Revista EIA

UNIVERSIDAD EIA

Mejora de Prácticas en Desarrollo de Software en un Contexto Universitario de I+D+I

Revista EIA
ISSN 1794-1237
e-ISSN 2463-0950
Año XIX/ Volumen 22/ Edición N.44
Julio - diciembre 2025
Reia4425 pp. 1-31

Publicación científica semestral
Universidad EIA, Envigado, Colombia

Para citar este artículo / To reference this article /
Diaz Zamboni, J. E.; García, J. Caupolicán Ré, L.; Salinas Sosa, S.; Vertiz Del Valle, D. C.; Rizzato, J. F. Cabrera, J. M.; Gonzalez, C. I. y Insfrán, J. F.
Mejora de Prácticas en Desarrollo de Software en un Contexto Universitario de I+D+I
Revista EIA, 22(44), Reia4425 pp. 1-31
https://doi.org/10.24050/reia.v22i43.1885

Autor de correspondencia:
Diaz Zamboni, J. E.
Bioingeniería, Doctodardo en Ingeniería
Correo electrónico:
javier.diaz@uner.edu.ar

Recibido: 16-04-2025
Aceptado:10-06-2025
Disponible online: 01-07-2025

Javier Eduardo Diaz Zamboni[1]
Justo García[1]
Lautaro Caupolicán Ré[1]
Santiago Salinas Sosa[1]
Diana Carolina Vertiz Del Valle[1]
Juan Francisco Rizzato[1]
Juan Manuel Cabrera[1]
César Iván Gonzalez[1]
Jordán Francisco Insfrán[1]

1. Universidad Nacional de Entre Ríos (UNER), Argentina

Resumen

Este trabajo presenta un estudio de caso enmarcado en una investigación participativa, centrado en las actividades de diseño y desarrollo de software de un grupo de docentes y estudiantes pertenecientes a un laboratorio de investigación, desarrollo e innovación en el área de la informática y computación aplicada en la Facultad de Ingeniería de la Universidad Nacional de Entre Ríos. A partir de la información de un conjunto de proyectos de software llevados a cabo por el grupo y las opiniones recopiladas mediante una encuesta a los miembros de cada equipo, este estudio –de naturaleza mixta– busca identificar aprendizajes y oportunidades de mejora, y proponer acciones para fortalecer las prácticas de ingeniería de software del grupo. El análisis de datos combina un enfoque cuantitativo, basado en la asignación de puntuaciones y generación de indicadores de valoración de proyectos a partir de las respuestas cerradas de opción única de la encuesta, con un análisis cualitativo de contenido de las respuestas abiertas mediante un proceso de filtrado y síntesis asistido con inteligencia artificial generativa. Los resultados obtenidos de la encuesta reflejan importantes aprendizajes y aspectos a mejorar que se discuten con aportes técnicos y metodológicos, entre los que se destacan la incorporación de registros de decisiones arquitectónicas, propuestas de temas para la formación de los miembros del grupo, una forma de colaboración

público-privada en la que el grupo puede contribuir a la innovación tecnológica en empresas y reflexiones sobre el desarrollo de software en la universidad. El análisis realizado en este trabajo sobre las prácticas de ingeniería de software del grupo, es un aporte más al conjunto de esfuerzos que dan importancia y valor al software que se desarrolla en universidades.

Palabras clave: universidad, desarrollo de software, ingeniería de software, lecciones aprendidas, mejora de prácticas, encuesta, proyecto de software, diseño y arquitectura de software, programación, pruebas de software

Improvement of Software Development Practices in a University R&D&I Context

Abstract

This paper presents a case study framed in a participatory research, focused on the analysis of the software design and development activities of a group of teachers and students belonging to a research, development and innovation laboratory in the area of computer science and applied computing at the Faculty of Engineering of the National University of Entre Ríos. Based on information from a set of software projects carried out by the group and the opinions collected through a survey to the members of each team, this study –of a mixed nature– aims to recognize learnings, identify areas for improvement and propose actions to strengthen the software engineering practices of the group. The data analysis combines quantitative approach, based on the assignment of scores and generation of project valuation indicators from the closed single-choise responses of the survey, with a qualitative content analysis of the open-ended responses through a filtering and synthesis process assisted with generative artificial intelligence. The results obtained from the survey reflect important learnings and areas for improvement that are discussed with technical and methodological contributions, among which are the incorporation of architectural decisions records, topics proposals for the training of the members of the group, a form of public-private collaboration in which the group can contribute to technological innovation in companies and reflections on software development at the university. The analysis carried out in this work on the group's software engineering practices is one more contribution to the set of efforts that give importance and value to the software developed in universities.

Keywords: university, software development, software engineering, lessons learned, practice improvement, survey, software project, software design and architecture, programming, software testing

1. Introducción

El diseño y desarrollo de software en la universidad es una actividad frecuente de gran valor. No solo da soporte a la docencia, sino que también es crucial en otras actividades universitarias fundamentales como la investigación, la extensión y la innovación. En muchos casos, los equipos de cátedra desarrollan herramientas propias para apoyar sus clases, mientras que los diversos grupos, ya sean de investigación, extensión e incluso aquellos formados temporalmente para colaborar con otros organismos y empresas, desarrollan software en el marco de sus proyectos. Sin embargo, a pesar de su relevancia, es común que el software no esté lo suficientemente documentado, que la gestión o el mantenimiento sea limitado o que su difusión, tanto para usuarios internos (como nuevos miembros del equipo) como externos (otros grupos, organizaciones o la sociedad en general), sea ineficaz. Esto dificulta no solo su reutilización y sostenibilidad en el tiempo, sino también su disponibilidad para terceros mientras el desarrollo tenga un ciclo de vida activo.

Un desafío particular en este contexto es que el desarrollo de software en la universidad enfrenta un alto recambio de personas en los equipos. Gran parte del software es desarrollado por estudiantes, quienes realizan contribuciones valiosas, pero cuya permanencia en los proyectos es temporal. Como resultado, muchos desarrollos quedan descontinuados o requieren un esfuerzo significativo para ser retomados por nuevos integrantes, lo que dificulta la continuidad y evolución de los proyectos.

A menudo, el desarrollo de software en la universidad se lleva a cabo sin una planificación estructurada, respondiendo a necesidades inmediatas más que a una estrategia de largo plazo. Si bien en algunos casos se aplican algunas metodologías y herramientas de gestión que permiten almacenar y mantener el código fuente, estas prácticas suelen ser insuficientes para garantizar la sostenibilidad del software. Esto se debe a que no existe una solución única para abordar esta problemática: el software es inherentemente complejo (Booch, 1996) y su desarrollo es una actividad esencialmente humana, cuya mejora no depende únicamente de metodologías o herramientas, sino también de comprender el contexto en el que se

desarrolla (Sommerville, 2005). Analizar cómo se trabaja en cada entorno, identificar buenas prácticas y reconocer áreas de mejora permite tomar decisiones que fortalezcan esta actividad universitaria, alineándola con los objetivos de cada grupo y de la institución.

Este desafío no se limita a un contexto particular dentro de la universidad (por ejemplo, desarrollo de software para la docencia), y distintos autores han abordado la problemática del desarrollo de software en la universidad desde diversas perspectivas y situaciones. Por ejemplo, (Katz et al., 2019) analizaron el papel del personal universitario dedicado al desarrollo de software de investigación, destacando su importancia en la academia y la falta de reconocimiento profesional en este ámbito. El estudio presenta tres modelos organizativos de ingenieros de software de investigación en universidades del Reino Unido y Estados Unidos, examinando su estructura, funcionamiento e impacto en la producción y mantenimiento del software. Los autores resaltan la necesidad de mayor apoyo institucional para estos grupos, incluyendo su integración en propuestas de financiamiento y la sostenibilidad a largo plazo del software de investigación.

