar una página web completa dentro del documento. A su vez, se puede exportar el cuaderno como una presentación de diapositivas HTML, pudiendo ejecutar código en ellas. El lenguaje de programación está determinado por el *kernel* utilizado para el documento, y aunque no es posible utilizar más de un lenguaje dentro del mismo documento, se puede instalar el *kernel* SoS [14] que permite inlcuir diferentes *kernels* en un mismo cuaderno.

Además, tiene la posibilidad de agregarle extensiones para enriquecer el funcionamiento del cuaderno. Dentro de estas extensiones se encuentra *nbgrader* [15], una herramienta para crear y calificar tareas. La misma permite a los instructores crear una única copia maestra de una tarea, incluyendo pruebas y soluciones, a partir de la cual se genera una versión para estudiantes sin las soluciones, evitando tener que mantener dos versiones separadas. Además, *nbgrader* permite calificar automáticamente las tareas enviadas, y a su vez los instructores pueden calificar manualmente las respuestas libres y dar feedback personalizado para cada entrega.

nbgrader puede usarse también con *JupyterHub* [16], una instalación centralizada que permite a los usuarios iniciar sesión, crear cuadernos, y ejecutar código en un servidor remoto. Esto significa que el sistema puede ser preparado una única vez para todos los estudiantes que lo vayan a utilizar. Así, luego de crear una tarea, los instructores pueden distribuirla a los estudiantes, quienes pueden traerse una copia

de la tarea directamente a través de la interfaz de Jupyter Notebook server. Los estudiantes pueden enviar sus tareas ya completas a través de la misma interfaz, para que pase a estar disponible para los instructores.

- **Cuadernos Wolfram** [17]

  Los cuadernos computacionales Wolfram permiten combinar texto, gráficos y código vivo. A su vez, ofrece acceso directo a los recursos que se necesiten estableciendo conexiones con lenguajes externos, servicios, programas, formatos y dispositivos. Estos documentos son creados y manipulados utilizando el lenguaje Wolfram, que a su vez incorpora maneras de conectar a una base de conocimiento Wolfram, Wolfram Cloud [18] y otras tecnologías Wolfram. Cuenta con diversas interfaces: escritorio, web, mobile, todas integradas y sin necesidad de configurar compiladores, servidores o algún otro sistema externo. También ofrece una variedad de formas de compartir los documentos, de manera pública y privada. El servicio es pago, y provee una versión de prueba gratuita durante 14 días.

- **CDF - Formato de Documento Computable** [19]

  El Formato de Documento Computable es un tipo de documento electrónico de la compañía Wolfram Research. Ofrece una plataforma de visualización interactiva de contenido que usa el *kernel* de Mathematica [20], el cual permite realizar cálculos matemáticos avanzados en tiempo real. Está diseñado para crear fácilmente documentos mediante una interfaz gráfica, permitiendo al usuario cambiar los parámetros de una función definida en el entorno de Mathematica dinámicamente.

  Podría considerarse que estos documentos compiten con los documentos PDF estáticos utilizados para presentaciones y artículos con enfoque técnico. Un caso particular sería su utilización para crear material didáctico para distintos niveles educativos, especialmente en áreas como matemáticas, física y biología [21].

  Los CDF pueden crearse con Mathematica, o con cualquier otro producto Wolfram de creación de contenidos, y pueden ser implementados como objetos incrustados dentro de páginas web, o visualizados como documentos en pantalla completa en un navegador mediante Wolfram Cloud [18]. Para implementarlo en el software Wolfram CDF Player gratuito se cuenta con dos opciones de licencia, gratuita o paga.

- **Whiteboards o pizarras *online***

  El concepto de *whiteboard* o pizarra *online* se refiere a un espacio de trabajo colaborativo para compartir notas, imágenes, dibujos y conversaciones a través de Internet. La idea es más bien parecida a la de un pizarrón interactivo, y no a la de Cuaderno Activo. Con estas pizarras un instructor puede explicar contenido de manera remota y sincrónica a sus estudiantes, como si estuvieran en un aula convencional. El objetivo es facilitar a instructores y estudiantes la preparación y presentación de material didáctico en la Web en tiempo real, permitiendo la discusión en torno a un tema mientras se dibuja en la pizarra con texto, colores y formas [22].

  Algunas de estas herramientas son: Educreations [23], ShowMe [24] y Doceri [25]. Incluso se utilizan las herramientas de Google Drive [26].

- **Google Colaboratory** [27]

Google Colab es un documento ejecutable que permite escribir, ejecutar y compartir código dentro de Google Drive. Básicamente, es un Jupyter notebook almacenado en Google Drive. Colab ejecuta el notebook directo en la nube, de modo que se puede ejecutar código Python sin ninguna configuración local. Además de poder utilizar toda la funcionalidad que ofrece Python, también pueden utilizarse bibliotecas integradas en Colab con fragmentos de código que pueden insertarse para crear visualizaciones interactivas de los datos. Los notebooks de Colab pueden compartirse de la misma manera que un Google Doc dentro de Google Drive, e incluso pueden exportarse a GitHub. Por otro lado, los notebooks se almacenan en el formato estándar de Jupyter notebook, por lo cual pueden ser vistos y ejecutados en Jupyter Notebook y cualquier entorno compatible.

- **OpenModelica Notebook** [28]

  Es un subsistema de OpenModelica [29], un entorno de código abierto de modelado, compilación y simulación sobre el lenguaje de modelado y simulación orientado a objetos Modelica [30], que se utiliza junto al sistema DrModelica para la enseñanza y el aprendizaje de Modelica de manera interactiva. DrModelica consta de una serie de cuadernos electrónicos interactivos conteniendo texto explicativo, ejemplos, ejercicios y respuestas. Estos cuadernos son documentos que pueden contener texto, código, enlaces, video y visualizaciones, y en el caso de DrModelica todos los documentos pueden ser intervenidos, agregando, cambiando o eliminando cualquier parte, además de que el usuario puede crear un cuaderno nuevo con sus producciones. A su vez, se ha trabajado en una generalización de OMNotebook para que pueda manejar otros lenguajes de programación con propósito educativo [31].

- **Apache Zeppelin Notebook** [32]

  Apache Zeppelin es un proyecto de código abierto orientado al análisis de grandes datos para sistemas de procesamiento distribuido. Es una herramienta basada en web, con un estilo de cuaderno y sesiones compartidas de análisis colaborativo entre usuarios, que busca cubrir todo el flujo de descubrimiento de datos dentro de un solo cuaderno.

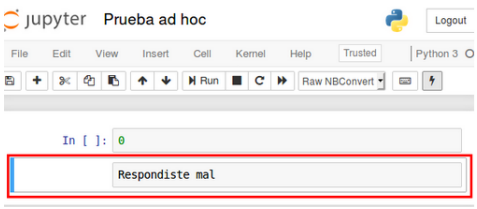
  Tiene una arquitectura de *backend* que permite conectar distintos intérpretes vía una interfaz, e integración con Apache Spark [33]. Un cuaderno Zeppelin puede contener diferentes párrafos, cada cual con su propio lenguaje/intérprete.

- **Sage Worksheets** [34] y **CoCalc** [35]

  Sage Worksheets busca crear un entorno colaborativo tipo cuaderno para SageMath [36], un software matemático gratuito de código abierto. Sage Math puede ser utilizado en varias formas: en una computadora local, en una red local y en la nube. Sage Worksheets corresponde a este último caso, ejecutando en la plataforma CoCalc (antes conocido como SageMathCloud).

  CoCalc provee un entorno de cómputo y colaboración *on line* sin necesidad de instalar y ejecutar el software, sólo hace falta tener una cuenta e iniciar sesión.Se basa en la creación de proyectos, cada uno con sus archivos correspondientes, y permite el trabajo colaborativo en tiempo real. Cada archivo se abre con un editor online correspondiente, ya sea Sage Worksheet, Jupyter Notebook, o un documento LaTeX, además de dar lugar a operar en una terminal Linux, ejecutar aplicaciones gráficas

o editar texto plano. Provee una sala de chat, lo cual sumado a la colaboración en tiempo real lo vuelve particularmente útil para la enseñanza. A este respecto, CoCalc provee la posibilidad de gestionar cursos, permitiendo distribuir, obtener, calificar y enviar tareas, y hacer un seguimiento de cada estudiante en tiempo real. Ofrece un plan gratuito y un plan de suscripciones pagas, que difieren en la quota de memoria y espacio de disco otorgado.



1.2.2. Sistemas de Gestión de Aprendizaje (LMS)

Un sistema de gestión de aprendizaje o *Learning Management System* (LMS) es una infraestructura para la entrega y administración de contenido de educativo que permite identificar y evaluar el aprendizaje o las metas de capacitación individuales y organizacionales, seguir la trayectoria hacia el cumplimiento de esas metas, y recopilar y presentar datos para supervisar el proceso de aprendizaje de una organización en su conjunto. Un LMS ofrece contenido pero también maneja el registro y la administración de cursos, el seguimiento y los informes, proveyendo una estructura para el proceso de aprendizaje completo dentro de una organización [37].

Existen LMSs desarrollados en la nube, donde un proveedor mantiene el sistema y los estudiantes y colaboradores se autentican con nombre y contraseña, no hace falta instalar ningún software pero no suelen ser personalizables. Otros LMSs requieren descargar el software, y suelen permitir mucha más personalización, pero en general son pagos y requieren conocimiento previo específico. Algunos LMSs se instalan en el escritorio y/o pueden ser accedidos desde múltiples dispositivos y facilitan el trabajo colaborativo.

Los LMSs que son de código abierto suelen ser también gratuitos y online, permiten modificar y redistribuir el producto para adaptarse a las necesidades en cada caso y en muchos casos cuentan con una comunidad online activa donde se puede acudir en caso de necesitar ayuda o encontrar problemas, pero suele requerir personal con una base de conocimientos específica para implementar el software. Otros LMSs son privativos y requieren una suscripción o licencia paga mensual o anual, pero suelen ofrecer soporte dedicado y características más amigables para los usuarios.

Algunos de los LMSs más populares son Open edX [38], Blackboard [39], Edmodo [40], Moodle [41], Chamilo [42] y Canvas [43], pero es un área en constante crecimiento y repleto de opciones. Un caso particular es Google Classroom [44], uno de los servicios de Google en el plan GSuit para Centros Educativos [45]; la plataforma es similar a otras aplicaciones de Google, también se integra con otras herramientas Google, y cuenta con una aplicación *mobile* [46].

1.2.3. Estándares y especificaciones para contenido digital

Definiremos primero un objeto de aprendizaje (OA). Un OA es el componente más pequeño de contenido dentro de un LMS, con el potencial de reutilización en múltiples

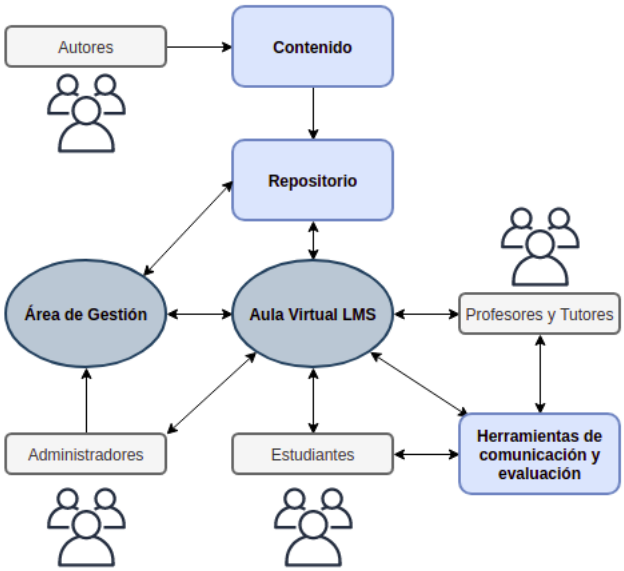


Fig. 1.2: Características de los LMS [47]

contextos, pudiéndose adaptar a múltiples necesidades y múltiples audiencias. Crear OAs tan adaptables y escalables puede ser difícil (aunque podría utilizarse cualquier medio digital para apoyar el aprendizaje), y para maximizar la reutilización los OAs deben adherir a ciertos estándares. La aplicación de dichos estándares es lo que resulta en una implementación más fácil dentro de los LMS [37].

La intención de estos estándares es, entonces, agilizar y unificar la manera de diseñar e integrar contenido digital en todas las plataformas, ya que en caso contrario implicaría mayor esfuerzo de desarrollo para que, por ejemplo, un LMS pueda integrar más de una herramienta web o que un mismo OA sea utilizable en LMSs diferentes. A la fecha, los sistemas de gestión de aprendizaje implementan al menos uno de estos estándares.

En este trabajo, decidimos clasificar estas especificaciones según el siguiente criterio: aquellas que se enfocan en la **interoperabilidad** entre LMSs y otras herramientas, y aquellas que se enfocan en el **seguimiento** de la actividad de cada estudiante con el fin de poder obtener métricas o **analíticas** sobre las cuales tomar decisiones.

Interoperabilidad

No hay una definición única y consensuada de interoperabilidad. En el caso más general, en el contexto de las tecnologías de la información, la interoperabilidad es la habilidad de trasferir e interpretar datos e información a través de sistemas, aplicaciones, o componentes [48]. Para ello, los sistemas deben poder conectarse entre sí, a menudo a través de una interfaz explícita y acordada, de manera que la parte receptora pueda entender los datos, y que los humanos a cada lado del intercambio de información entiendan y puedan actuar sobre ese intercambio. Por este motivo, ninguna definición corta de interoperabilidad termina siendo satisfactoria: la interoperabilidad es muy específica del contexto. Uno de los beneficios principales de la interoperabilidad es permitir el trabajo conjunto y eficiente entre sistemas diversos.

Para este trabajo, analizamos dos especificaciones existentes para interoperabilidad de sistemas relacionados con el aprendizaje: **LTI** (Figura 1.3) y **SCORM** (Figura 1.4).

- **LTI** [49]

  El estándar LTI (*Learning Tools Interoperability*), desarrollado por el IMS Global Learning Consortium, provee a desarrolladores y administradores de sistemas un mecanismo estándar para integrar aplicaciones de aprendizaje (referidas como *tool provider*) con sistemas ya existentes, como los LMSs (referidos como *tool consumer*) y permitir a los instructores mayor libertad para seleccionar la que mejor se corresponde con sus necesidades pedagógicas [50].

  El núcleo de LTI es establecer una conexión segura y confirmar la autenticidad de la herramienta, y establece extensiones opcionales que añaden características como el intercambio de actividades y calificaciones entre la herramienta y el LMS. Esto extiende el conjunto de características de un LMS mediante el uso de herramientas y actividades por fuera de la plataforma y sin necesidad de que el desarrollador de la actividad tenga que crear y mantener integraciones individuales para diversos LMSs. El desarrollador sólo necesita implementar el estándar IMS LTI y proveer los detalles de conexión a quien lo quiera utilizar. Los instructores no deben pedir nuevas integraciones en sus sitios, sólo hace falta que conecten la herramienta al estilo *plug and play* [50]. Además, los estudiantes son automáticamente autenticados en la herramienta integrada, de manera que sólo deben iniciar sesión una vez [51].