En el artículo de (Ekedahl et al., 2018), se presentan una serie de lecciones aprendidas en base a la experiencia práctica en el contexto de la investigación y el trabajo académico. Las experiencias se basan en información tanto de proyectos de investigación como en un curso y varias tesis de pregrado desarrolladas durante un período de tres años. Las lecciones aprendidas incluyen: la existencia de una relación de compromiso entre la creación rápida de prototipos y la seguridad del software, el código fuente proporcionado como ejemplo no cumple con estándares de software de producción, el cumplimiento de los estándares acelera el desarrollo, los sistemas de IoT carecen de soporte de depuración, las licencias de código abierto varían, la interoperabilidad entre plataformas es deficiente y las tarifas de servicio varían entre plataformas, dificultando la comparación de costos. En el artículo de (Polack-Wahl, 2006), se plantea una problemática de la enseñanza: los profesores de ingeniería de software se enfrentan a un gran desafío cuando intentan asignar proyectos del mundo real a los estudiantes, ya que no siempre logran reproducir la incertidumbre y los cambios constantes que

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

estos implican. Además, este enfoque incrementa su carga de trabajo y puede generar desinterés en los estudiantes, quienes suelen preocuparse más por la calificación que por el aprendizaje. En este sentido, la autora comparte experiencias obtenidas de diferentes tipos de proyectos de ingeniería de software, mostrando los efectos positivos de proyectos procedentes de organizaciones sin ánimo de lucro y departamentos universitarios.

Los estudios mencionados reflejan la diversidad de la problemática del desarrollo de software en el ámbito universitario y, en general, proponen acciones contextualizadas, a partir del análisis de proyectos, lecciones aprendidas e identificación de áreas de mejora. En esta misma línea, el presente artículo aborda un estudio de caso enmarcado en una investigación participativa, centrado en la actividad de diseño y desarrollo de software de un grupo de docentes y estudiantes que conforman un laboratorio de investigación, desarrollo e innovación en la Facultad de Ingeniería de la Universidad Nacional de Entre Ríos (FIUNER). El objetivo es identificar aprendizajes y oportunidades de mejora en las prácticas del grupo y proponer acciones que fortalezcan sus prácticas de ingeniería de software, a partir de la sistematización de información proveniente de proyectos concretos, experiencias compartidas y percepciones de sus integrantes. Se parte de la hipótesis de que la sistematización de estas experiencias, junto con la reflexión colectiva de los participantes, permite identificar aprendizajes significativos y proponer acciones viables para mejorar las prácticas del grupo. Para ello, se analizan doce proyectos de software desarrollados por el grupo y los resultados de una encuesta aplicada a sus integrantes. Este trabajo constituye así una reflexión documentada identificando áreas de mejora y proponiendo acciones para su perfeccionamiento en línea con el objetivo planteado. Asimismo, aporta elementos que puedan ser de utilidad para experiencias similares en otros contextos universitarios.

Si bien se intenta alcanzar cierta generalidad, es importante reconocer que los resultados y reflexiones están fuertemente condicionados por la forma de trabajo del grupo y las particularidades de su contexto. Este contexto abarca no solo la organización y dinámicas propias del grupo, sino también el

entorno institucional, normativo y cultural en el que desarrolla su actividad. Por esta razón, antes de abordar la metodología empleada en este trabajo, se presenta una sección que describe el grupo y su contexto. Luego se detalla el enfoque metodológico utilizado para la recopilación y análisis de datos. A continuación, en la sección resultados y discusión se presentan los aprendizajes y propuestas de mejora obtenidas de la encuesta, identificando áreas de mejora y proponiendo acciones que abordan los diversos desafíos del grupo de manera integral. Finalmente, en la sección de conclusiones, se sintetizan los principales aportes del trabajo.

El grupo y su contexto

Los docentes de la UNER tienen funciones que incluyen actividades de docencia, investigación, desarrollo e innovación, y extensión universitaria. Es decir, la docencia universitaria implica realizar un conjunto de actividades integralmente que hacen al desarrollo pleno de las funciones docentes (Estatuto de la UNER, 2023). Esta concepción integral influye en la organización del trabajo, habilitando una sinergia entre formación, desarrollo e investigación en proyectos tecnológicos.

Estas funciones se llevan a cabo en las facultades, cada una con su propia organización interna, en consonancia con su oferta académica y perfil institucional. En el caso de la FIUNER, dicha organización se estructura en departamentos académicos, los cuales constituyen subunidades funcionales que integran campos disciplinares, que se desarrollan a través de asignaturas, espacios curriculares y núcleos de extensión e investigación, desarrollo e innovación (I+D+i). Los núcleos constituyen estructuras clave para la articulación de la docencia con la investigación y la extensión, permitiendo potenciar recursos y capacidades institucionales para el cumplimiento de los objetivos institucionales (Consejo Directivo FIUNER, 2023b). Esta estructura favorece la formación de grupos de trabajo interdisciplinarios y colaborativos, como los que requiere el desarrollo de software en contextos universitarios.

Los núcleos pueden ser grupos de estudio, laboratorios de I+D+i, centros e institutos, y se pueden integrar con personal de la propia

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

facultad, de otras facultades de la misma universidad, investigadores y profesionales de otros organismos, universidades y otros organismos similares de promoción y financiamiento del sistema científico-técnico, así como por profesionales independientes, becarios, tesistas de grado y de posgrado, personal administrativo y técnico (Consejo Directivo FIUNER, 2023a).

El grupo al que se hace referencia en este trabajo conforma un laboratorio de I+D+i, dedicado a la investigación y desarrollo en el área de la informática y computación aplicada. El laboratorio está integrado por docentes y estudiantes de la FIUNER, con un total de seis miembros docentes y seis estudiantes de pregrado y grado al momento de la escritura de este trabajo. Los estudiantes pertenecientes al laboratorio poseen algún tipo de beca que otorga la universidad o algún otro organismo, que tiene el objetivo de fomentar y desarrollar determinadas competencias, dando un apoyo económico para incorporarse en actividades extracurriculares. Esta dinámica favorece la formación temprana en investigación, aunque también introduce una rotación frecuente que puede afectar la continuidad de los proyectos.

En esta línea, los miembros estudiantes del laboratorio realizan actividades formativas en investigación y desarrollo con un tiempo de permanencia que está entre dos a tres años. Pasado este tiempo dejan el laboratorio debido, principalmente, a la finalización de sus estudios en la universidad. El trabajo en equipo del grupo se organiza en proyectos, actividades y tareas. Se realizan reuniones quincenales, con presentación de seminarios y resúmenes de cada miembro, en las que cada uno cuenta en qué ha estado trabajando, avances y dificultades. En estas reuniones también se dedica un tiempo para ensayar ideas para nuevas propuestas de proyectos. Este tipo de dinámica busca generar un entorno de aprendizaje continuo y colaboración entre los miembros del grupo.

La actividad de I+D+i del laboratorio tiene su enfoque principal en el área de conocimiento de la ingeniería de software conocida como construcción de software (Washizaki, 2024). En esta línea, actualmente en el laboratorio se están desarrollando temas relacionados a Internet de las Cosas (IoT), interfaces de usuario, lenguajes de programación, algoritmos y estructuras de datos,

visión computacional, entre otros. Estos temas responden a líneas actuales de investigación y desarrollo tecnológico, con potencial de transferencia tanto a la práctica profesional como a propuestas pedagógicas.

En su mayor parte, los proyectos son de pequeña escala y tienen un alcance acotado, para permitir su estudio y exploración. Otros, en menor proporción, pueden alcanzar mayor complejidad por su tiempo de desarrollo o por el alcance del proyecto. Además, se busca integrar a la docencia el conocimiento y las experiencias obtenidas en esta actividad. En este sentido, el laboratorio impulsa el estudio y desarrollo de aplicaciones, así como propuestas pedagógicas y didácticas para la enseñanza de la programación, aprovechando los aprendizajes generados en sus actividades de investigación y desarrollo. Esta articulación entre proyectos técnicos y prácticas docentes constituye un rasgo distintivo del grupo, que se busca potenciar con este trabajo.

Al momento de la escritura del presente, artículo algunos de los proyectos de software del laboratorio se encuentran en desarrollo, otros cuentan con la distribución de una versión estable, y otros han sido suspendidos. En determinados casos, el objetivo fue desarrollar un prototipo en el contexto de una investigación; en otros, realizar una actividad de vinculación tecnológica, o bien una con fines pedagógicos o didácticos. Varios proyectos tuvieron su origen como actividad formativa dentro del contexto de becas estudiantiles; otros, como aplicaciones desarrolladas en el marco de un proyecto de investigación y desarrollo del equipo; y otros, surgieron a partir de la actividad docente o de investigaciones no formalizadas en proyectos. Esta diversidad de orígenes y objetivos en los proyectos analizados aporta una visión rica y multifacética sobre cómo se produce y gestiona el desarrollo de software en contextos universitarios.

2. Metodología

Se consideraron doce proyectos de desarrollo de software realizados por el grupo en los últimos cuatro años. Cada proyecto cuenta sus objetivos, actividades y tareas, miembros asignados, forma de

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

seguimiento y un responsable del mismo. La documentación de cada proyecto se gestiona a través de una carpeta compartida en la nube y, al menos, un repositorio en la plataforma de desarrollo colaborativo GitHub para la gestión del desarrollo.

Encuesta a los miembros de los equipos de cada proyecto

La encuesta se diseñó en secciones basadas en una selección de categorías integradoras, que relacionan tanto aspectos técnicos, como también habilidades blandas. Las categorías seleccionadas fueron cuatro: diseño y arquitectura, programación y lenguajes de programación, pruebas de software y aportes al desarrollo de software en la universidad. Esta última categoría incluye dos subcategorías que indagaban sobre aspectos del desarrollo de software en la universidad y la posibilidad de que los proyectos en los que participaron los encuestados contribuyan a la enseñanza y aprendizaje en el ámbito universitario (ver Anexo III - Cuestionario y resultados de procesamiento cualitativo de la encuesta).

Si bien el cuestionario no fue sometido a una validación formal, fue revisado y mejorado por miembros del grupo. El diseño se basó principalmente en la experiencia del grupo en los proyectos de desarrollo y en encuestas de características similares encontradas en la literatura. Las respuestas fueron completadas de manera confidencial y voluntaria. Posteriormente, fueron anonimizadas para su procesamiento y publicación. Por su parte, se debe reconocer que el formato de la encuesta y la participación tanto de miembros como de ex miembros del grupo introducen sesgos. No obstante, la naturaleza de la investigación es participativa y contribuye, más allá de los sesgos y la precisión estadística, al análisis y reflexión del grupo.

Finalmente, la encuesta incluía hacia el final un conjunto de preguntas abiertas para que los encuestados mencionaran otros temas que consideraran relevantes y que no hubieran sido abordados previamente en las preguntas categorizadas. Si bien se recibieron pocas respuestas a estas últimas preguntas abiertas, las que aportaron información relevante a los fines de este estudio fueron incorporadas directamente en el análisis general, sin pasar por el

proceso de preprocesamiento y síntesis con inteligencia artificial generativa aplicado al resto de las preguntas como se detalla a continuación.

Los encuestados completaron la encuesta de manera no anónima, respondiendo específicamente por cada proyecto en el que participaron. Para el análisis cuantitativo, se asignaron puntuaciones numéricas a las respuestas de opción única según su nivel de valoración, de acuerdo con la categoría evaluada (adecuación, completitud, impacto, etc.). La escala utilizada otorga 3 puntos a las respuestas más positivas, 2 a las intermedias, 1 a las menos favorables y 0 a las opciones no aplicables o de menor valor categórico. Estas preguntas están asociadas a un observable en alguna dimensión de análisis, por ejemplo la percepción de nivel de calidad de la documentación del proyecto.

A partir de estas respuestas, se calculó una valoración individual del proyecto para cada encuestado. Este indicador corresponde a la suma de los puntajes obtenidos en las preguntas aplicables, dividida por la puntuación máxima posible. De esta forma, se obtiene un valor entre 0 y 1 que representa la percepción global del encuestado sobre el proyecto. Esta valoración incluye las siguientes dimensiones evaluadas. La valoración individual de un proyecto se calcula como se muestra en la ecuación (1).

$$V_i = \frac{\sum_{j=1}^{n} p_{ij}}{3 \cdot n} \tag{1}$$

donde V_i es la valoración individual del integrante *i*, p_{ij} es la puntuación de la pregunta j otorgada por el integrante *i* y *n* es el número total de preguntas con respuesta cerrada de opción única.

Para obtener una visión global de cada proyecto, se computó el promedio de las valoraciones individuales, generando un indicador denominado valoración general del proyecto, dado por la fórmula de la ecuación (2):

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

$$VG = \frac{1}{m}\sum_{i=1}^{m} V_i \quad (2)$$

donde *VG* es la valoración general y *m* es el número de integrantes que respondieron sobre el proyecto considerado. Las determinaciones de estas valoraciones fueron realizadas en hojas de cálculo. Asimismo, se obtuvieron la distribución y el promedio de las respuestas a cada una de las preguntas de opción única, con el objetivo de obtener una representación global de cada observable de interés por separado, que son las percepciones asociadas a cada aspecto consultado. Para ello, cada promedio fue redondeado al entero más próximo 3, 2, 1 ó 0 con el fin de asignarle una categoría a la percepción en la escala Muy Adecuada, Medianamente Adecuada, Poco Adecuada, No Adecuada, respectivamente.

Para analizar las preguntas en las que los encuestados expresaron opiniones o reflexiones, se llevó a cabo un análisis cualitativo de contenido asistido con el modelo de inteligencia artificial generativa que incorpora ChatGPT (OpenAI, 2025). Las respuestas obtenidas a cada una de este tipo de preguntas, se procesaron de la misma manera. Primero, se realizó un filtrado inicial para descartar respuestas, o partes de ellas, que fueran muy específicas para un proyecto en particular y que no aportaran a la visión general de la consulta. Luego, en un segundo filtrado, las respuestas se simplificaron en frases concisas, identificando patrones de generalización, priorizando conceptos clave por sobre menciones a herramientas específicas. Finalmente, el conjunto preprocesado se analizó con ChatGPT para generar una síntesis que reflejara los temas más recurrentes y relevantes identificados en las respuestas.

Por ejemplo, al aplicar el proceso de preprocesamiento a una de las respuestas obtenidas para la pregunta: ¿Qué aspectos del diseño y la arquitectura mejorarías en este proyecto? se obtuvo inicialmente la siguiente respuesta:

> *"Haber desarrollado más el diseño y la arquitectura, es un proyecto que al ser pensado como SaaS fue ya pensado con*

cierto diseño y arquitectura propia del caso. Hubiese estado bueno haber incorporado o incorporar personas ajenas al desarrollo para clarificar algunos aspectos de diseño y/o arquitectura. Fomentar los espacios de intercambio."

Luego de aplicar el primer filtrado, se obtuvo:

"Haber desarrollado más el diseño y la arquitectura; haber incorporado o incorporar personas ajenas al desarrollo para clarificar algunos aspectos de diseño y/o arquitectura. Fomentar los espacios de intercambio."

Finalmente, luego del segundo filtrado, se obtuvo la versión simplificada:

"desarrollar más el diseño y la arquitectura; incorporar más personas al proyecto; fomentar espacios de intercambio"

Este proceso se aplicó a la totalidad de las respuestas para cada pregunta. La lista de respuestas simplificadas se utilizó como entrada para el proceso de síntesis generado por ChatGPT, cuya instrucción (prompt) puede consultarse en la Tabla 1.