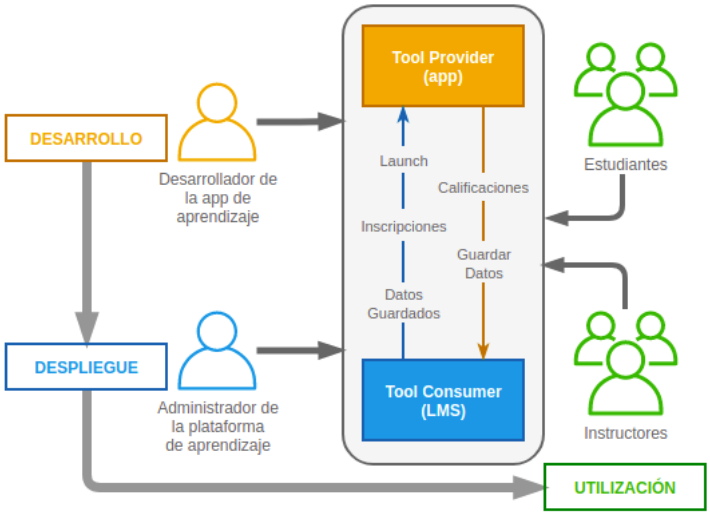


Fig. 1.3: Funcionamiento general de LTI[1]

- **SCORM** [52]

  SCORM (*Sharable Content Object Reference Model*) es un estándar de la iniciativa ADL (*Advanced Distributed Learning*) [53] que apunta a estandarizar los componentes de enseñanza basada en tecnología. Sus objetivos fundamentales son la fácil portabilidad de contenido de aprendizaje entre distintos LMS, así como la reutilización de OAs.

[1] Adaptado de https://resources.learning.powerschool.com/eajeanes/third-partytoolspublic/cms_page/view/19247415

SCORM denomina *Sharable Content Object* (SCO) a la unidad más pequeña que puede ser administrada por un LMS. Sólo un LMS puede lanzar un SCO, un SCO no debe lanzar otro SCO. Para poder ser reutilizados en diferentes situaciones de aprendizaje, los SCOs deben ser pequeñas unidades independientes del contexto de aprendizaje. SCORM especifica cómo debe prepararse y describirse el contenido (basado en XML), cómo se lanza el contenido y se comunica con el LMS (basado en JavaScript) y cómo puede navegarse entre las partes del curso (reglas y atributos en XML).

De esta forma, los SCOs que implementen SCORM son interoperables con todos los LMSs que implementen SCORM. Además, pueden intercambiar datos de usuario, metadatos, y datos de interacción (como la posición actual en el SCO, comentarios, anotaciones, duración, calificaciones). Un LMS que implemente SCORM puede manejar historiales de aprendizaje de los estudiantes y seguimiento de datos que luego pueden mostrarse a los tutores como retroalimentación.

Como contraparte, aspectos didácticos o de derechos de autor quedan fuera de consideración en este estándar. Tampoco permite repetir partes de de un SCO debido a un resultado negativo, ni permite a los tutores adaptar contenido específico por estudiante [54].

Fig. 1.4: Módulo de contenido empaquetado SCORM [55]

- **Breve comparación**

  Si bien tanto LTI como SCORM tienen el propósito de poder desarrollar aplicaciones compatibles con múltiples plataformas, hay dos diferencias bien marcadas que deben ser tenidas en cuenta:

  - SCORM requiere colocar el contenido dentro de un paquete descargable (por ejemplo, ZIP), el cual debe seguir la especificación dada para organizar el contenido que se encuentra dentro del paquete. Cualquier cambio en un SCO implica una nueva distribución y configuración del mismo. Esto da lugar a algunas dificultades al producir cursos con base en diferentes proveedores de SCO, el mantenimiento de SCOs y los cursos que los utilizan, y la calidad de los cursos basados en SCOs de diferentes proveedores [54].

- LTI permite que el contenido sea incluido directamente dentro del LMS, por ejemplo en un *iframe*, siempre que el contenido o la aplicación soporten el protocolo LTI para que otra aplicación web, como un LMS, pueda conectarla. Esto permite mantener todo el contenido centralizado y proveer acceso LTI al mismo, y que cualquier arreglo, cambio o mejora llegue a todos los lugares en que se usa sin tener que distribuir una nueva copia del OA [51].

Seguimiento de actividad y analíticas de aprendizaje

La Analítica del Aprendizaje o *Learning Analytics* (LA) es un área de investigación en la que confluyen expertos en tecnología educativa, inteligencia artificial, pedagogía, minería de datos o psicología, con gestores, administradores, políticos y empresas relacionadas con la educación. Por su naturaleza, LA es un campo multidisciplinar, en el que tecnólogos, pedagogos y otros agentes deben colaborar para capturar, procesar e interpretar la información generada desde el punto de vista educativo [56]. Representan un área de gran crecimiento dentro del área de investigación de las tecnologías para las mejoras en el aprendizaje o *Technology Enhanced Learning* (TEL) por cuanto sirven como estrategia para entender y optimizar los procesos de enseñanza y aprendizaje a través del análisis de datos generados a partir de las actividades realizadas por el estudiante en un entorno virtual de aprendizaje [57].

Son muchos y variados los retos a los que la comunidad de LA se enfrenta. Entre otros, a problemas técnicos propios del tratamiento de datos, como su procedencia y naturaleza heterogénea (múltiples fuentes y formatos, falta de estandarización), su calidad (la necesidad de discriminar los datos útiles de entre la gran cantidad de datos disponibles y trabajar hacia su estandarización), la diversidad de herramientas y técnicas para su procesamiento, y una visualización efectiva para que el usuario final pueda beneficiarse de la información generada [56] .

Para obtener algunas certezas sobre cómo los usuarios interactúan con una aplicación web (educativa o no), la información recolectada necesita tener cierto detalle y granularidad. Los LMS suelen rastrear la mayoría de las interacciones de usuario que pueden ser utilizadas para analíticas de aprendizaje, pero muchas de estas actividades tienen lugar por fuera de los LMS, como las actividades de programación [58].

Hay muchas aproximaciones destinadas a medir y colectar información respecto a las interacciones de usuario dentro de una página web y entre distintos sitios para llevar a cabo estas analíticas. En este trabajo hemos analizado dos especificaciones para seguimiento y análisis de datos relacionados con entornos y experiencias de aprendizaje en línea: **xAPI** (Figura 1.5) y **Caliper Analytics** (Figura 1.6).

- **xAPI** [59]

  Experience API (xAPI - *Experience Application Programming Interface*) es un proyecto de la ADL [53] que provee un formalismo independiente de una plataforma para recolectar eventos que ocurren en cualquier experiencia de aprendizaje (actividades de aprendizaje en tecnología mobile, simulaciones, aprendizaje offline, juegos, aprendizaje social, aprendizaje colaborativo, etc.). xAPI busca mejorar la interoperabilidad entre sistemas que recolectan e intercambian datos de aprendizaje de estudiantes, y superar las limitaciones de SCORM.

La especificación de xAPI describe el formato para representar actividades discretas de aprendizaje (como sentencias JSON) y los requerimientos para los *Learning Record Stores* (LRS), repositorios para estos registros de aprendizaje, que puedan recolectarlos e intercambiarlos. Un LRS puede ser parte de un LMS que soporte xAPI, o un sistema aparte. Por otro lado, xAPI permite a los distintos LRSs hablar entre sí, de manera de poder seguir la actividad de un o una estudiante a través de varios sistemas.

Las sentencias xAPI contienen 3 elementos principales: metadata (id, marca de tiempo, etc), información descriptiva (actor, verbo y objeto) y datos complementarios (contexto), y típicamente tienen la forma "actor" + "verbo" + "objeto" o "Yo hice esto". El "actor" es información única que describe a un sujeto específico, como un estudiante o grupo de estudiantes; el "verbo" clasifica el tipo de actividad en la que participó el actor con posibles enlaces a una descripción del evento legible por personas; el "objeto" enlazará a un artefacto que es típicamente un subproducto de o está relacionado con la actividad [60].

La especificación de xAPI sólo provee de una estructura sintáctica para componer las sentencias. No tiene un vocabulario central definido, sino que es la comunidad la que debe definir y compartir la estructura de las sentencias xAPI y el vocabulario como recetas (o definiciones semánticas) específicas para un dominio. Lo que luego se haga con los datos recolectados depende de quien los utilice, por lo que se debe decidir la mejor forma de obtener información significativa. El correcto modelado de la información contextual dentro de las sentencias xAPI es crítico a efectos de las analíticas que se busquen [61].

Fig. 1.5: Flujo de datos en xAPI [62]

- **Caliper Analytics** [63]

  Caliper Analytics es una especificación de IMS Global que extiende LTI y otros estándares relacionados, y busca abordar problemas relativos a la recopilación de información de las actividades de aprendizaje desde distintos sistemas. Para ello provee un formato común y extensible para describir, colectar e intercambiar dicha información a través de múltiples entornos de aprendizaje.

  Este estándar utiliza los modelos de información de varias especificaciones IMS (*Learning Tools Interoperability* o LTI, *Learning Information Services* o LIS, y *Question and Test Interoperability* o QTI) para soportar y avanzar en la medición del aprendizaje a través de la generación de analíticas. La Figura 1.6 muestra la arquitectura

de IMS Caliper. En la misma se destacan los perfiles de métricas de aprendizaje IMS o *Learning Metric Profiles*, donde se tienen en cuenta las acciones/actividades realizadas por los estudiantes y los elementos relevantes de su contexto; y el sensor de aprendizaje o *Learning Sensor API*, que representa las acciones que se ejecutan en las actividades que se plantean dentro de un contexto de aprendizaje [57].

La idea de realizar este seguimiento es proveer información de la actividad de los estudiantes a los docentes, para permitirles analizar qué comportamientos y contenidos dan lugar a un mejor proceso de enseñanza-aprendizaje, personalizar los contenidos en tiempo real en función de los patrones observados, y armar sistemas de advertencia temprana y medidas predictivas para resolver rápidamente las dificultades con las que se puede encontrar un estudiante y que podrían obstaculizar su aprendizaje.

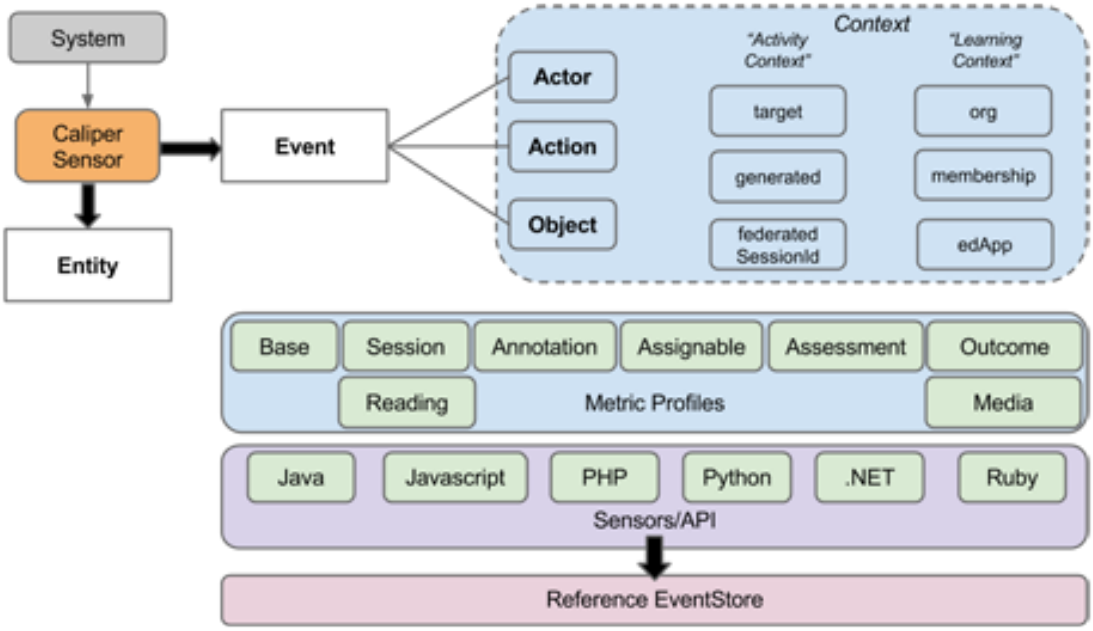


Fig. 1.6: Funcionamiento IMS Caliper [64]

1.3. Enfoque y alcance de este trabajo

Dado el concepto de Cuaderno Activo definido inicialmente, decidimos trabajar con los llamados **cuadernos computacionales**, ya que son las herramientas que más se acercan al funcionamiento ideado, favoreciendo la interactividad y permitiendo intercalar prosa con código, ecuaciones, figuras y multimedia. De esta manera, se pueden crear tareas que incluyan instrucciones junto con código, donde los estudiantes pueden proveer soluciones a los ejercicios, escribir código para computar y visualizar un resultado particular. Además,

dada la interactividad de estos cuadernos, los estudiantes pueden iterar en la resolución de un problema sin necesidad de ir y venir entre una línea de comandos y un editor de texto, y pueden rápidamente ver los resultados junto al código que los produce.

Además, estos cuadernos permiten considerar diferentes niveles de complejidad, dependiendo de las necesidades del docente y la motivación del estudiante [65]:

- Usar explicaciones soportadas por gráficos obtenidos de simulaciones.
- Permitir a los estudiantes jugar con las simulaciones como si fueran cajas negras, es decir, modificando diferentes parámetros y obteniendo resultados sin entrar en los métodos computacionales.
- Permitir a los estudiantes modificar código para incrementar las simulaciones dadas, permitiéndoles adquirir nuevas habilidades computacionales.

En particular, decidimos utilizar los cuadernos de Jupyter. Esta elección se basa en su código abierto y flexibilidad, además de la gran popularidad que ha alcanzado en el ámbito científico y, en particular, para propósitos educativos, con la consiguiente existencia de múltiples recursos en línea dedicados a Jupyter Notebook. Esta popularidad se debe a la posibilidad de aislar al documento en sí de su centro computacional, lo que permite olvidarse de la instalación local. El instructor puede usar un servidor externo (o incluso su portátil) para proporcionar a los estudiantes las lecturas, diapositivas, o tareas interactivas. Esta característica permite remover el obstáculo más común que surge al intentar presentar un nuevo documento o programa: que el estudiante tenga que instalarlo localmente en su computadora [65].

Cabe notar que en la medida en que un cuaderno crece en tamaño y complejidad, se vuelven más complicados de trabajar desde una perspectiva de ingeniería de software (testeo, documentación, reuso, modularidad, etc.). Sin embargo, el objetivo inicial de los Cuadernos Activos es el de *introducir y ejercitar conceptos y problemas de programación desde una perspectiva más exploratoria*, donde el resultado de cada cómputo informa el siguiente paso y guía la formación de intuiciones sobre el problema en mano [5].