Tabla 1. *Instrucción utilizada para la generación de síntesis*

Analiza el siguiente conjunto de datos, que consiste en respuestas preprocesadas de encuesta sobre [*aquí se especifica tema, por ejemplo: "diseño y arquitectura"*]. Tu tarea es generar un párrafo que resuma de manera coherente y precisa los principales [*aprendizajes o propuestas de mejoras*] expresadas en las respuestas. El párrafo debe cumplir con los siguientes criterios: claridad y cohesión: Integra de forma fluida las ideas similares sin redundancias; estilo formal y académico: usa un tono profesional y neutral, adecuado para un artículo; neutralidad: limita el resumen a describir los aprendizajes o propuestas, sin agregar conclusiones o juicios adicionales; estructuración: organiza la información en un solo párrafo, asegurando las interrelaciones entre los distintos puntos mencionados.

Una vez obtenidas las síntesis generadas por el modelo, se realizó un análisis comparativo entre estas y las respuestas originales de los encuestados. Esto permitió verificar la fidelidad de la síntesis y garantizar que los resultados reflejaran con precisión las opiniones y propuestas recogidas. En este último análisis comparativo, donde se encontrara alguna cuestión menor de redacción o en la forma de expresar una idea, se corrigió manualmente para dar la forma final a la síntesis de las respuestas.

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

Apoyados en la metodología presentada y, puesto que las encuestas se realizan a los integrantes del equipo por proyecto, de estas se obtienen aprendizajes y aspectos a mejorar por proyecto. A partir de estos, se identifican áreas de mejora y se realizan propuestas de mejora que buscan abordar los desafíos de manera integral. Estos se desarrollan en lo que sigue del trabajo.

3. Resultados y discusión

En la Tabla 2 se sintetizan los proyectos de diseño y desarrollo considerados en este trabajo. En ella se destacan el nombre del proyecto, la categoría de software a la que pertenece, su objetivo o propósito, el estado actual, repositorio o referencias asociadas y la Valoración General del proyecto obtenido del procesamiento de los resultados de la encuesta. Más detalles de los proyectos se pueden consultar en el Anexo I - Resumen de proyectos.

Tabla 2. *Lista de proyectos considerados en el análisis del presente artículo*

Nombre del proyecto	Categoría de software	Objetivo o propósito del proyecto	Estado actual	Repositorio o referencia	Valoración General
UsiLac 4.0	IoT - prototipo - simulador	Servicio a terceros - vinculación tecnológica	En desarrollo	No disponible	74%
Monitor de Agua Potable	IoT - prototipo	Propuesta de innovación	Suspendido	No disponible	83%
Monitor de consumo eléctrico	Prototipo IoT	Propuesta de innovación	En desarrollo	No disponible	67%
MicroStitch	Aplicación científica	Software para aplicaciones de investigación. Sistema de registro de imágenes panorámicas	En desarrollo	https://github.com/FIUNER-LICA/micro-stitch	85%
Biblioteca ad hoc C++ contratos	Biblioteca de clases	Biblioteca de clases	Versión estable	https://github.com/FIUNER-LICA/biblioteca-dbc-cpp	78%
Ubiquibot	Aplicación interactiva - prototipo	Investigación de Interfaz de Usuario en IoT	Versión estable	No disponible	75%
SmartClaim	Pedagógico	Proyecto diseño y desarrollo para estudiantes de un curso de diseño y programación orientada a objetos	En desarrollo	https://github.com/FIUNER-LICA/JADiCC-2024-Experiencia-Docente-Resultados	30%
Bastón Háptico	Recurso didáctico	Para ser programado por estudiantes, presentación en ferias de carreras	En desarrollo	No disponible	85%
AI Mimic Hand	Recurso didáctico - Robótica	Para ser programado por estudiantes, presentación en ferias de carreras	En desarrollo	No disponible	91%
Sitio web del grupo	Sitio Web	Visibilidad de las actividades del grupo	Versión estable	https://lica.ingenieria.uner.edu.ar/	65%
Herramienta GazeTracker	Aplicación para investigación	Software para aplicaciones de investigación	En desarrollo	No disponible	52%
Monitoreo cluster	Herramienta IT	Herramienta de monitoreo de recursos	En desarrollo	No disponible	67%

En general, los proyectos muestran una valoración general positiva. Sin embargo el proyecto SmartClaim es una excepción. Al ser una propuesta pedagógica, aunque existe una versión desarrollada por docentes, el desarrollo principal del proyecto está a cargo de los estudiantes que cursan la materia impartida por los docentes del grupo. Debido a que los estudiantes no participaron en la encuesta, algunas preguntas de valoración no fueron respondidas.

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

En relación a los resultados crudos de la encuesta, se obtuvieron un total de 20 respuestas que fueron procesadas conforme a lo planteado en la metodología. Si bien el tamaño de la muestra puede parecer pequeño, representa la totalidad de las respuestas recibidas de forma voluntaria de la mayoría de los miembros y exmiembros del grupo, por lo que se considera adecuada y representativa para los fines del estudio de caso. Las respuestas de la encuesta anonimizadas pueden verse en el Anexo II - Respuestas crudas anonimizadas y su procesamiento cuantitativo y el resultado del procesamiento de las respuestas puede verse en el Anexo III - Cuestionario y resultados de procesamiento cualitativo de la encuesta.

En relación a las respuestas de opción única, la tabla 3 presenta la lista de preguntas de las nueve preguntas consideradas y los observables asociados para la determinación de la valoración individual por proyecto.

Tabla 3. *Lista de preguntas de opción única, observables asociados y niveles de percepción*

Pregunta de opción única de la encuesta	Observable asociado	Nivel promedio de valoración de las percepciones
¿Qué tan adecuado consideras el trabajo de diseño y arquitectura realizado en este proyecto?	Percepción sobre el diseño y la arquitectura	Medianamente Adecuada
¿Qué tan adecuada consideras la elección del o de los lenguajes de programación para este proyecto?	Percepción sobre la programación y los lenguajes utilizados	Muy Adecuada
¿Se emplearon buenas prácticas de programación durante el desarrollo?	Percepción sobre la aplicación de buenas prácticas	Medianamente Adecuada
¿Te han resultado apropiadas las pruebas realizadas al software de este proyecto?	Percepción sobre las pruebas realizadas	Medianamente Adecuada
¿Crees que este proyecto contribuye o ha contribuido al desarrollo de software en la universidad?	Percepción sobre el aporte al desarrollo de software universitario	Muy Adecuada
¿Qué nivel de impacto consideras que ha tenido o tiene el proyecto en la comunidad destino o en los usuarios finales?	Percepción sobre el impacto en la comunidad o usuarios	Medianamente Adecuada
¿Consideras que el proyecto podría ser valioso como herramienta de enseñanza o aprendizaje en la universidad?	Percepción sobre el valor del proyecto en enseñanza y aprendizaje	Medianamente Adecuada
¿Se han cumplido los objetivos del proyecto?	Percepción sobre el cumplimiento de los objetivos	Medianamente Adecuada
¿Cómo clasificarías la calidad de la documentación del proyecto?	Percepción sobre la calidad de la documentación	Medianamente Adecuada

Los resultados muestran una valoración global mayormente positiva sobre los proyectos analizados. En general, en las preguntas cerradas se puede observar que presentan una percepción medianamente adecuada en la mayoría de las categorías y muy adecuada en lenguajes de programación y en el desarrollo de software en la universidad. Mayor información sobre la distribución puede verse en el Anexo III - Cuestionario y resultados de procesamiento cualitativo de la encuesta.

Los resultados y discusión de las preguntas abiertas de la encuesta se organizan en cuatro grandes subsecciones: diseño y arquitectura, programación y lenguajes de programación, pruebas de software y desarrollo de software en la universidad. Cabe recordar que esta última subsección integra las dos categorías de la encuesta: aspectos generales del desarrollo de software en la universidad y su posible contribución a la enseñanza y el aprendizaje. En cada subsección, se presentan dos partes. En la primera, se presentan los resultados como una síntesis de los aprendizajes y aspectos a mejorar recolectados a partir de las encuestas, según el proceso descrito en la metodología. En la segunda, se discuten estos resultados realizando identificación de áreas y propuestas de mejora. La discusión se desarrolla en base a trabajos propios y de otros autores, con el objetivo de enriquecer el aporte del estudio.

Diseño y Arquitectura

Resultados: aprendizajes y aspectos a mejorar

Respecto de la pregunta sobre qué aprendizajes relacionados con diseño y arquitectura se obtuvieron realizando cada proyecto, se destaca en las respuestas el enfoque centrado en el usuario, incluyendo la importancia de ponerse en el rol de este para definir mejor los requerimientos y validar estrategias antes de iniciar el desarrollo. También, destacan aprendizajes relacionados con la planificación inicial de los proyectos, arquitecturas modulares, organización en capas y la aplicación de patrones de diseño para garantizar un código reutilizable y fácil de integrar en nuevos proyectos. Se mencionan aprendizajes sobre protocolos de comunicación, herramientas para automatización de tareas y

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

configuración. Además, se resalta el valor del trabajo en equipo, la documentación y la coordinación de tareas, así como la importancia de realizar pruebas para detectar fallas y proponer mejoras. También, se reconoce el valor de la aplicación de metodologías iterativas para analizar problemas y estructurar aplicaciones de manera organizada.

Un análisis general de las respuestas de los encuestados a la pregunta sobre aspectos a mejorar en el diseño y la arquitectura de cada proyecto, evidencia la necesidad e importancia de contar con una planificación inicial sólida y bien definida. Asimismo, se plantea la necesidad de establecer objetivos más claros y específicos desde el inicio, junto con etapas definidas, especialmente en el tiempo dedicado a la investigación. Además, se destaca la importancia de incorporar metodologías más sistemáticas y de documentar las decisiones de diseño y arquitectura. Otro aspecto mencionado en las respuestas, es la necesidad de priorizar la escalabilidad y mantenibilidad en el diseño y la arquitectura. Algunos comentarios en la encuesta sugieren la incorporación de herramientas como contenedores. También se plantea mejorar el diseño y uso de pruebas, realizar más pruebas con usuarios reales y estudios de casos de uso.

Discusión: áreas y propuestas de mejoras

Una forma integradora de abordar aspectos señalados en relación a la arquitectura es a través de un enfoque de reconocimiento de las principales características arquitectónicas en los proyectos (Richards y Ford, 2020), con el objetivo de determinar el estilo arquitectónico más adecuado para cada caso. Contar con arquitecturas iniciales establecidas como modelos facilita la comprensión del sistema para quienes lo especifican, estudian, utilizan y desarrollan. El uso de prototipos de aplicaciones que implementan una determinada arquitectura permite mantener o generalizar estos modelos para futuros diseños y desarrollos en el marco de la actividad universitaria. Esto no solo facilita una mayor claridad y organización en el desarrollo de prototipos de aplicaciones, sino que también crea una base sólida para extender y adaptar el trabajo en futuros proyectos o nuevas áreas de exploración científica e ingenieril (Fairbanks, 2023). Además, se pueden incorporar aspectos de seguridad en los prototipos, disminuyendo los riesgos asociados al

prototipado rápido, abordando los desafíos de seguridad (Ekedahl et al., 2018).

Respecto a la documentación sobre diseño y arquitectura, la reciente incorporación de un repositorio de registros de decisiones arquitectónicas (ADR, por sus siglas en inglés, *Architectural Decision Record*) en la organización de GitHub del laboratorio (Diaz Zamboni, 2024), aunque aún en desarrollo, promete ser un recurso de documentación valioso para el grupo. Su objetivo es documentar, de forma accesible, las decisiones tomadas por el equipo sobre aspectos importantes de alguna arquitectura. Cada ADR registra la decisión de arquitectura adoptada, su contexto y sus implicaciones, lo que permite guiar el desarrollo y evitar cambios innecesarios de herramientas, lenguajes de programación, o frameworks sin fundamentos sólidos. Además, el repositorio incluye una selección curada de ADRs comunes y reutilizables, que pueden servir como base para nuevos proyectos. Se recomienda que cada proyecto cuente con su propio conjunto de ADRs, adaptados a su contexto particular, pero apoyados en este repositorio base como punto de partida. Investigaciones recientes respaldan estas prácticas, mostrando que el uso de ADRs favorece la toma de decisiones colaborativas, mejora la trazabilidad del diseño, mejora significativamente la eficiencia de revisión del código y se está adoptando de forma creciente en proyectos de código abierto, (Buchgeher et al., 2023; Godbole, 2024).

Cabe destacar que el potencial de los ADRs se amplifica al integrarse con otras documentaciones, como los casos de uso, el análisis de requerimientos y la documentación de procesos internos del grupo. Esto permite registrar no solo las decisiones técnicas, sino también el proceso de diseño centrado en el usuario, identificando, priorizando con precisión sus necesidades y la forma de abordarlo.

Programación y lenguajes de programación

Resultados: aprendizajes y aspectos a mejorar

La encuesta revela diversos aprendizajes en programación y lenguajes de programación. Se destaca el manejo de protocolos de comunicación, así como la importancia de contar con un sistema de simulación de datos cuando no se dispone del hardware

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

real. En cuanto a buenas prácticas de desarrollo, se mencionan aprendizajes en documentación y control de versiones, enfatizando la necesidad de utilizar estilos de código apropiados, generar documentación clara y gestionar adecuadamente los repositorios en frameworks de desarrollo. En términos de lenguajes específicos, en los proyectos se profundizaron conocimientos en C++, Python y JavaScript, explorando aspectos como concurrencia, asincronía y modularización. Además, se destaca el trabajo con frameworks como Flask para el desarrollo web y SQLAlchemy para la gestión de bases de datos, adquiriendo experiencia en validación de datos y manejo de sesiones. En el ámbito de visión computacional, se experimenta con procesamiento de imágenes en Python utilizando OpenCV y Numpy, así como en el desarrollo de interfaces gráficas con Qt Creator y PySide. También, se aprendió sobre la integración entre back-end y front-end, programación orientada a objetos y el uso de Bash para la automatización de tareas. Algunas respuestas dan cuenta de aprendizajes relacionados a la documentación en entornos visuales como Node-RED y su integración con GitHub. Finalmente, se destaca que, si bien Python es un lenguaje versátil que facilita la implementación de funcionalidades complejas, puede presentar limitaciones en cuanto a rendimiento.

En cuanto a los aspectos a mejorar en programación y la elección de lenguajes en el proyecto, se resaltó en las respuestas la importancia de aplicar buenas prácticas, documentar la lógica del funcionamiento y establecer convenios claros sobre la escritura del código fuente, incluyendo la declaración de variables, métodos y clases. También se propuso la modularización como estrategia para mejorar el código y la incorporación apropiada de mecanismos de manejo de errores. En términos de rendimiento, algunas respuestas mencionan que, ante la presencia de cuellos de botella en tareas de procesamiento, ciertas partes del software podrían implementarse en C++ para mejorar la eficiencia. Respecto a las herramientas de desarrollo, se sugirió la evaluación de nuevos entornos de desarrollo para embebidos, buscando una plataforma más flexible y escalable, por ejemplo, que permita de forma sencilla realizar procesos de compilación cruzada para simplificar el despliegue de aplicaciones en

lugar de tener que compilar manualmente cada combinación código fuente-hardware.

Discusión: áreas y propuestas de mejoras

El análisis de la encuesta refleja seguridad en los miembros del grupo respecto al conocimiento y aplicación de lenguajes de programación en los diferentes proyectos, como también lo reflejó la percepción obtenida en el análisis cuantitativo. A pesar de que el debate sobre qué lenguaje es el más adecuado para enseñar o utilizar es común (Sobral, 2021), no se evidenció en las respuestas la preferencia de un lenguaje específico. En general, las respuestas manifestaron diversos aprendizajes en diferentes lenguajes de programación. Se destacó la pertinencia de lenguajes flexibles y con capacidad de prototipado rápido para etapas iniciales del desarrollo. Además, surgieron propuestas para mejorar el rendimiento mediante el uso de lenguajes como C++ en contextos de procesamiento intensivo. Sin embargo, incorporar criterios de decisión sobre qué lenguaje es más apropiado para cada aplicación o según el propósito del desarrollo (por ejemplo, prototipar un algoritmo, desarrollar una biblioteca o interactuar con un middleware) resulta valioso, ya que permite tomar decisiones fundamentadas y coherentes, además de facilitar la documentación y la comunicación dentro del equipo. Dichas decisiones pueden documentarse mediante ADRs -como se propuso al final de la sección anterior-, que ofrecen un registro claro y accesible de las elecciones realizadas. Por ejemplo, para la creación de un chatbot, un ADR puede definir y fundamentar el lenguaje a utilizar, ya sea un lenguaje gráfico basado en flujos o un lenguaje de programación basado en texto.