Por su parte, decidimos no trabajar con sistemas de gestión de aprendizaje, ya que se orientan más a la administración de contenidos. Si bien esto no es un tema menos importante, se aleja de la idea inicial de Cuaderno Activo como único documento holístico y con impacto significativo en las metodologías de enseñanza. Es de notar, a su vez, que mediante los estándares de interoperabilidad enunciados, los Cuadernos Activos desarrollados podrían ser administrados y utilizados a través de estas plataformas.

2. ARQUITECTURA BASE

A continuación se presenta brevemente el funcionamiento de Jupyter Notebook y JupyterHub, para luego introducir una propuesta para la presentación de contenido dinámico en el Cuaderno Activo.

2.1. Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código vivo, ecuaciones, visualizaciones y texto explicativo, entre otras opciones de contenido interactivo [8].

Un cuaderno o *notebook* consiste en una secuencia de celdas, que son campos para introducir texto. El comportamiento de una celda está determinada por su tipo: una celda de código permite editar y escribir código que luego será ejecutado, una celda de Markdown permite introducir texto enriquecido mediante Markdown, y una celda raw permite escribir output directamente sin tener que ser evaluado (notar que tanto el código como Markdown deben ser ejecutados) [66].

La aplicación web de Jupyter Notebook se basa en una arquitectura cliente-servidor. En la Figura 2.1 se pueden distinguir tres componentes principales [67]:

- **Aplicación Web:** ejecuta en el navegador del cliente y permite editar código, ejecutarlo desde el mismo navegador asociando el resultado al código que lo generó, mostrar los resultados en representaciones multimedia, editar texto e introducir notación matemática con LaTeX, etc.

- **Notebook Server:** comunica la aplicación web con el *kernel* usando un protocolo de red basado en mensajes JSON sobre ZeroMQ [68] y WebSockets, así como Tornado [69] para servir consultas HTTP. Es responsable de guardar y cargar los notebooks. Un mismo Notebook Server puede soportar notebooks con distintos lenguajes, conectando cada notebook con el *kernel* correspondiente.

- **Kernel:** es un proceso separado responsable de correr código de usuario en un lenguaje de programación determinado. El *kernel* no conoce el documento del notebook, sólo toma las celdas de código enviadas a ejecutar (mediante el *Notebook Server*) cuando el usuario lo indica, tal como se muestra en la Figura 2.2.

También tenemos los documentos de los notebooks, que contienen las entradas y salidas (*inputs* y *outputs*) de una sesión interactiva, además del texto adicional, fórmulas, representaciones gráficas, etc. Estos documentos son internamente archivos JSON y son guardados con la extensión .ipynb [66].

2.1.1. Extensiones frontend y backend

Tanto el cliente como el servidor de Jupyter Notebook son completamente personalizables. Su comportamiento puede ser extendido creando, respectivamente:

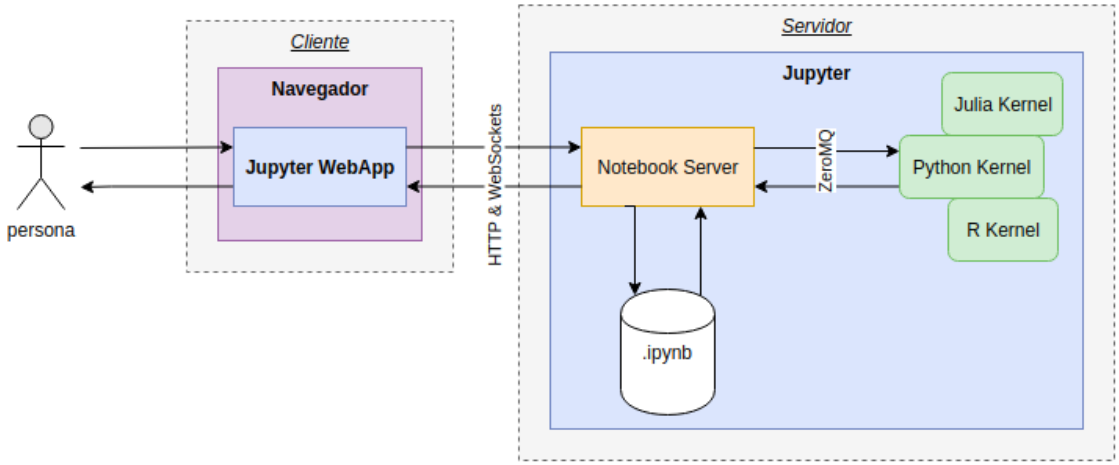


Fig. 2.1: Arquitectura Jupyter Notebook

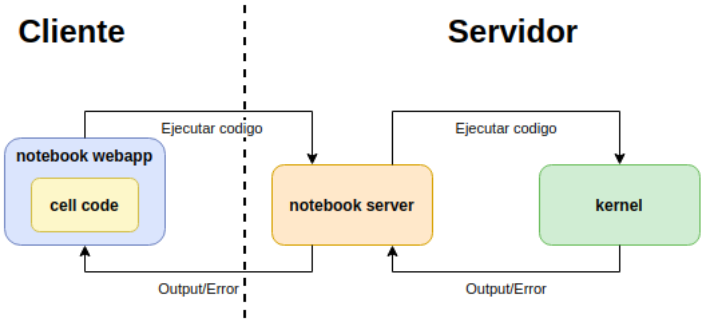


Fig. 2.2: Cómo funciona Jupyter Notebook

- **nbextension:** es un módulo JavaScript que corre en la aplicación web del Notebook, y que tiene acceso al DOM (Document Object Model) completo y a la API JavaScript del frontend de Jupyter, con lo cual puede modificar la experiencia de usuario, añadir botones, crear nuevos atajos de teclado, etc. Son extensiones exclusivas del front end.
- **server extension:** es un módulo Python importable que carga con el Notebook Server y que permite personalizar y extender el comportamiento del mismo, por ejemplo, añadiendo nuevos endpoints HTTP.

2.1.2. JupyterHub

JupyterHub es una herramienta ofrecida por el proyecto Jupyter, consisitendo en un servidor multiusuario, donde cada usuario puede acceder a su propio entorno y manejar sus propios Jupyter Notebooks. Más específicamente, presenta una página de identificación cuando se accede a través del navegador, administra IDs de usuario y, una vez que se permite el acceso, exhibe un servidor de notebooks específico para ese usuario. Allí puede subir desde su computadora cualquier tipo de archivo, descargar, crear y trabajar con Jupyter Notebooks [65]. Así, JupyterHub provee una base para ofrecer a los usuarios un entorno unificado para acceder a Jupyter Notebooks desde sus propios navegadores, sin la necesidad de una instalación local.

Tal como se muestra en la Figura 2.3, comprende cuatro subsistemas principales [70]:

- **Hub:** es el corazón de JupyterHub, se trata de un proceso multi-usuario de Tornado que maneja las cuentas de usuario, autenticación y coordina los Single-User Notebook Servers usando un *Spawner*.
- **Configurable HTTP Proxy:** la cara pública de JupyterHub que usa un proxy dinámico para routear consulas HTTP al Hub y a los Single-User Notebook Servers.
- **Single-User Notebook Server:** se inicia un servidor de Jupyter Notebook dedicado para cada usuario del sistema, cuando el mismo se autentica. El componente que monitorea estos servidores se llama *Spawner*.
- **Authentication Class:** maneja cómo los usuarios acceden al sistema, ya sea utilizando usuarios existentes en el sistema, cuentas de GitHub o cualquier sistema utilizado en la organización.

La manera en que el *Spawner* controla cómo se inicia cada notebook server individual para cada usuario también es configurable. Por defecto, se inicia un notebook server en la misma máquina donde corre el sistema, bajo su nombre de usuario. Otras alternativas permiten iniciar cada notebook server en contenedores separados, por ejemplo utilizando Docker [71].

Para usar JupyterHub se necesita un servidor Unix accesible en la red al equipo de personas que utilizará los notebooks. Puede ser un servidor dentro de una red interna o puede correr públicamente en Internet.

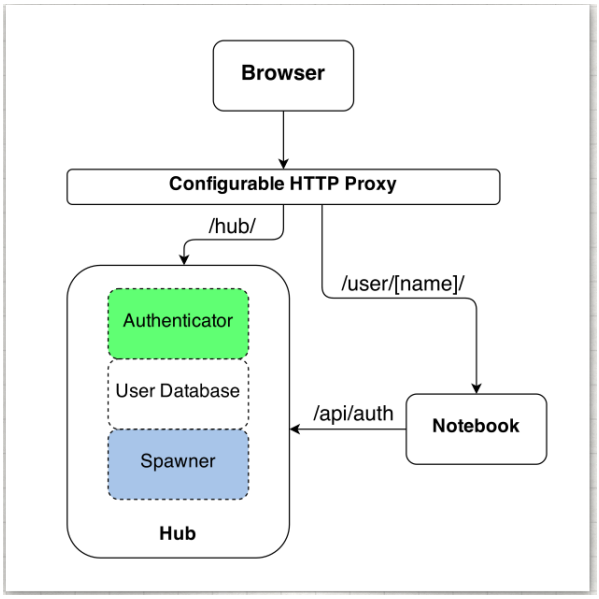


Fig. 2.3: Arquitectura JupyterHub [70]

2.2. Sistema de corrección e indicador de próximo ejercicio

Como ya adelantamos, en este trabajo buscamos utilizar Cuadernos Activos (CA) para la enseñanza. Buscamos que los mismos CA sean dinámicos, de manera que su estructura

y contenidos progresen en el tiempo acorde a decisiones tomadas por el docente respecto a cada estudiante. Para ello, la herramienta deberá proveer información y sugerencias sobre el trabajo de un estudiante.

La idea es que los bloques o celdas que constituyen un CA estén relacionados, y que el estudiante pueda utilizar el CA de manera secuencial. Así, al suceder un evento en particular (como por ejemplo, que se complete un ejercicio) los sucesivos bloques se irían "activando" o "habilitando" en el orden previsto por el docente. A su vez, la habilitación o no de bloques subsecuentes podría depender de condiciones tales como "haber contestado satisfactoriamente un porcentaje de los ejercicios ofrecidos", etc.

Frente a esta intención, se decide implementar un prototipo de corrector automático que pudiera recibir la respuesta de un estudiante a un ejercicio y decidir cuál es el siguiente contenido (texto, advertencia, ejercicio, etc.) a mostrar en el CA. Este corrector sería un *backend* corriendo en un servidor (potencialmente distinto al servidor en el que ejecuta el Jupyter Server), por lo cual se debe establecer la comunicación entre el CA y dicho *backend* corrector.

2.2.1. Implementación

Implementamos[1] un servidor en Python utilizando Tornado, que es la tecnología utilizada por Jupyter, estableciendo que las consultas al mismo se realicen mediante una API REST con datos en formato JSON, de la siguiente manera:

1. El corrector recibe una consulta POST con el nombre de usuario que solicita la corrección, un identificador del ejercicio, y la respuesta dada por el usuario.
2. Se almacena esta información en una base de datos, de forma de poder hacer un seguimiento y consultar estadísticas.
3. Se evalúa la respuesta provista.
4. El corrector responde a la consulta con la información necesaria para mostrar el siguiente bloque en el CA.

2.3. Propuesta de diseño: aprovechando las extensiones de Jupyter Notebook

Dado que se trata de agregar un nuevo comportamiento a Jupyter optamos por implementar extensiones, ya que es la forma en la que Jupyter permite extender su funcionalidad.

Por un lado, para el *frontend*, desarrollamos una *nbextension* con las siguientes características:

- Se provee un nuevo botón en el cuaderno, que inicia la corrección.
- Cuando el estudiante resuelve el ejercicio, selecciona la celda con la respuesta y hace click en el botón.
- Una vez presionado el botón, la *nbextension* recupera la respuesta provista en la celda y la envía para ser corregida.

[1] Todas las implementaciones de este trabajo se encuentran en https://github.com/giselabelen/cuadernosactivos

- La *nbextension* se comunica con el sistema corrector mediante un protocolo que se describe más adelante.
- Cuando la *nbextension* recibe la corrección, genera una nueva celda en el cuaderno con el contenido a mostrar a continuación.

Para la comunicación entre la *nbextension* y el sistema corrector, planteamos dos enfoques:

- **Opción A:** la *nbextension* se comunicaría directamente con el sistema corrector mediante la API REST provista para ello (ver Figura 2.4).
- **Opción B:** la *nbextension* enviaría la información pertinente al Notebook Server, y éste haría la conexión con el sistema corrector. A su vez, el sistema corrector enviaría la respuesta al Notebook Server, para que éste la transmita a la *nbextension*. Esta opción requiere implementar una *Jupyter server extension* para establecer un nuevo HTTP endpoint destinado a ser el intermediario, y los controladores (*handlers*) REST necesarios (ver Figura 2.5).

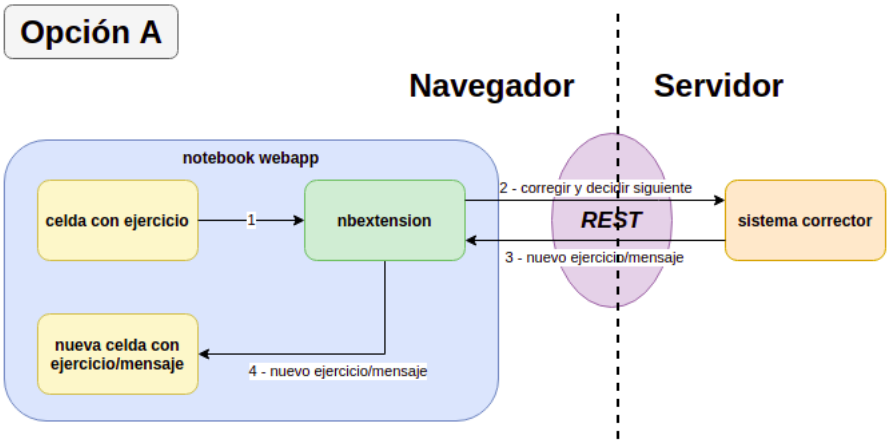


Fig. 2.4: Opción A: *nbextension* acoplada al sistema corrector

Un primer análisis de estas dos opciones sugiere que la opción A puede ser más práctica, dado que prescinde de un intermediario. Sin embargo, se genera acoplamiento entre la *nbextension* y el sistema corrector, ya que la primera debe conocer la ubicación del segundo para poder realizar la consulta. Esto puede ser un problema si pensamos en gran escala: el desarrollo y mantenimiento del *backend* afectaría directamente el desarrollo y mantenimiento de la *nbextension*.

En cuanto a la opción B, en primera instancia se muestra más complicada al tener que implementar una *server extension* y hacer la mediación entre la *nbextension* y el sistema corrector. Pero a su vez se muestra más prolija y robusta al desacoplar *nbextension* y *backend*: la *nbextension* siempre hablaría con el Notebook Server, cuya ubicación puede ser conocida gracias a funcionalidad provista en la API de *frontend* de Jupyter, y es la *server extension* la que quedaría acoplada al corrector, lo cual es más apropiado siendo que ambos viven en el *backend* del sistema.