Pruebas de software

Resultados: aprendizajes y aspectos a mejorar

En las respuestas se mencionan diversos aprendizajes en relación con las pruebas de software en cada proyecto, destacándose la importancia de la automatización de pruebas y la correcta interpretación de los resultados. Aprendieron a complementar diferentes tipos de pruebas, diseñar estrategias para detectar errores y evaluar el cumplimiento de los requerimientos. Un aspecto clave

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

identificado fue la dependencia del código con la tecnología utilizada, lo que puede dificultar o incluso impedir la implementación de ciertas pruebas. También se resalta el impacto de la complejidad oculta en el rendimiento de diferentes funciones en embebidos y se menciona la exploración de métodos efectivos para evaluar algoritmos, incluyendo la generación de gráficos, documentación y la medición del tiempo de ejecución en distintas secciones del software. Las respuestas destacan que el diseñar datos de prueba representativos mejora su calidad, y que realizar pruebas a componentes de manera aislada antes de integrarlos, ahorra tiempo y esfuerzo. Se destaca, también, la importancia de realizar pruebas frecuentes durante el desarrollo para detectar problemas en etapas tempranas, así como la necesidad de incluir pruebas de usabilidad. Además, se reconoció que seguir buenas prácticas de programación facilita el desarrollo de las pruebas y que el acceso a documentación y foros en línea fueron clave para la resolución de problemas y la optimización del proceso de prueba. Finalmente, se identificaron desafíos en las pruebas de chatbots en comparación con las pruebas en otros tipos de proyectos, porque emergen dificultades para automatizar el análisis de respuestas correctas del bot ante entradas y salidas con formato de lenguaje natural donde importa que las respuestas tengan sentido para un intérprete humano.

En cuanto a los aspectos a mejorar en las pruebas del software en los proyectos, se sugiere en las respuestas la necesidad de profundizar en las pruebas unitarias, incluyendo pruebas de transmisión de datos entre dispositivos y pruebas de integración para verificar el funcionamiento del sistema en su conjunto. Se destaca la importancia de mejorar las pruebas de seguridad, estabilidad y manejo de recursos, además de incorporar pruebas automatizadas para optimizar el proceso. También se propone ampliar la cobertura de pruebas, asegurando evaluar el rendimiento de los distintos algoritmos implementados. Se menciona la necesidad de documentar detalladamente el diseño de las pruebas, especificando con mayor claridad los requisitos funcionales y no funcionales para garantizar una evaluación completa. Algunas respuestas sugieren aumentar el número de personas en la prueba de aceptación y estructurar mejor las tareas mediante listas de verificación para asegurar una cobertura

amplia de funcionalidades. Además, se plantea la idea de entrenar un modelo de inteligencia artificial para generar datos de entrada (e.g. mensajes al chatbot), con el fin de detectar fallos en el sistema. Finalmente, se resalta la importancia de una mayor rigurosidad en el testing, la realización de pruebas con distintas lógicas de programación para medir tiempos de respuesta y también se enfatiza el realizar actividades de formación en esta área.

Discusión: áreas y propuestas de mejoras

El análisis de estas opiniones plantea claramente el desafío de reforzar el proceso de las pruebas de software, no solo incorporando pruebas en el ciclo de desarrollo, sino también adoptando criterios para establecer qué tipos de pruebas son más apropiadas en cada proyecto. Como señala (Khorikov, 2020), no se trata simplemente de escribir pruebas, sino de diferenciar entre aquellas que realmente aportan valor al proyecto y aquellas que resultan inútiles, evitando esfuerzos innecesarios en la escritura de tests que no contribuyen a la calidad del software. En este sentido, es crucial que los miembros del equipo desarrollen habilidades para integrar pruebas documentadas en el proceso de desarrollo y diseñar pruebas efectivas, incluyendo la distinción entre pruebas de falsación y pruebas formales (Martin, 2018).

En esta misma línea, los autores de este trabajo consideran que la incorporación de diseño por contratos (Meyer, 1992) en el desarrollo de software puede complementar y potenciar el proceso de pruebas de software, aportando a varios de los aspectos a mejorar mencionados en la encuesta. Los contratos pueden aportar a la detección y corrección temprana de errores, ya que si forman parte del código fuente y se expresan con una sintaxis diferenciada, proporcionan los elementos necesarios para explotar la potencia de las pruebas de software, tanto empíricas como formales. En esta dirección, se destacan algunos avances realizados por el grupo. Por un lado, aquellos relacionados con la incorporación de la programación basada en contratos en la enseñanza (Insfrán y Diaz Zamboni, 2023), en línea con otros estudios (De Carvalho et al., 2020;

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

Kraemer, 2018). Por otro lado, se destacan aportes al desarrollo de software confiable en el lenguaje C++ (García et al., 2024).

Desarrollo de software en la universidad

Resultados: aprendizajes y aspectos a mejorar

Las respuestas mencionan diversos aprendizajes adquiridos durante el desarrollo de software en la universidad. Se destaca el valor del trabajo en equipo y la importancia de la coordinación de tareas. Se mencionan aprendizajes respecto a comunicar resultados y a pensar en el desarrollo de software en la universidad como un producto distribuible y construído de manera colaborativa. Se resaltan la diversidad de proyectos en el grupo y reflexiones sobre la falta de aprovechamiento del software desarrollado en el ámbito académico y la necesidad de que la universidad fomente su utilización. Se valoran las buenas prácticas de programación adoptadas por el grupo y se destaca como valiosa la experiencia de trabajar en un proyecto con cierto grado de avance para identificar mejoras a través de las pruebas.

En cuanto a los aprendizajes obtenidos aplicables a la enseñanza y el aprendizaje en la universidad, en las respuestas se destaca el valor de algunos proyectos para el aprendizaje de conceptos relacionados a la orientación a objetos, arquitectura, y características arquitectónicas como la confiabilidad y la robustez del software. Se valora positivamente el impacto motivacional que puede tener el trabajar en proyectos con cierto nivel de aplicación real, contribuyendo a la preparación de los estudiantes para futuros desafíos laborales. Además, se enfatiza el desafío de integrar código desarrollado por diferentes personas, reflejando la realidad del trabajo en equipo en el ámbito profesional. Se destaca el potencial que existe en estos proyectos para introducir a estudiantes de los primeros años en competencias clave del desarrollo de software y la gestión de proyectos.

Como aspectos a mejorar en los proyectos para maximizar el impacto en los usuarios finales y la comunidad, en las respuestas se destaca la importancia de una mayor difusión de resultados. Se propone, por ejemplo, mantener actualizado el sitio web, publicar

regularmente las nuevas producciones del grupo y establecer un marco claro para la distribución del software. Asimismo, se sugiere incluir usuarios en el proceso de desarrollo, para evaluar el uso del software y obtener retroalimentación directa. Se recomienda mejorar la documentación para facilitar la adopción del software y optimizar la compatibilidad con hardware, así como la gestión eficiente de recursos energéticos en algunos casos. En particular, se destaca el caso del proyecto UsiLac 4.0, en el que contar con un prototipo, permitiría a las empresas analizar la incorporación de tecnología de software antes de realizar inversiones.