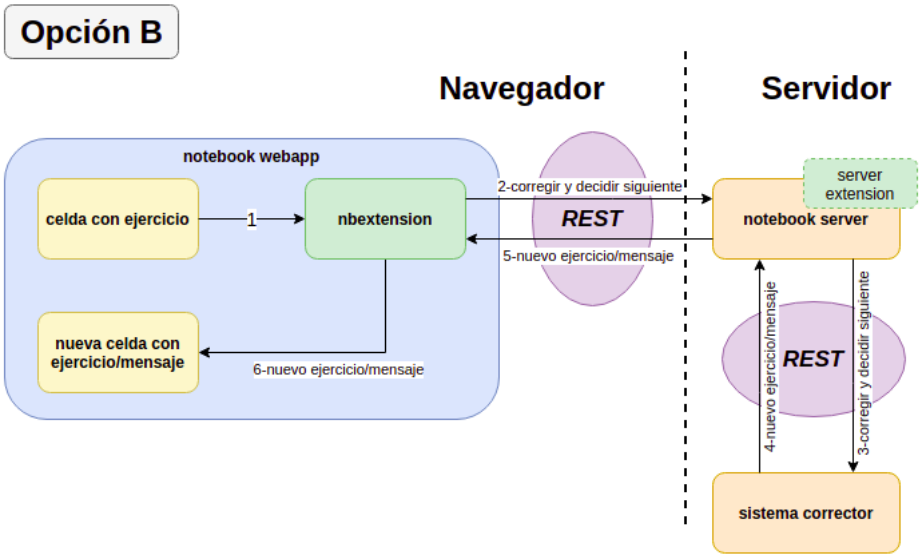


Fig. 2.5: Opción B: *server extension* como intermediario entre *nbextension* y sistema corrector

Implementamos ambas opciones, de manera de poder comparar una con la otra, y las observaciones avalan las suposiciones presentadas. Además, nos encontramos con un factor que no se había tomado en cuenta en el primer análisis: la identificación del usuario.

A la hora de buscar identificar el usuario que solicita la corrección, encontramos que el Notebook Server provee dentro de sus controladores una manera de consultar el usuario actual. Es decir, es una consulta que puede hacer una *server extension*. Por lo tanto, sería útil para la opción B, mas no para la opción A que no utiliza al Notebook Server y una *server extension* para intermediar. Utilizando esta consulta en la implementación de la opción B, encontramos que sólo provee información relevante cuando el Notebook Server es lanzado dentro de JupyterHub, informando el nombre de usuario, tipo de usuario, su última actividad, etc.; mientras que iniciando el Notebook Server de manera individual, la consulta es respondida con un valor que varía entre sesión y sesión del mismo usuario, así que no permite identificarlo.

Ciertamente, lanzando el Notebook Server individualmente no tiene sentido intentar identificar distintos usuarios, y dado que la administración de usuarios y Notebook Servers a través de JupyterHub supone una ventaja alineada a la intención de este trabajo, la manera de obtener el usuario actual no representa una complicación. Pero sí nos permite concluir que la opción B es la más adecuada para seguir avanzando.

3. PRUEBAS DE CONCEPTO SIMPLES: SECUENCIAS DE EJERCICIOS

En este capítulo se presentan las primeras pruebas de concepto abordadas con el objetivo de explorar el funcionamiento del sistema corrector mediante el uso de *nbextensions* y *server extensions* de Jupyter.

Llevamos a cabo dos pruebas de concepto orientadas a obtener una corrección y sugerencia de siguiente ejercicio. El primero y más simple sólo toma el valor ingresado en una celda e indica si es una respuesta correcta o no, mientras que el segundo evalúa el desempeño en una serie de ejercicios simples e indica el siguiente paso a seguir.

3.1. Corrección sin seguimiento de actividad

La persona que está utilizando el CA escribe un valor en una celda, hace click en el nuevo botón *Enviar la respuesta*, presente en la barra de herramientas, y aparece una nueva celda inmediatamente posterior a la utilizada, indicando si la respuesta es correcta o no.

El objetivo de esta prueba es corroborar qué componentes entran en juego para poder hacer una "corrección" y de qué manera interactúan.

Se describen a continuación los casos de prueba abordados.

1. Respuesta correcta:
   a) La persona que utiliza el CA ingresa el valor 1 en una celda (de cualquier tipo).
   b) Luego presiona el botón *Enviar la respuesta* para enviar la respuesta y obtener la corrección (Figura 3.1).
   c) Aparece una nueva celda inmediatamente debajo de la celda utilizada en (*a*) con el mensaje "`Respondiste bien`" (Figura 3.2).
2. Respuesta incorrecta:
   a) La persona que utiliza el CA ingresa un valor distinto de 1 en una celda (de cualquier tipo).
   b) Luego presiona el botón *Enviar la respuesta* para enviar la respuesta y obtener la corrección (Figura 3.3).
   c) Aparece una nueva celda inmediatamente debajo de la celda utilizada en (*a*) con el mensaje "`Respondiste mal`" (Figura 3.4).

La Figura 3.5 muestra la dinámica de esta prueba de concepto, para la cual desarrollamos:

- Una ***nbextension*** que habilita el botón *Enviar la respuesta* en el frontend, recupera el valor ingresado en la celda por quien utiliza el CA y envía esta información a la *server extension* correspondiente mediante una consulta POST REST. Luego recibe la respuesta de la *server extension* indicando el resultado de la "evaluación", y añade la nueva celda con el mensaje adecuado.

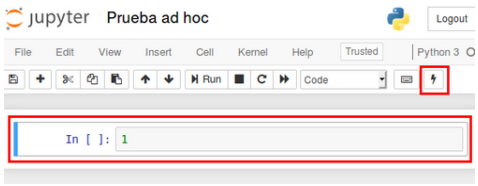


Fig. 3.1: Caso de prueba 1: se ingresa el valor correcto y se solicita la corrección

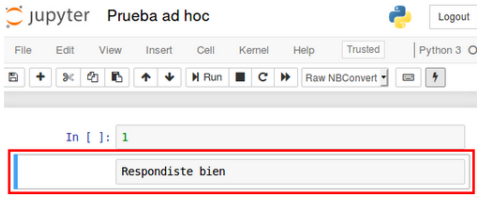


Fig. 3.2: Caso de prueba 1: se obtiene el mensaje de respuesta correcta

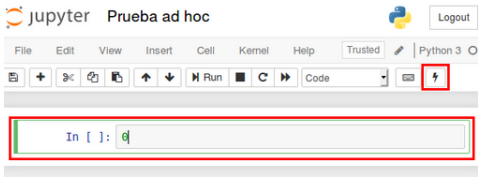


Fig. 3.3: Caso de prueba 2: se ingresa un valor incorrecto y se solicita la corrección

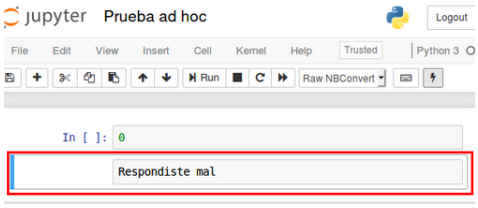


Fig. 3.4: Caso de prueba 2: se obtiene el mensaje de respuesta incorrecta

- Una ***server extension*** que recibe la información de la celda recuperada por la *nbextension*, consulta además el nombre de usuario de quien está utilizando el CA, y envía ambos datos (nombre y respuesta) mediante una consulta POST REST a un servidor encargado de realizar la corrección propiamente dicha. Luego recibe la respuesta de este servidor y la reenvía a la *nbextension*.
- Un **servidor** capaz de tomar decisiones, que recibe la información correspondiente desde la *server extension* y evalúa si la respuesta es o no correcta.

3.2. Secuencias de ejercicios con seguimiento de actividad

Quien utiliza el CA solicita un ejercicio presionando el nuevo botón *Siguiente ejercicio* presente en la barra de herramientas. El ejercicio aparece en una celda nueva, el usuario lo resuelve en la celda contigua y, presionando el mismo botón que antes, lo envía a corregir a la vez que solicita el siguiente ejercicio. La sucesión de ejercicios variará dependiendo del desempeño que vaya teniendo quien utiliza el CA.

El objetivo de esta prueba es presentar un flujo dinámico en el que una persona puede recorrer muchos caminos posibles hasta completar lo que se considere como "aprobable", además de permitir un seguimiento simple pero completo del recorrido de cada persona.

Para lograr esto, ideamos 3 guías de 5 ejercicios cada una, representando distintos niveles de dificultad:

- Guía 1 (fácil): sumas de números naturales de 1 cifra.
- Guía 2 (medio): sumas de números naturales de 2 cifras.
- Guía 3 (difícil): sumas de números naturales de 3 cifras.

Las guías se presentan a quien utiliza el CA en orden creciente de dificultad (o sea: guía1 $<$ guía2 $<$ guía3). Cada vez que quien utiliza el CA complete los ejercicios de una guía,

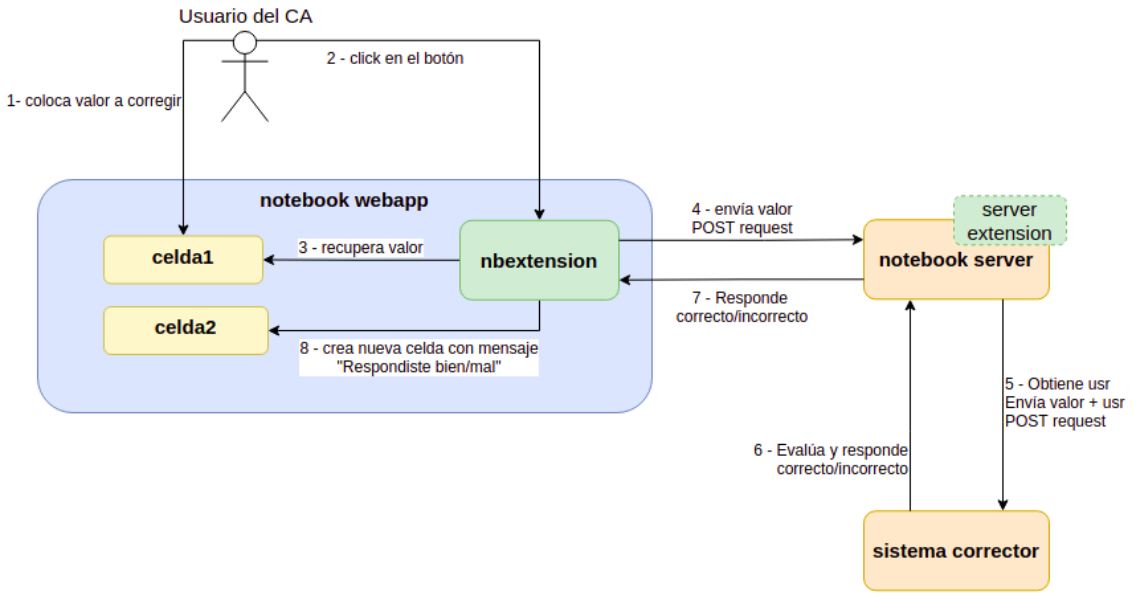


Fig. 3.5: Prueba de concepto simple sin seguimiento

el sistema responsable de corregir y decidir qué sigue evalúa la cantidad de aciertos en las respuestas. Si hay al menos 3 aciertos, se continúa con los ejercicios de la siguiente guía (si es la guía 3, notifica: "`bien ahí, aprendiste todo`"). En caso contrario, se presentan los ejercicios de la guía anterior (si es la guía 1, notifica: "`vamos a repasar`"). Esta dinámica se muestra en la Figura 3.6.

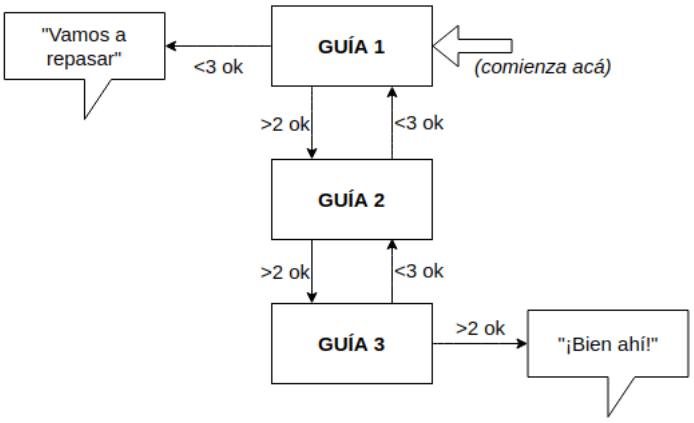


Fig. 3.6: Prueba de concepto simple con seguimiento de respuestas

La Figura 3.8 muestra la dinámica de esta prueba de concepto, para la cual creamos:

- Una ***nbextension*** que habilita el botón *Siguiente ejercicio* en el *frontend*, recupera la respuesta ingresada en la celda por quien utiliza el CA y el número de guía y ejercicio resuelto (estos dos valores se mantienen en los metadatos de la celda en la que se resuelve el ejercicio) y envía esta información a la *server extension* correspondiente mediante una consulta POST REST. Luego recibe la respuesta de la *server extension*

indicando el número de guía y ejercicio a resolver a continuación y el ejercicio en sí (o mensaje especial), y añade una nueva celda con el ejercicio/mensaje y con el número de guía y ejercicio correspondiente en sus metadatos.

- Una ***server extension*** que recibe la información enviada por la *nbextension*, consulta además el nombre de usuario de quien está utilizando el CA, y envía todos estos datos mediante una consulta POST REST a un servidor encargado de realizar la corrección y elección de actividad siguiente. Luego recibe la respuesta de este servidor y la reenvía a la *nbextension*.
- Una **base de datos SQL** (Figura 3.7) con:
  - una tabla para los **ejercicios** (identificados con número de guía y número de ejercicio) con su correspondiente respuesta correcta y un indicador de fin de guía o del siguiente ejercicio de la misma guía;
  - una tabla para el **seguimiento del desempeño** de quien utiliza el CA, con una entrada para cada solicitud de ejercicio, indicando usuario, timestamp, identificadores de la actividad resuelta (guía-ejercicio), respuesta propuesta por el usuario y un indicador de cuántas veces esta persona ha resuelto este ejercicio (recordar que una persona puede tener que repetir más de una vez una guía).
- Un **servidor** que recibe la información correspondiente desde la *server extension* y con ello puede evaluar y proponer la siguiente actividad. Esto implica:
  - almacenar la solución dada por quien utiliza el CA en la base de datos;
  - en caso de que sea el último ejercicio de una guía, evaluar la cantidad de respuestas correctas;
  - buscar en la base de datos el siguiente ejercicio a enviar, dependiendo de la guía y ejercicio actual y del desempeño en la guía (si es que la finalizó);
  - responder a la *server extension* con los datos del ejercicio siguiente o mensaje a presentar.