En lo que concierne a la adaptación de los proyectos para su aplicación en la enseñanza y el aprendizaje, en las respuestas a la encuesta se proponen algunas estrategias. En una respuesta, se plantea incorporar el proyecto en un curso extracurricular. En otra, se sugiere adaptar su estructura en módulos independientes para facilitar la comprensión, reorganizar el código para permitir un enfoque de aprendizaje progresivo y crear tutoriales paso a paso. También, se menciona la posibilidad de incorporar elementos interactivos, como cuestionarios, y vincular el proyecto a plataformas académicas. En el proyecto AI Mimic Hand se propuso implementar conectividad remota para que los estudiantes puedan realizar pruebas a distancia. Otras propuestas incluyen la incorporación de un modelo de lenguaje en el chatbot para mejorar la interacción y el uso de diagramas UML para clarificar su arquitectura. Asimismo, se plantea la modularización como una estrategia clave para que los estudiantes comprendan mejor conceptos como concurrencia y asincronía, además de facilitar la adaptación del proyecto a distintos problemas específicos de estudio.

Discusión: áreas y propuestas de mejoras

Un área de mejora que surge con mayor frecuencia en las opiniones vertidas en la encuesta es la necesidad de mejorar la difusión de los proyectos de software realizados en la universidad. Una forma de hacerlo es contar con un repositorio público en un sistema de control de versiones que incluya documentación clara, tutoriales e instaladores cuando sea necesario. Además, el uso de

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

una plataforma de lanzamientos permitiría publicar las versiones estables, mientras que un repositorio de preservación digital facilita la asignación de un DOI, asegurando su accesibilidad, citabilidad y preservación a largo plazo. GitHub ofrece tanto el sistema de control de versiones como la plataforma de lanzamiento. Además, es posible integrarlo con Zenodo que permite crear registros públicos de los desarrollos de software permitiendo visibilizar de una forma accesible los desarrollos del grupo de referencia. Este proceso de distribución de software se ha aplicado en los proyectos de la Biblioteca de contratos en C++ y MicroStitch. Sin embargo, aún existen importantes desafíos por resolver. Uno de ellos es evaluar si la difusión de los desarrollos universitarios alcanza a las personas y sectores interesados, generando el impacto esperado. Otro desafío clave es definir el marco normativo de la universidad en relación con la producción de software, incluyendo la elección de licencias y las regulaciones que rigen su desarrollo y uso.

Otra área de mejora identificada es la formación y capacitación de los miembros del grupo. Los temas propuestos por los encuestados están relacionados principalmente con lenguajes de programación y pruebas de software. No obstante, del análisis de los resultados de la encuesta, surgen dos temas adicionales de gran relevancia. Uno está directamente relacionado con una de las categorías seleccionadas y se puede denominar fundamentos de diseño y arquitectura de software. El otro tema que aparece de manera recurrente en las respuestas es el concepto de API (del inglés, *Application Program Interface*). En cuanto a la formación en lenguajes de programación, se propone trabajar en una estrategia pedagógica que incorpore una mirada más general de los lenguajes programación, permitiendo el desarrollo de habilidades que faciliten el autoaprendizaje de nuevos lenguajes. Esto puede lograrse mediante la incorporación de contenidos que fundamentan los paradigmas de programación y los lenguajes, y su relación con la arquitectura (Martin, 2018). Esto puede ser articulado con propuestas que promuevan el desarrollo del pensamiento computacional como base para la formación en ingeniería de software (Guevara-Reyes et al., 2025). A su vez, el fortalecimiento de competencias en diseño y arquitectura se alinea con recientes investigaciones que destacan la necesidad de acercar

la formación académica a los desafíos de la industria, incorporando estrategias pedagógicas centradas en proyectos, atributos de calidad y patrones arquitectónicos, como parte esencial de la formación en arquitectura de software (Pantoja Yépez et al., 2024). Adicionalmente, este enfoque favorece la incorporación de conceptos de diseño y arquitectura aplicables tanto a la enseñanza de la informática y computación en los espacios curriculares existentes, así como en instancias de formación extracurricular como cursos o talleres.

Otra área de mejora a fortalecer es la vinculación del grupo con el entorno. Es común que los grupos de investigación y desarrollo de las universidades, colaboren con empresas y otros organismos externos. La experiencia con el proyecto UsiLac 4.0 destaca un modelo de colaboración público-privada, en el que la investigación y desarrollo pueden contribuir a la innovación tecnológica en pequeñas empresas que no cuentan con un área propia de I+D+i.

Según la revisión realizada por (Catalá-Pérez y de-Miguel-Molina, 2018), este tipo de colaboración es necesaria y beneficiosa tanto para los actores públicos, como para los privados, dado que es un mecanismo esencial de gobernanza e impulso de la innovación. Sin embargo, desde la perpectiva de la formación de los estudiantes, (Polack-Wahl, 2006) advierte que, si bien este tipo de proyectos permite a los estudiantes trabajar en un producto real, también puede deteriorar su formación académica. Como alternativa, la autora propone que la integración de estudiantes en proyectos se realice en aquellos solicitados por organizaciones sin ánimo de lucro y en el marco de la propia universidad. En este sentido, se observa una tensión entre la formación académica y el desarrollo de software para la industria, lo que plantea un dilema que debe analizarse al abordar este tipo de proyectos en la universidad.

4. Conclusiones

Este trabajo presentó un análisis de las actividades de diseño y desarrollo de software de un grupo de docentes y estudiantes pertenecientes a un laboratorio de investigación, desarrollo e innovación en el área de la informática y computación aplicada en la Facultad de Ingeniería de la UNER. Se analizaron de manera integral

Revista EIA https://doi.org/10.24050/reia.v22i44.1886

doce proyectos de diseño y desarrollo de software realizados por el grupo, utilizando la información disponible de cada proyecto y las opiniones recopiladas mediante una encuesta a los miembros involucrados en cada proyecto.

Entre los aportes más relevantes de este trabajo se destaca la incorporación de ADRs, aunque aún en fase de implementación, constituyen una forma eficaz de documentación. La aplicación de los ADRs está más extendida en la industria del software que en la academia, por lo que es un avance significativo para la sistematización de decisiones en grupos de diseño y desarrollo de software en la universidad. A medida que esta práctica se consolide, se espera que los ADRs contribuyan significativamente en los futuros desarrollos del grupo.

Otra área de mejora identificada es la difusión de los resultados de los desarrollos del grupo. A nivel general, se identificó la necesidad de visibilizar más ampliamente los desarrollos realizados en la universidad. Para ello, se planteó la utilización de repositorios públicos e integrar con plataformas que permitan preservar y distribuir para dar mayor alcance a los proyectos. No obstante, para que estas estrategias sean sostenibles y efectivas, resulta fundamental avanzar en la definición de un marco normativo institucional que regule la producción de software universitario, estableciendo criterios claros sobre licencias, derechos de uso y políticas de distribución.

En relación a la vinculación del grupo con el entorno se destaca la experiencia con el desarrollo de UsiLac 4.0 como un modelo de colaboración público-privada para fomentar la innovación en pequeñas empresas sin área propia de I+D. Sin embargo, se advierte que este tipo de proyectos puede generar tensiones entre los objetivos de formación académica y los de desarrollo de software para la industria. En este sentido, la participación de estudiantes en proyectos solicitados por organizaciones sin ánimo de lucro o dentro de la universidad, permite equilibrar estos aspectos en tensión. Este tema representa un área que merece mayor profundización y estudio.