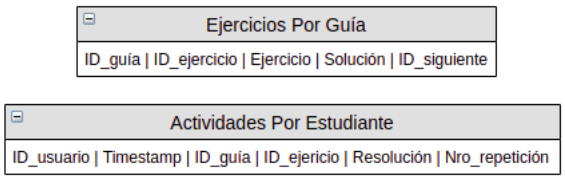


Fig. 3.7: Tablas de la base de datos de la prueba de concepto simple con seguimiento de respuestas

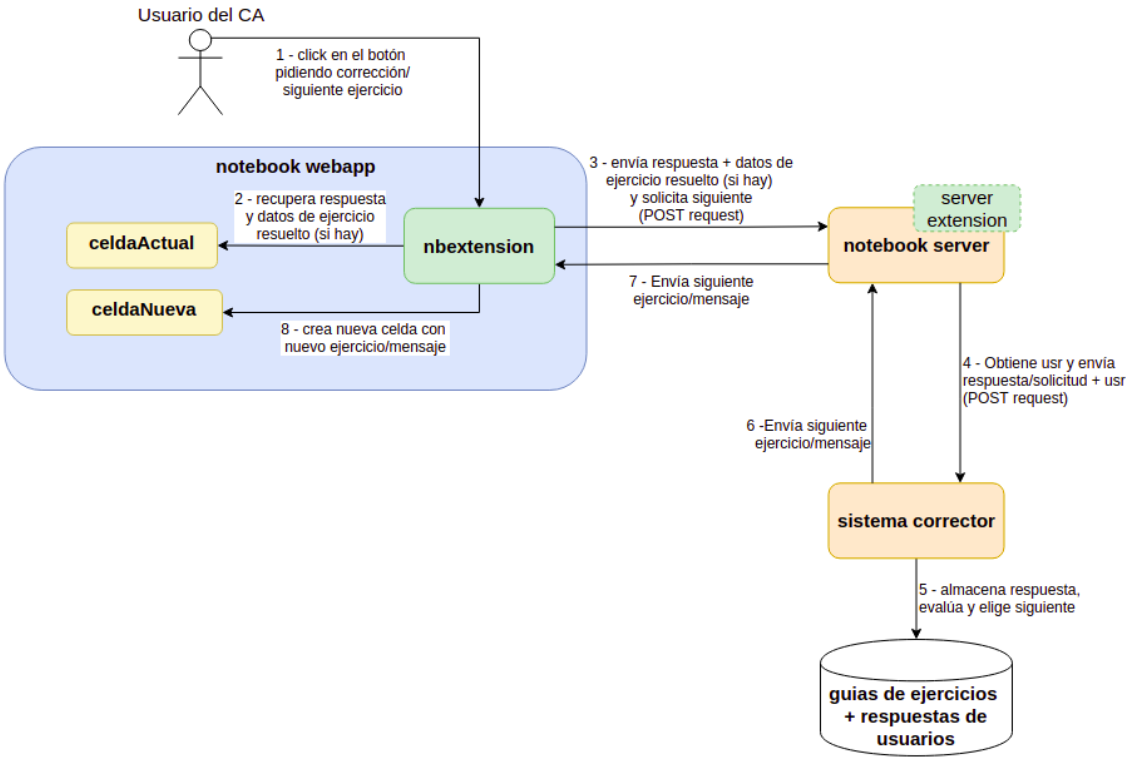


Fig. 3.8: Prueba de concepto simple con seguimiento de respuestas

4. PRUEBAS DE CONCEPTO AVANZADAS: INTEGRACIÓN CON APLICACIONES

Hasta aquí, analizamos y aprovechamos la arquitectura de Jupyter para implementar un prototipo de sistema que corrige e indica el siguiente contenido del CA de forma dinámica, en función del desempeño del usuario hasta ese momento.

Partiendo de esta base, la siguiente pregunta tiene que ver con cómo involucrar una herramienta externa a Jupyter como contenido del CA, y cómo hacer un seguimiento del desempeño del usuario en este caso. Basándonos en la arquitectura planteada anteriormente con el sistema corrector, un primer enfoque para la comunicación con una herramienta externa podría ser la siguiente:

1. El usuario del cuaderno pide un nuevo ejercicio haciendo click en el botón de una *nbextension*.
2. La *nbextension* envía una consulta al *notebook server* preguntando por el contenido a mostrar.
3. El *notebook server*, mediante una *server extension*, obtiene el usuario que hizo el pedido y reenvía la consulta al sistema que toma decisiones.
4. El sistema que toma decisiones consulta el historial del usuario y los criterios pedagógicos que deban considerarse, y decide el contenido a mostrar.
5. El sistema que toma decisiones responde la consulta del *notebook server* con los datos para iniciar el nuevo contenido.
6. El *notebook server* responde la consulta de la *nbextension* con los datos para iniciar el nuevo contenido.
7. La *nbextension* crea una nueva celda en el cuaderno con el contenido correspondiente de la herramienta externa (potencialmente dentro de un *iframe*).
8. El usuario del cuaderno interactúa con la herramienta externa (a través del *iframe*), y ésta va enviando los datos necesarios de seguimiento al repositorio que tiene el historial del usuario.
9. Cuando el usuario quiere solicitar el siguiente contenido, vuelve al paso 1.

¿Qué implicancias tiene esta secuencia? La resolución de un ejercicio y la solicitud de otro son dos pasos separados, y por lo tanto, dos clicks separados. Ciertamente, la intención inicial era que un mismo click iniciara la corrección y la decisión subsiguiente. Evaluando esta situación, observamos que:

- una herramienta externa podría contar con un mecanismo propio de asistencia durante la resolución y posterior corrección de un ejercicio determinado; y
- un usuario podría intentar resolver varias veces un mismo ejercicio (en el mismo *iframe*) antes de querer saber cuál es el siguiente;

por lo que sería suficiente con obtener los datos de seguimiento de la actividad del usuario y almacenarlo en su historial (incluyendo quizás cantidad de intentos y retroalimentación de la herramienta externa), de manera de permitir tomar la decisión del siguiente ejercicio cuando esto sea solicitado. En el caso en que la herramienta externa no contase con estas facilidades, bastaría con que pueda enviar la resolución del ejercicio en algún formato para que luego nuestro sistema corrector se ocupe de la corrección en sí misma, aunque esto también nos da la pauta de que, potencialmente, se deba desarrollar un sistema corrector específico (o un sistema general con métodos específicos) para cada herramienta a incluir.

Ahora bien, para poder implementar esta secuencia de pasos aún queda pendiente definir cómo se integraría la herramienta externa en la celda del CA, qué datos de seguimiento se querrían recabar y en qué formato, y cómo se le daría la adaptabilidad a nuestro sistema corrector. En estas preguntas se basarán las siguientes secciones del presente capítulo.

4.1. Motivación: Integración con *Pilas Bloques*

Para llevar a cabo estas pruebas de concepto, buscamos integrar al CA la herramienta **Pilas Bloques** [72] (PB), una aplicación desarrollada por la iniciativa Program.AR [73] de la Fundación Sadosky [74]. La herramienta está pensada como ayuda al docente y al estudiante en el proceso de aprendizaje de la programación en un entorno escolar. Sus desafíos, con diversos niveles de dificultad, buscan acercar a los estudiantes al mundo de la programación por medio de bloques. En la Figura 4.1 se muestra uno de los desafíos que presenta la herramienta.

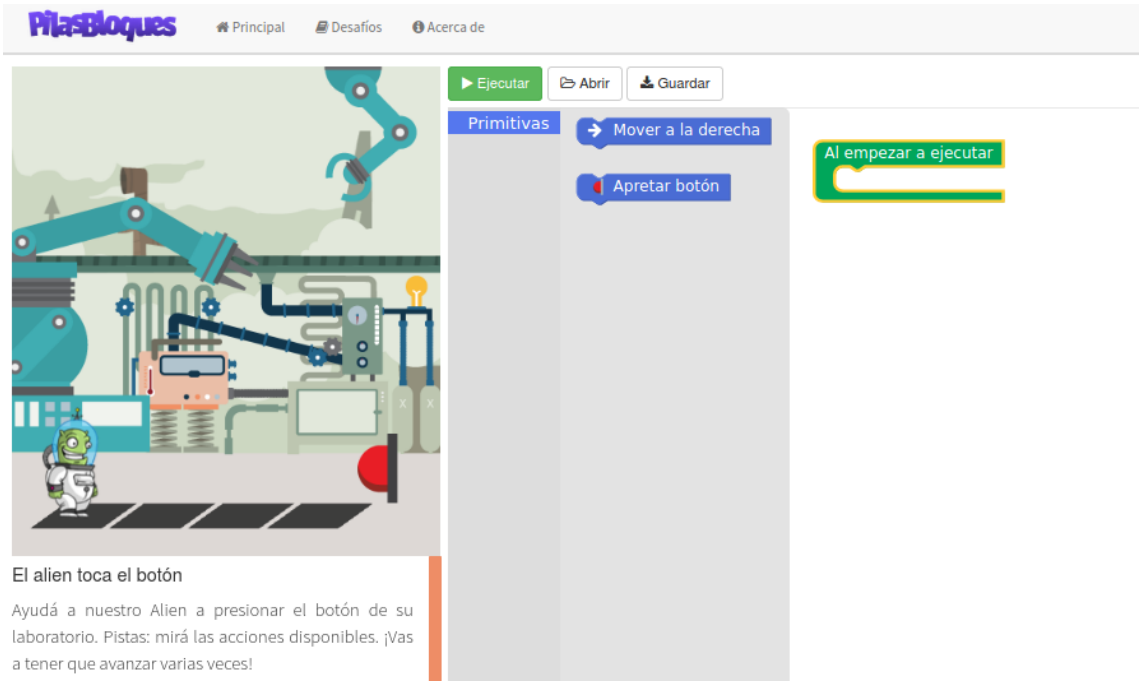


Fig. 4.1: Desafío de Pilas Bloques

4.2. Elección de LTI

A la hora de elegir con qué especificación trabajar para garantizar la interoperabilidad de herramientas y el seguimiento de las actividades, conversamos con los desarrolladores de PB ya que, al momento de desarrollar esta tesis, la herramienta no cuenta con la implementación de ningún protocolo para integrarse a otros entornos. En dicha conversación, conocemos su intención de implementar LTI para poder integrar PB a LMSs como Moodle y que, con esa intención, cuentan con un incipiente desarrollo de un *backend* en Python Flask [75] para recibir y procesar consultas HTTP.

Por otro lado, consideramos que trabajar con LTI sería más ventajoso que elegir SCORM, debido a que este último se muestra más engorroso de implementar y mantener. Además, cada vez más herramientas *online* deciden trabajar con LTI [76], lo cual significa más variedad de aplicaciones con las que se podría trabajar a futuro en el CA.

De todos modos, esta decisión (o cualquier otra) implica dedicar una etapa al desarrollo de dicho protocolo, por lo cual postergamos la decisión relativa al seguimiento y analíticas (caso xAPI vs. Caliper). Por el mismo motivo, la adaptabilidad del sistema corrector ante distintos caminos pedagógicos se relega a una próxima etapa, siendo que, además, constituye un área de estudio en sí misma.

4.3. LTI 1.0 (Basic LTI)

4.3.1. Idea

Para comenzar con la etapa de implementación de LTI, apuntamos primeramente a la versión 1.0 [77], que establece el flujo de trabajo para realizar lo que llaman *Basic LTI Launch*, es decir, lo mínimo para lanzar la herramienta externa o *Tool Provider* (en nuestro caso, Pilas Bloques) dentro del entorno principal o *Tool Consumer* (en nuestro caso, un cuaderno de Jupyter). La idea principal es autorizar el acceso de usuarios desde el *Tool Consumer* (TC) al *Tool Provider* (TP), y para ello en general la responsabilidad de autenticación recae en el TC.

Basic LTI Launch

El procedimiento para el *LTI Launch* es el siguiente:

1. Un usuario hace click sobre un link LTI.
2. El TC prepara los parámetros para el lanzar la herramienta:
   a) Típicamente se devuelve una página HTML al navegador con un formulario con los parámetros ocultos.
   b) Los datos son firmados digitalmente con OAuth1.0a [78].
3. Se hace una consulta POST automática al TP mediante JavaScript [79], de manera que sea el navegador quien envíe la consulta y no el servidor de TC.
4. El TP responde con una presentación de la herramienta en el navegador.

Para que este *LTI launch* pueda tener lugar, hay ciertos aspectos que tanto el TC como el TP deben poder cubrir. Desde el punto de vista de nuestra prueba de concepto

con Pilas Bloques como TP y un cuaderno de Jupyter como TC, los requerimientos son los siguientes:

- Pilas Bloques debe:
  - **Poder verse dentro de un *iframe***. PB lo permite, pero es de notar que ciertas aplicaciones web no admiten esta funcionalidad por temas de seguridad.
  - **Poder aceptar y procesar consultas POST.** PB es una aplicación web completamente *frontend* (JavaScript) que corre localmente en el navegador, por lo que aquí debemos implementar un *backend* que dé lugar al manejo de tales consultas.
  - Proveer una **URL específica** para acceder, una ***consumer key*** para identificar unívocamente al TC, y un ***shared secret*** o secreto compartido para asegurar los mensajes.
- El cuaderno de Jupyter debe:
  - Permitir configurar una conexión al TP usando la URL, la *consumer key* y el *shared secret* provistos por el mismo.

Parámetros para el lanzamiento

Para el momento de enviar el mensaje de lanzamiento por HTTP POST, LTI especifica los campos que pueden utilizarse como parámetros. Muy pocos campos son técnicamente requeridos, pero se debe considerar que cada TP puede tener diferentes requerimientos, según su funcionalidad. Por lo tanto, LTI recomienda que el TC provea tanta información como sea posible en cada lanzamiento para maximizar la probabilidad de que el TP obtenga los datos necesarios para funcionar apropiadamente, y que los TPs estén preparados para funcionar con información parcial, ya sea porque el TC no tiene la información o porque el TC fue configurado para no compartir esa información con el TP.

Los parámetros obligatorios son los que permiten identificar que se trata de un mensaje de *LTI launch*: la versión de LTI con la que se está trabajando, la localización del enlace que está siendo lanzado (de manera de diferenciar distintas instancias de lanzamiento de un mismo enlace) y la *consumer key* correspondiente al TC, además de los parámetros que se agregan al realizar el cifrado con OAuth1.0a. LTI provee otros parámetros no obligatorios para poder identificar al usuario, el rol del mismo (estudiante, instructor, etc.), datos de la cuenta que realiza el lanzamiento, datos de la institución que utiliza el TC, entre otros. Además, se pueden extender los campos especificados con otros personalizados, siempre que comiencen con el prefijo `ext_`.

En nuestro caso, nos conformamos en principio con que tanto el cuaderno como PB manejen un *LTI launch* con los parámetros mínimos obligatorios. Más adelante ambos pueden ampliarse para ajustarse a sus respectivas necesidades.

Al recibir el POST

Una vez que el TP ha recibido la consulta POST correspondiente al *LTI launch*, debe poder proceder de la siguiente manera [80]:

1. Verificar que se trata de un *LTI launch request*. Es decir, chequear que los parámetros obligatorios están presentes y no vacíos.

2. Validar el mensaje con OAuth1.0a, utilizando el *shared secret*.

3. Evaluar si están presentes y de forma correcta los parámetros que el TP requiere, además de los obligatorios.

4. Actualizar los datos del usuario que realiza el *launch* (en caso de que sea necesario y el TC lo informe) y establecer una sesión de usuario.

5. Redireccionar al usuario.

4.3.2. Implementación

Para implementar LTI 1.0 nos basamos en el enfoque presentado al inicio de este capítulo. La dinámica lograda se muestra en la Figura 4.2.

Aprovechando la arquitectura dada por Jupyter, desarrollamos una extensión que permite al usuario solicitar un nuevo ejercicio (de Pilas Bloques en este caso), y que el mismo se muestre en una celda del cuaderno, habiendo sido validada con un *LTI launch*. Creamos, entonces, una *nbextension* que se invoca al presionar su correspondiente botón en el cuaderno y que se comunica con el *notebook server* para identificar el usuario y el ejercicio a solicitar (siguiendo el procedimiento dado en las pruebas de concepto simples), para luego llevar a cabo el *LTI launch* utilizando el firmado digital de OAuth1.0a.[1] Para ello, genera una nueva celda y utiliza el *HTML built-in magic* [81] (que provee el Jupyter *kernel* de Python para ejecutar código HTML en una celda) para crear un formulario oculto con los parámetros establecidos y asociarlo a un *iframe* que mostrará la respuesta obtenida del lanzamiento. Luego se ejecuta automáticamente la celda, de manera que se envía el formulario y se realiza el *LTI launch*. El resultado es un *iframe* en la celda, conteniendo la instancia del ejercicio solicitada. Además, para ocultar de la vista del usuario el código HTML que da lugar a este lanzamiento, utilizamos la *nbextension "hide input"* [82] que permite ocultar el input de una celda, dejando sólo visible el resultado.

Del lado de Pilas Bloques, implementamos las rutinas necesarias para recibir la consulta POST solicitando el *LTI launch*. Para ello utilizamos la biblioteca PyLTI [83], que provee facilidades para realizar los controles correspondientes a la hora de recibir dicha consulta POST. Utilizando las herramientas provistas por esta biblioteca, la nueva rutina chequea que el POST se trate de un *LTI launch* y redirecciona a una URL específica.

De esta manera, configurando *consumer key* y *shared secret* tanto en Jupyter como en Pilas Bloques, se logra realizar el *LTI launch* con éxito: se muestra un *iframe* en la celda con el ejercicio de PB correspondiente, con el cual el usuario puede interactuar libremente.

4.3.3. Resultado de la POC

Se describe a continuación el caso de prueba abordado para esta prueba de concepto.

1. La persona que utiliza el CA se coloca en una celda cualquiera del cuaderno y presiona el botón *Solicitar Ejercicio* para solicitar un nuevo ejercicio de Pilas Bloques. (Figura 4.3)

[1] Utilizando esta biblioteca: https://github.com/ddo/oauth-1.0a

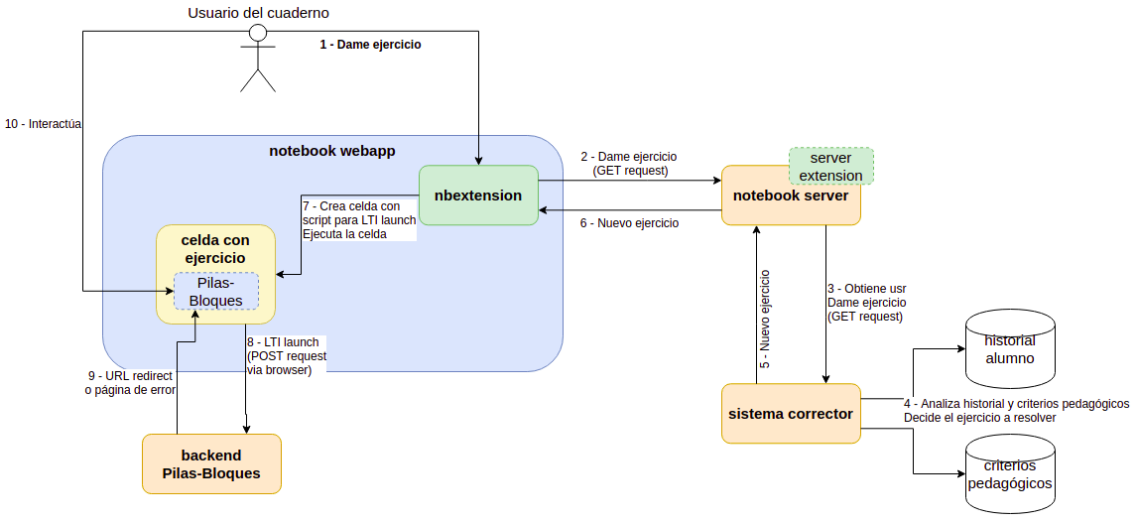


Fig. 4.2: Dinámica de la prueba de concepto LTI 1.0

2. Aparece una nueva celda en el CA conteniendo un *iframe* con una instancia de un ejercicio de Pilas Bloques. (Figura 4.4)

El *iframe* con el ejercicio a resolver no es más que el output correspondiente al *Basic LTI Launch*. Esto se verifica seleccionando la celda con el *iframe* y haciendo click en el botón correspondiente a la *nbextension hide-input*. Como se observa en la Figura 4.5, esto muestra el input de la celda con el ejercicio de Pilas Bloques, conteniendo el formulario HTML oculto con la petición para el *LTI Launch*.

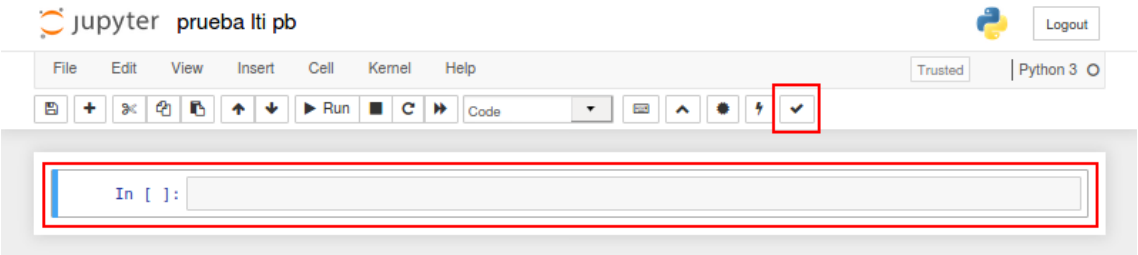


Fig. 4.3: Prueba LTI 1.0: el estudiante solicita un nuevo ejercicio de Pilas Bloques presionando en el nuevo botón *Solicitar Ejercicio* en el menú de controles.

4.4. LTI 1.1

La especificación LTI 1.1 [84] es la siguiente iteración de *Basic LTI* o LTI 1.0. La principal diferencia entre LTI 1.0 y LTI 1.1 es que en este último se permite el pasaje de datos desde el TP hacia el TC (en nuestro caso, desde Pilas Bloques hacia el Cuaderno) utilizando los *IMS Leraning Information Services* (LIS - Servicios de Información de Aprendizaje) [85] (Figura 4.6).

La especificación IMS LIS define cómo los sistemas administran el intercambio de información que describe personas, grupos, afiliaciones, cursos y resultados dentro del contexto

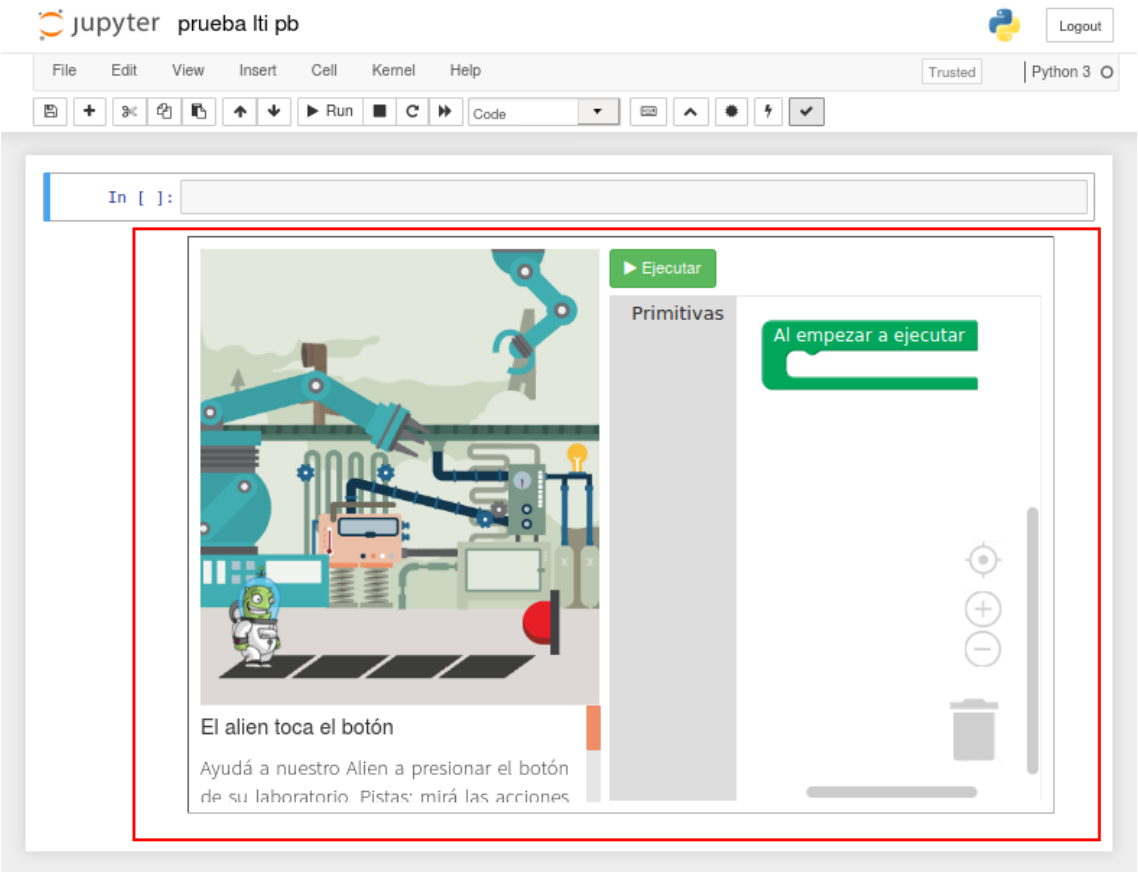


Fig. 4.4: Prueba LTI 1.0: se obtiene un nuevo ejercicio de Pilas Bloques

de aprendizaje. LTI no requiere servicios LIS, pero el TC puede enviar información al respecto al TP utilizando valores en la petición del *Launch*.

4.4.1. Idea principal

La idea general es que el TC provea de ciertos datos al TP de manera de permitirle invocar un subconjunto de estos servicios LIS, los cuales pueden ser provistos por el TC o bien ser otros sistemas. Estos servicios no necesitan ser invocados sincrónicamente en el contexto particular del *launch* y sesión de usuario: el TP puede retener los datos mencionados de un *launch* e invocar al servicio mucho después de que la sesión del usuario haya terminado.

LTI Basic Outcomes Service

El *Basic Outcomes Service* o Servicio Básico de Resultados permite el pasaje de calificaciones desde el TP hacia el TC para establecer, obtener y/o borrar resultados LIS asociados a una combinación particular de usuario/recurso. Para esto, se especifican nuevos campos a ser incluidos en el *launch*, los cuales identifican la URL del servicio de *outcomes* o resultados que se pone a disposición del TP, y una columna-fila correspondiente al *launch* dentro del libro de calificaciones del TC.

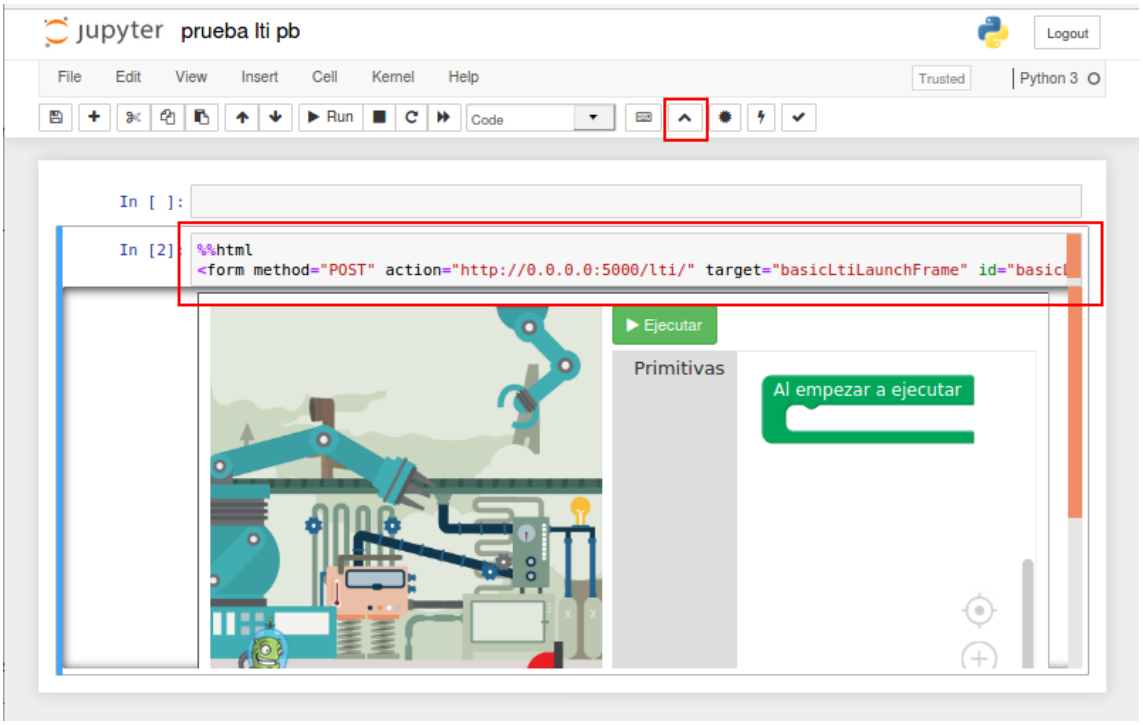


Fig. 4.5: Prueba LTI 1.0: Formulario HTML oculto para LTI Launch

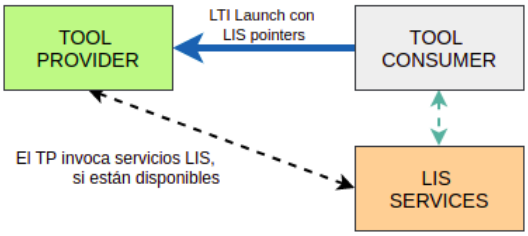


Fig. 4.6: El TP aprovechando los servicios LIS

El único tipo de calificación soportada por este servicio es la calificación numérica decimal en el rango 0.0 - 1.0, y la interpretación que se le dé al número informado dependerá de un acuerdo entre el TC y el TP, pudiendo simbolizar cuán correcto fue resuelto un ejercicio, o cuán completo, etc.