Este estudio presenta algunas limitaciones que deben ser consideradas al interpretar sus resultados. En primer lugar, el

cuestionario utilizado no fue sometido a una validación formal; si bien fue revisado internamente por miembros del grupo, su diseño se basó principalmente en instrumentos similares presentes en la literatura y en la experiencia acumulada del grupo en proyectos de desarrollo. En cuanto a los posibles sesgos, las respuestas a la encuesta fueron confidenciales, lo que puede haber influido en la forma de responder, aunque se garantizaron condiciones de anonimización para su procesamiento y publicación. Participaron tanto miembros actuales como ex miembros del grupo, lo que enriquece la mirada retrospectiva, pero también introduce cierta variabilidad en las experiencias relatadas. Asimismo, al tratarse de valoraciones subjetivas, no se implementaron mecanismos adicionales de mitigación del sesgo, lo cual se reconoce como una oportunidad de mejora para futuras investigaciones. Finalmente, si bien se busca alcanzar cierta generalidad en los aportes, es importante señalar que los resultados y reflexiones están fuertemente condicionados por la forma de trabajo del grupo y las particularidades de su contexto organizacional, institucional y cultural, como se ha descrito en la introducción.

Este ejercicio de análisis y reflexión sobre las prácticas de ingeniería de software del grupo representa un aporte más al conjunto de esfuerzos que le dan importancia y valor al software que se desarrolla en la universidad. Más allá de identificar áreas de mejora de un grupo en particular, este ejercicio contribuye al fortalecimiento del desarrollo de software en la universidad como un área clave de estudio y aplicación de la ingeniería de software. Las propuestas de mejoras en las prácticas de ingeniería de software, en la formación del grupo y en la difusión y sostenimiento del software producido en los grupos, fortalecen el papel de la universidad como un espacio de innovación y generación de conocimiento en ingeniería de software, que impacta tanto en el ámbito académico como en el entorno productivo y social.

5. Agradecimientos

Los autores agradecen a las y los participantes de la encuesta y compartieron generosamente sus experiencias y reflexiones, las cuales resultaron fundamentales para este trabajo. Asimismo,

desean expresar un reconocimiento a la Facultad de Ingeniería y a la Universidad Nacional de Entre Ríos por brindar el marco institucional y normativo necesario para la realización de Proyectos de Investigación y Desarrollo (PID), que hacen posible el fortalecimiento de estas iniciativas académicas y tecnológicas.

Financiamiento

El presente trabajo es un resultado del proyecto de investigación y desarrollo PID-UNER 6241 *Estudio del proceso de diseño y desarrollo de interfaces de usuario para aplicaciones de internet de las cosas*. Universidad Nacional de Entre Ríos.

Referencias

Booch, G. (1996). *Análisis y diseño orientado a objetos con aplicaciones* (J. M. Cueva Lovelle & A. Cernuda del Río, Trads., 2.a ed.). Pearson Educación.

Buchgeher, G.; Schöberl, S.; Geist, V.; Dorninger, B.; Haindl, P.; Weinreich, R. (2023). Using architecture decision records in open source projects—An MSR study on GitHub. *IEEE Access*, 11, 63725-63740. https://doi.org/10.1109/ACCESS.2023.3287654

Catalá-Pérez, D.; de-Miguel-Molina, M. (2018). La colaboración público-privada como instrumento de impulso a la innovación: Definición de un marco de análisis. *Reforma y Democracia*, 72, 43-86.

Consejo Directivo FIUNER. (2023a, marzo 28). *Reglamento General de los Núcleos de Extensión e Investigación, Desarrollo e Innovación (I+D+i) en el ámbito de la FI-UNER*. Res CD 023/23. https://digesto.uner.edu.ar/documento.frame.php?cod=138023

Consejo Directivo FIUNER. (2023b, agosto). *Reglamento para el funcionamiento de los departamentos académicos Facultad de Ingeniería—Universidad Nacional de Entre Ríos*. https://digesto.uner.edu.ar/documento.frame.php?cod=144215

De Carvalho, D.; Hussain, R.; Khan, A.; Khazeev, M.; Lee, J.; Masiagin, S.; Mazzara, M.; Mustafin, R.; Naumchev, A.; Rivera, V. (2020). Teaching programming and design-by-contract. En M. E. Auer; T. Tsiatsos (Eds.), *The Challenges of the Digital Transformation in Education* (Vol. 916, pp. 68-76). Springer International Publishing. https://doi.org/10.1007/978-3-030-11932-4_7

Diaz Zamboni, J. E. (2024). *Registros de decisiones arquitectónicas del LICA*. https://github.com/FIUNER-LICA/lica-decisiones-arquitectonicas

Ekedahl, U.; Mihailescu, R.-C.; Ma, Z. (2018). Lessons learned from adapting «things» to IoT platforms in research and teaching. *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 1457-1460. https://doi.org/10.1145/3167132.3167421

Estatuto de la Universidad Nacional de Entre Ríos, Argentina. (2023). https://digesto.uner.edu.ar/documento.frame.php?cod=139356

Fairbanks, G. (2023). Software architecture is a set of abstractions. *IEEE Software*, 40(4), 110-113. https://doi.org/10.1109/MS.2023.3269675

García, J.; Insfrán, J.; Diaz Zamboni, J. E. (2024). Aporte a la confiabilidad de programas C++ mediante una herramienta ad hoc para implementar contratos. *XXX Congreso Argentino de Ciencias de la Computación (CACIC 2024)*. https://sedici.unlp.edu.ar/handle/10915/176464

Godbole, R. (2024). Choosing the right open source framework: A comprehensive guide. *International Journal of Research in Computer Applications and Information Technology*, 7(2), 2340-2349. https://doi.org/10.34218/IJRCAIT_07_02_172

Guevara-Reyes, J.; Vinueza-Morales, M.; Ruano-Lara, E.; Vidal-Silva, C. (2025). Implementation of personalized frameworks in computational thinking development: Implications for teaching in software engineering. *Frontiers in Education*, 10, 1584040. https://doi.org/10.3389/feduc.2025.1584040

Insfrán, J. F.; Diaz Zamboni, J. E. (2023). Reflexiones sobre la enseñanza de confiabilidad y seguridad del software en bioingeniería con programación basada en contratos. *Jornadas Argentinas de Didáctica de las Ciencias de la Computación (JADiCC 2023)*.

Katz, D. S.; McHenry, K.; Reinking, C.; Haines, R. (2019). Research software development & management in universities: Case studies from Manchester's RSDS group, Illinois' NCSA, and Notre Dame's CRC. *2019 IEEE/ACM 14th International Workshop on Software Engineering for Science (SE4Science)*, 17-24. https://doi.org/10.1109/SE4Science.2019.00009

Khorikov, V. (2020). *Unit testing principles, practices, and patterns* (1st ed.). Manning Publications.

Kraemer, E. (2018). Teaching the design-by-contract concept in a software engineering course using RESOLVE. *ACM SIGSOFT Software Engineering Notes*, 43(3), 18-18. https://doi.org/10.1145/3229783.3229796

Martin, R. (2018). *Arquitectura limpia: Guía para especialistas en la estructura y el diseño de software*. Anaya Multimedia.

Meyer, B. (1992). Applying «design by contract». *Computer*, 25(10). https://doi.org/10.1109/2.161279

OpenAI. (2025). *ChatGPT* [Software]. https://chatgpt.com/

Pantoja Yépez, W. L.; Hurtado Alegría, J. A.; Bandi, A.; Kiwelekar, A. W. (2024). Training software architects suiting software industry needs: A literature review. *Education and Information Technologies*, 29(9), 10931-10994. https://doi.org/10.1007/s10639-023-12149-x

Polack-Wahl, J. A. (2006). Lessons learned from different types of projects in software engineering. *Frontiers in Education: Computer Science & Computer Engineering*. https://api.semanticscholar.org/CorpusID:33621074

Richards, M.; Ford, N. (2020). *Fundamentals of software architecture: An engineering approach* (1st ed.). O'Reilly Media, Inc.

Sobral, S. R. (2021). The old question: Which programming language should we choose to teach to program? En T. Antipova (Ed.), *Advances in Digital Science* (Vol. 1352, pp. 351-364). Springer International Publishing. https://doi.org/10.1007/978-3-030-71782-7_31

Sommerville, I. (2005). *Ingeniería del software* (M.-R. Miguel, Ed., 7.a ed.). Pearson Educación.

Washizaki, H. (Ed.). (2024). *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)* (Version 4.0). IEEE Computer Society.