Dado que estos servicios utilizan firmado OAuth, para evitar revelar la clave y el secreto, la mejor práctica es que estos servicios sean invocados como servicios web servidor a servidor. No es posible proveer al navegador de la clave y el secreto para firmar estos mensajes sin arriesgarse a la pérdida de los mismos.

El flujo básico de un caso de uso para describir el retorno de un puntaje decimal desde el TP hacia el TC sería:

1. El docente crea un enlace a un recurso en un curso e indica que es para recibir puntajes desde el TP, configurando cualquier ruta necesaria entre el enlace LTI y el libro de calificaciones.

2. El usuario del TC lanza el enlace y el TC incluye el endpoint de servicio para el *LIS*

Basic Outcomes Service y el identificador correspondiente (fila/columna) en el libro de calificaciones del TC, para permitir al TP invocar al servicio y establecer, leer y/o borrar puntajes.

3. El sistema TP invoca al *Outcomes Service* del TC para establecer, leer y/o borrar puntajes según sea necesario. Estas operaciones pueden ser realizadas en cualquier momento de servidor a servidor (o sea, no sólo durante el período del launch).

Operaciones Básicas

A continuación se describen las operaciones básicas soportadas por el *LTI Basic Outcomes Service*:

- **Reemplazar un resultado** (*replaceResultRequest*): Establece una calificación numérica para el resultado de un recurso en particular. Depende del TC definir si esta operación realmente reemplaza la calificación o si el TC mantiene un historial con todas las calificaciones. El TC debe verificar la validez de la calificación entrante y debe responder con un código de "falla" (*failiure*) cuando una calificación esté por fuera del rango 0.0-1.0 o cuando no sea un número válido.
- **Leer un resultado** (*readResultRequest*): Retorna la calificación actual para un usuario/recurso particular. Si el TC mantiene un historial de calificaciones, el TP verá la calificación "más reciente". Si no existe la calificación solicitada (porque no se estableció o porque fue borrada), el TC debe retornar una respuesta válida con una calificación "vacía" (no retornar una calificación de 0.0 ni un estado de falla).
- **Borrar un resultado** (*deleteResultRequest*): Elimina una calificación para un resultado de un usuario/recurso particular. Si el TC mantiene un historial de calificaciones, debe definir su significado interno para esta operación. Puede borrar la calificación más reciente y volver a la anterior, o borrar todas las calificaciones. De cualquier manera, el TC debe proveer una vista al TP que refleje que ya no hay una calificación asociada al usuario/recurso en cuestión. Así que una operación de lectura luego de una operación de borrado normalmente debería retornar una calificación vacía, a menos que la calificación haya sido manipulada en el TC por algún otro servicio.

La especificación LTI 1.1 recomienda implementar este servicio de manera que el *endpoint* acepte cualquier petición bien formada y devuelva, al menos, una respuesta bien formada de "no admitido".

4.4.2. Implementación

A la hora de implementar el servicio de resultados, nos enfrentamos a una pregunta no trivial, referente a en qué momento se enviarán las calificaciones desde PB hacia el cuaderno. Para esta prueba de concepto en particular, elegimos enviar la calificación en el mismo momento en el que se calcula. Esto puede ser modificado más adelante si se desea otro comportamiento.

Para permitir el pasaje de resultados desde PB hacia el Cuaderno, tomamos el servidor que se construyó para corregir y decidir un próximo ejercicio, y le agregamos un *endpoint*

para el servicio de resultados. A su vez, añadimos a la consulta para el LTI launch los parámetros necesarios con la URL del servicio y un identificador para el campo de la base de datos donde el servicio luego almacenará la calificación. Por economía expresiva, seguimos llamando a este servidor como "sistema corrector"; si bien en este caso la corrección del ejercicio es llevada a cabo por el entorno de PB, otro entorno funcionando como TP podría no hacerlo, teniendo que delegar tal acción a nuestro *backend*.

En cuanto a Pilas Bloques, añadimos un método para calcular una calificación en el momento en que el usuario ejecuta el código del ejercicio, y asociamos este evento de ejecutar el código con el envío de la nota. Nuevamente utilizando la biblioteca PyLTI, creamos una rutina en el sistema que recibe la nota calculada (mediante una consulta POST desde el *frontend*), chequea que la nota esté asociada a una sesión de usuario y la envía de la manera especificada al *endpoint* indicado en el *LTI Launch* de esa sesión-ejercicio.

Es de notar que, en realidad, el puntaje que notifica PB no es enviado al Cuaderno, como en el caso de las pruebas de concepto simples (que utilizaban extensiones al cuaderno de Jupyter para ello), sino que es enviado directamente al *endpoint* que implementa el *LIS Basic Outcomes Service*.

4.4.3. Resultado de la POC

La dinámica de la operación de retorno de una calificación desde una instancia de Pilas Bloques lanzada dentro de un cuaderno de Jupyter via LTI se resume de esta manera (Figura 4.7):

1. La persona que está utilizando el Cuaderno Activo e interactúa con la instancia de PB via *LTI Launch* resuelve el ejercicio propuesto en el *iframe* y presiona el botón correspondiente para ejecutar la solución propuesta (también dentro del *iframe*).

2. Al momento de ejecutar la solución, PB envía la calificación para esta solución a su propio *backend*.

3. El *backend* de PB comunica esta calificación al *endpoint* indicado para este fin al momento de realizar el *LTI Launch*.

4. La rutina que maneja las consultas para este *endpoint* recibe la calificación y actualiza el historial del estudiante que está ejecutando el Cuaderno.

5. Luego de actualizar el historial, se notifica el éxito (o no) de dicha actualización al *backend* de PB.

6. El *backend* de PB notifica el éxito (o no) de la actualización al *frontend* ejecutando en el *iframe* del Cuaderno.

Para el usuario del Cuaderno, la experiencia de las pruebas de concepto para LTI 1.0 y LTI 1.1 son indistinguibles. En ambos casos se solicita un nuevo ejercicio presionando en el nuevo botón *Solicitar ejercicio* presente en la barra de herramientas, cuyo efecto es la aparición de una nueva celda con un *iframe* conteniendo una instancia de ejercicio de Pilas Bloques. Dentro de este *iframe*, el usuario interactúa de la misma forma con el entorno de PB, resolviendo el ejercicio y ejecutándolo.

En el caso particular de la prueba de concepto para LTI 1.1, la ejecución de la solución propuesta desencadena en PB el cálculo de la calificación y el envío de la misma mediante el *LTI Basic Outcomes Service*, reflejando el cambio en la base de datos que contiene el historial de este estudiante. Sin embargo, esto sucede entre los *backends* y el usuario no lo percibe.

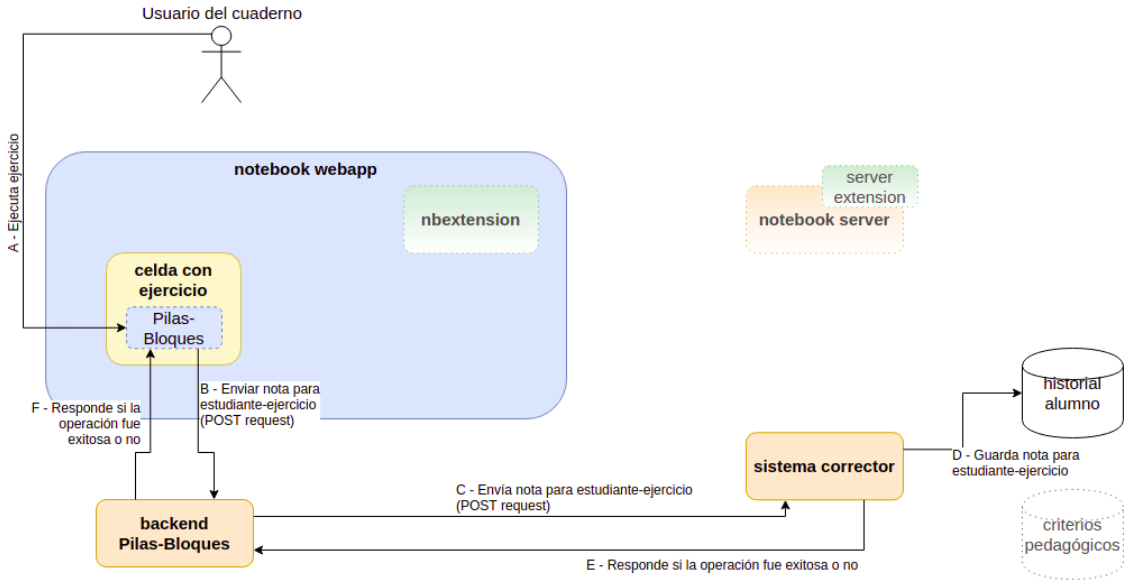


Fig. 4.7: Prueba de Concepto LTI 1.1

5. CONCLUSIONES Y TRABAJO FUTURO

5.1. Conclusiones

Cuando se piensa en combinar material producido por el docente con las intervenciones de los estudiantes, pueden surgir diversas formas de organizar tal diversidad de recursos. Particularmente, en la enseñanza de la programación habitualmente se requiere cambiar de vista de manera sistemática entre el contenido dado por el docente, procesadores de texto y entornos para ejecutar los programas.

En este contexto surgen los Cuadernos Activos (CA) como documentos holísticos que permiten incluir tanto contenido teórico como actividades prácticas, y donde todo este contenido se va presentando dinámicamente en función del desempeño de cada estudiante y las decisiones de los docentes. Estos CAs permiten la intervención directa por parte del estudiante, quien escribe, ejecuta y analiza sus programas en un mismo entorno.

La base para nuestros Cuadernos Activos son los Cuadernos Computacionales. Se trata de documentos que consisten en una sucesión de celdas que pueden contener texto, código y resultados de ejecutar el código, dando lugar a la creación de narrativas computacionales reproducibles. En particular, nos apoyamos en los Cuadernos de Jupyter (*Jupyter Notebook*), los cuales son de código abierto, son fácilmente extensibles y se han vuelto muy populares en el ámbito educativo y científico.

Para lograr la presentación de contenido en forma dinámica dentro del cuaderno, implementamos un servidor que oficia de corrector e indicador del próximo contenido a mostrar. Establecimos una API REST que permite realizar consultas, proporcionando datos de un estudiante y su respuesta a un ejercicio, almacenar dicha información, y obtener una respuesta con la información necesaria para mostrar el siguiente bloque en el CA. Desde el lado del Cuaderno, implementamos extensiones que permiten recuperar una respuesta provista en una celda y los datos del usuario para luego comunicarse con el *backend* que corrige e indica el siguiente contenido a mostrar.

Para cada prueba de concepto implementamos un *endpoint* en el *backend* corrector y un par de *Jupyter nbextension* y *Jupyter server extension* en el Cuaderno:

- En una primera prueba de concepto simple, una *nbextension* habilita un nuevo botón en la interfaz para que el usuario solicite la corrección de un ejercicio, y una *server extension* media entre ésta y el *backend* corrector. La respuesta de la corrección es mostrada en una nueva celda.

- Para la siguiente prueba de concepto simple, añadimos la noción de seguimiento de la actividad de un estudiante y la presentacion del contenido en forma dinámica. Una *nbextension* habilita un nuevo botón en la interfaz para que el usuario solicite el siguiente contenido, y una *server extension* media entre ésta y el *backend* corrector. En este caso, se presentan sucesiones de sumas de números naturales con distintos niveles de dificultad, y el sistema corrector "decide" qué secuencia de sumas deberá responder el estudiante en función de las respuestas provistas anteriormente. Esto implica, además, la creación de una base de datos para almacenar tanto los ejercicios como el desempeño de cada estudiante.

- La prueba de concepto subsiguiente contempla la incorporación de una nueva herramienta dentro del Cuaderno como parte de una actividad. Para ello implementamos la especificación LTI (*Learning Tools Interoperability*) en su versión 1.0, tanto en el Cuaderno (como *Tool Consumer*) como en la herramienta Pilas Bloques (como *Tool Provider*). Una *nbextension* habilita un nuevo botón en la interfaz para que el usuario solicite el siguiente desafío de PB y configura el lanzamiento de PB mediante LTI, y una *server extension* media entre la *nbextension* y el *backend* corrector. Tras presionar el botón, una nueva celda del Cuaderno presenta un *iframe* con el entorno de PB para la resolución del desafío informado desde el *backend*.

- Una nueva prueba de concepto extiende la anterior permitiendo a la herramienta Pilas Bloques informar al Cuaderno una calificación para el intento de resolución del desafío por parte del usuario. Para ello implementamos la versión 1.1 de LTI junto a un servicio básico de resultados que permite a PB comunicarse con un libro de calificaciones y reemplazar, leer o borrar un resultado. La ruta hacia dicho libro de calificaciones es informada en el lanzamiento LTI de la herramienta.

De esta manera, contamos con formas de convertir un Cuaderno Computacional de Jupyter en un Cuaderno Activo, mostrando de manera dinámica contenido diverso entre texto y ejercicios prácticos, incluyendo actividades provistas por herramientas externas al Cuaderno con la posibilidad de hacer un seguimiento del desempeño del estudiante en cada instancia presentada de dichas herramientas. Esto da lugar un trabajo más organizado y claro tanto para docentes como para estudiantes, sin tener que cambiar de vista sistemáticamente a la hora de poner en práctica los conceptos teóricos, y posibilitando el seguimiento de la actividad de cada estudiante para mayor comprensión, acompañamiento y adecuación de su trayectoria.

5.2. Trabajo Futuro

Una prueba de concepto que no exploramos en esta tesis es la de presentar desafíos de Pilas Bloques de manera secuencial y dinámica, de manera que el estudiante pueda solicitar un ejercicio de PB y el mismo sea asignado en función del desempeño del estudiante hasta el momento. Es decir, una versión de la prueba de concepto simple de secuencias de ejercicios con seguimiento de actividad, en la que los ejercicios son desafíos de PB lanzados mediante LTI. En este caso, el historial del usuario sería actualizado mediante el retorno de calificaciones desde PB, y el sistema corrector evaluaría el siguiente desafío a presentar basándose en tal historial.

Otro aspecto que requiere trabajo futuro es la generación de analíticas con base en un seguimiento más completo y estandarizado del comportamiento del estudiante. Considerando el uso de estos Cuadernos Activos incorporando herramientas externas al mismo, como el caso de nuestras pruebas de concepto con Pilas Bloques, especificaciones tales como xAPI y Caliper (analizadas en el primer capítulo de esta tesis) permitirían recabar datos relevantes según cada contexto de aprendizaje y cada herramienta utilizada la trayectoria del estudiante. Las analíticas resultantes podrían proveer información sobre los estudiantes que los docentes podrían utilizar para mejorar el proceso de enseñanza-aprendizaje, por ejemplo, personalizando contenidos o diseñando advertencias tempranas para detectar y tratar posibles obstáculos al aprendizaje.

A su vez, queda propuesto para una etapa posterior el desarrollo de la adaptabilidad del sistema que corrige y determina el siguiente contenido a mostrar de manera dinámica. En nuestras pruebas de concepto, el comportamiento de este sistema es muy simple, pero con miras a una utilización de Cuadernos Activos para cursos completos y complejos, el llamado *sistema corrector* debería a su vez complejizarse para abarcar un más amplio rango de caminos didácticos posibles. Una opción podría ser, por ejemplo, dotarlo de cierta inteligencia artificial al modo de los sistemas de recomendación.

Bibliografía

[1] David Koop and Jay Patel. Dataflow notebooks: encoding and tracking dependencies of cells. In *9th {USENIX} Workshop on the Theory and Practice of Provenance (TaPP 2017)*, 2017.

[2] Jonathan Bergmann and Aaron Sams. *Flip your classroom: Reach every student in every class every day*. International society for technology in education, 2012.

[3] B Fog and Clemens Nylandsted Klokmose. Mapping the landscape of literate computing. In *Proceedings of the 30th Annual Workshop of the Psychology of Programming Interest Group, ser. PPIG*, 2019.

[4] Donald Ervin Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

[5] Fernando Perez and Brian E Granger. Project jupyter: Computational narratives as the engine of collaborative data science. *Retrieved September*, 11(207):108, 2015.

[6] Keith J O'Hara, Doug Blank, and James Marshall. Computational notebooks for ai education. 2015.

[7] Adam Rule. Supporting Collaboration in Computational Notebooks with Cell Folding. https://medium.com/acm-cscw/supporting-collaboration-in-computational-notebooks-with-cell-folding-7a61160bdf98, 2018. [Online; recuperado el 18 de agosto de 2021].

[8] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016. 2016.

[9] Project Jupyter. Jupyter kernels. https://github.com/jupyter/jupyter/wiki/Jupyter-kernels. [Online; recuperado el 18 de agosto de 2021].

[10] Dropbox. https://www.dropbox.com. [Online; recuperado el 18 de agosto de 2021].

[11] Github. https://github.com/. [Online; recuperado el 18 de agosto de 2021].

[12] Jupyter nbviewer. https://nbviewer.jupyter.org/. [Online; recuperado el 18 de agosto de 2021].

[13] Markdown. https://markdown.es/. [Online; recuperado el 18 de agosto de 2021].

[14] SoS Polyglot Notebook. https://vatlab.github.io/sos-docs/. [Online; recuperado el 18 de agosto de 2021].

[15] Douglas S Blank, David Bourgin, Alexander Brown, Matthias Bussonnier, Jonathan Frederic, Brian Granger, Thomas L Griffiths, Jessica Hamrick, Kyle Kelley, M Pacer, et al. nbgrader: A tool for creating and grading assignments in the jupyter notebook. *The Journal of Open Source Education*, 2(11), 2019.

[16] Jeff Brown. Using jupyterhub in the classroom: Setup and lessons learned. *International Journal of Software Engineering & Applications*, 9(2):01–08, 2018.

[17] Wolfram Reaserch Inc. Cuadernos Wolfram. https://www.wolfram.com/notebooks/. [Online; recuperado el 18 de agosto de 2021].

[18] Wolfram Reaserch Inc. Wolfram Cloud. https://www.wolframcloud.com/. [Online; recuperado el 18 de agosto de 2021].

[19] Wolfram Reaserch Inc. CDF - Formato de Documento Computable. https://www.wolfram.com/cdf/. [Online; recuperado el 18 de agosto de 2021].

[20] Stephen Wolfram et al. *The MATHEMATICA® book, version 4*. Cambridge university press, 1999.

[21] Jiři Kopeckỳ. Using computable document format in teaching mathematics. *R&E-SOURCE*, 2014.

[22] Ana Hadiana, Tao Zhang, Vuthichai Ampornaramveth, and Haruki Ueno. Web e-learning system based on concept of online whiteboard. In *WEBIST (2)*, pages 387–391, 2006.

[23] Educreations Inc. Educreations. https://www.educreations.com/. [Online; recuperado el 18 de agosto de 2021].

[24] ShowMe. https://www.educreations.com/. [Online; recuperado el 18 de agosto de 2021].

[25] SP Controls Inc. Doceri. https://www.educreations.com/. [Online; recuperado el 18 de agosto de 2021].

[26] Eva Martín Roda and Silvana Sassano Luiz. Posibilidades de google drive para la docencia a distancia y en el aula. *Didáctica geográfica*, (16):203–220, 2015.

[27] Google. Google Colaboratory. https://colab.research.google.com. [Online; recuperado el 18 de agosto de 2021].

[28] Anders Fernström, Ingemar Axelsson, Peter Fritzson, Anders Sandholm, and Adrian Pop. Omnotebook–interactive wysiwyg book software for teaching programming. In *Proc. of the Workshop on Developing Computer Science Education–How Can It Be Done*, 2006.

[29] Peter Fritzson, Adrian Pop, Karim Abdelhak, Adeel Asghar, Bernhard Bachmann, Willi Braun, Daniel Bouskela, Robert Braun, Lena Buffoni, Francesco Casella, et al. The openmodelica integrated environment for modeling, simulation, and model-based development. *Modeling, Identification and Control*, 41(4):241–295, 2020.

[30] Peter Fritzson and Vadim Engelson. Modelica—a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer, 1998.

[31] Mohsen Torabzadeh-Tari, Peter Fritzson, Adrian Pop, and Martin Sjölund. Generalization of an active electronic notebook for teaching multiple programming languages. In *IEEE EDUCON 2010 Conference*, pages 1081–1083. IEEE, 2010.

[32] Khalid Huseynov. Introduction to data science with apache zeppelin.

[33] Apache Spark. Apache spark. *Retrieved January*, 17:2018, 2018.

[34] Sagemath Inc. Sage Worksheets. https://doc.cocalc.com/sagews.html. [Online; recuperado el 18 de agosto de 2021].

[35] Sagemath Inc. CoCalc - Collaborative Calculation and Data Science. https://cocalc.com/. [Online; recuperado el 18 de agosto de 2021].

[36] P Szabó and J Galanda. Sage math for education and research. In *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 1–4. IEEE, 2017.

[37] William Watson and Sunnie Lee Watson. An argument for clarity: What are learning management systems, what are they not, and what should they become. 2007.

[38] edX Inc. Open edX. https://open.edx.org/. [Online; recuperado el 18 de agosto de 2021].

[39] Blackboard Inc. Blackboard. https://www.blackboard.com/. [Online; recuperado el 18 de agosto de 2021].

[40] Edmodo. Edmodo. https://new.edmodo.com/. [Online; recuperado el 18 de agosto de 2021].

[41] Ajlan Al-Ajlan and Hussein Zedan. Why moodle. In *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 58–64. IEEE, 2008.

[42] Jean-Marie Maes. Chamilo 2.0: A second generation open source e-learning and collaboration platform. *International Journal of Advanced Corporate Learning (iJAC)*, 3(3):26–31, 2010.

[43] Instructure Inc. Canvas LMS. https://www.instructure.com/es/canvas. [Online; recuperado el 18 de agosto de 2021].

[44] I Ketut Sudarsana, Ida Bagus Made Anggara Putra, I Nyoman Temon Astawa, and I Wayan Lali Yogantara. The use of google classroom in the learning process. In *Journal of Physics: Conference Series*, volume 1175, page 012165. IOP Publishing, 2019.

[45] Google. Google Workspace for Education Fundamentals. https://edu.google.com/products/workspace-for-education/education-fundamentals/. [Online; recuperado el 18 de agosto de 2021].

[46] Randy Joy Magno Ventayen, Karen Lea A Estira, Ma Jasmine De Guzman, Christian Mark Cabaluna, and Nieva N Espinosa. Usability evaluation of google classroom: Basis for the adaptation of gsuite e-learning platform. *Asia Pacific Journal of Education, Arts and Sciences*, 5(1):47–51, 2018.

[47] José Barrón, Joel Quintanilla, J Rico, L Gordillo, and B Ojeda. Uso de un sistema para la gestión del aprendizaje (lms) de código libre en la universidad tecnológica del suroeste de guanajuato (utsoe). *Ciencias Administrativas y Sociales Handbook TV*, pages 328–340, 2014.

[48] John Palfrey and Urs Gasser. Interop, 2012.

[49] Ricardo Queirós, José Paulo Leal, and José Carlos Paiva. Integrating rich learning applications in lms. In *State-of-the-art and future directions of smart learning*, pages 381–386. Springer, 2016.

[50] Simon Booth, Susi Peacock, and Stephen P Vickers. Plug and play learning application integration using ims learning tools interoperability. In *Ascilite*, pages 143–147, 2011.

[51] Gavin Henrick. Moodle as a the central hub of learning with tools plugged in-learning tool interoperability. 2012.

[52] Rustici Software. SCORM. https://scorm.com/. [Online; recuperado el 18 de agosto de 2021].

[53] Advanced Distributed Learning Initiative. https://adlnet.gov/. [Online; recuperado el 18 de agosto de 2021].

[54] Oliver Bohl, Jörg Scheuhase, Ruth Sengler, and Udo Winand. The sharable content object reference model (scorm)-a critical review. In *International Conference on Computers in Education, 2002. Proceedings.*, pages 950–951. IEEE, 2002.

[55] James Ballard. Interoperability Standards. https://www.slideshare.net/Nine_Lanterns/interoperability-lti-and-experience-api-formerly-tincan, 2014. [Online; recuperado el 18 de agosto de 2021].

[56] Mariluz Guenaga and Pablo Garaizar. Del análisis del aprendizaje a su mejora: Retos y oportunidades. *Revista Iberoamericana de Tecnologías del Aprendizaje-VAEP-RITA*, 3(4):197–198, 2015.

[57] Cecilia Avila, Silvia Baldiris, Ramon Fabregat, and Sabine Graf. Co-creación y evaluación de recursos educativos abiertos inclusivos y accesibles: un mapeo hacia el ims caliper. *Revista Iberoamericana de Tecnologías del/da Aprendizaje/Aprendizagem, IEEE VAEP-RITA*, 3(4):219–229, 2015.

[58] Jose Luis Santos, Sten Govaerts, Katrien Verbert, and Erik Duval. Goal-oriented visualizations of activity tracking: a case study with engineering students. In *Proceedings of the 2nd international conference on learning analytics and knowledge*, pages 143–152, 2012.

[59] Kin Chew Lim. Case studies of xapi applications to e-learning. In *The twelfth international conference on eLearning for knowledge-based society*, pages 3–1, 2015.

[60] Jonathan M Kevan and Paul R Ryan. Experience api: Flexible, decentralized and activity-centric data collection. *Technology, knowledge and learning*, 21(1):143–149, 2016.

[61] Aneesha Bakharia, Kirsty Kitto, Abelardo Pardo, Dragan Gašević, and Shane Dawson. Recipe for success: lessons learnt from using xapi within the connected learning analytics toolkit. In *Proceedings of the sixth international conference on learning analytics & knowledge*, pages 378–382, 2016.

[62] Advanced Distributed Learning (ADL). xAPI-Spec. https://github.com/adlnet/xAPI-Spec. [Online; recuperado el 18 de agosto de 2021].

[63] Nicholas Lewkow, Neil Zimmerman, Mark Riedesel, and Alfred Essa. Learning analytics platform, towards an open scalable streaming solution for education. *International Educational Data Mining Society*, 2015.

[64] IMS Global Learning Consortium. Caliper Analytics. https://www.imsglobal.org/activity/caliper. [Online; recuperado el 18 de agosto de 2021].

[65] Elena Díaz García and Eduardo Cabrera Granado. Guide to jupyter notebooks for educational purposes. 2018.

[66] Jupyter Team. The Jupyter Notebook. https://jupyter-notebook.readthedocs.io, 2015. [Online; recuperado el 18 de agosto de 2021].

[67] Jupyter Team. Jupyter Project Documentation. https://jupyter.readthedocs.io, 2015. [Online; recuperado el 18 de agosto de 2021].

[68] Martin Sústrik et al. Zeromq. *Introduction Amy Brown and Greg Wilson*, 2015.

[69] Michael Dory, Allison Parrish, and Brendan Berg. *Introduction to Tornado: Modern Web Applications with Python*. " O'Reilly Media, Inc", 2012.

[70] Project Jupyter Team. JupyterHub. https://jupyterhub.readthedocs.io, 2016. [Online; recuperado el 18 de agosto de 2021].

[71] James Turnbull. *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

[72] Alfredo Sanzo, Fernando Schapachnik, Pablo Factorovich, and Federico Sawady O'Connor. Pilas bloques: A scenario-based children learning platform. In *2017 Twelfth Latin American Conference on Learning Technologies (LACLO)*, pages 1–6. IEEE, 2017.

[73] Fundación Sadosky. Program.AR. https://program.ar/. [Online; recuperado el 18 de agosto de 2021].

[74] Fundación Sadosky. Fundación Sadosky - Investigación y Desarrollo en TIC. http://www.fundacionsadosky.org.ar/. [Online; recuperado el 18 de agosto de 2021].

[75] Matt Copperwaite and Charles Leifer. *Learning Flask Framework*. Packt Publishing Ltd, 2015.

[76] IMS Global Learning Consortium Inc. Product Certifications. https://site.imsglobal.org/certifications. [Online; recuperado el 18 de agosto de 2021].

[77] IMS Global Learning Consortium Inc. Learning Tools Interoperability® 1.0. https://www.imsglobal.org/specs/ltiv1p0. [Online; recuperado el 18 de agosto de 2021].

[78] IMS Global Learning Consortium Inc. The OAuth 1.0 Protocol. https://tools.ietf.org/html/rfc5849, 2010. [Online; recuperado el 18 de agosto de 2021].

[79] JavaScript. https://www.javascript.com/. [Online; recuperado el 18 de agosto de 2021].

[80] IMS Global Learning Consortium Inc. Learning Tools Interoperability®: Verifying Launch Messages. https://www.imsglobal.org/learning-tools-interoperability-verifying-launch-messages. [Online; recuperado el 18 de agosto de 2021].

[81] The IPython Development Team. Built-in magic commands. https://ipython.readthedocs.io/en/stable/interactive/magics.html. [Online; recuperado el 18 de agosto de 2021].

[82] Jupyter notebook extensions - Hide Input. https://github.com/ipython-contrib/jupyter_contrib_nbextensions/tree/master/src/jupyter_contrib_nbextensions/nbextensions/hide_input. [Online; recuperado el 18 de agosto de 2021].

[83] MIT Office of Digital Learning. PyLTI - LTI done right. https://github.com/mitodl/pylti. [Online; recuperado el 18 de agosto de 2021].

[84] IMS Global Learning Consortium Inc. Learning Tools Interoperability® 1.1. https://www.imsglobal.org/specs/ltiv1p1. [Online; recuperado el 18 de agosto de 2021].

[85] IMS Global Learning Consortium Inc. IMS Global Learning Information Services Specification Primer. http://www.imsglobal.org/lis/lisv2p0p1/LISSpecPrimerv2p0p1.html. [Online; recuperado el 18 de agosto de 2021].