UNIVERSIDAD & SOCIEDAD
EVALUACIÓN

Fecha de presentación: febrero, 2025
Fecha de aceptación: mayo, 2025
Fecha de publicación: julio, 2025

COMPARATIVA DE FRAMEWORKS BACKEND Y ADOPCIÓN EDUCATIVA DE ENTITY FRAMEWORK CORE

COMPARATIVE EVALUATION OF BACKEND FRAMEWORKS AND EDUCATIONAL ADOPTION OF ENTITY FRAMEWORK CORE

Luis Antonio Llerena Ocaña [1]*
E-mail: ua.luisllerena@uniandes.edu.ec
ORCID: https://orcid.org/0000-0001-6440-0167
Fausto Alberto Viscaino Naranjo [1]
E-mail: ua.faustoviscaino@uniandes.edu.ec
ORCID: https://orcid.org/0000-0003-1760-6992
Walter Vinicio Culque Topanta [1]
E-mail: ua.walterculque@uniandes.edu.ec
ORCID: https://orcid.org/0000-0002-3421-2306
[1]Universidad Regional Autónoma de Los Andes, Ambato. Ecuador.
*Autor para correspondencia

Cita sugerida (APA, séptima edición)

Llerena Ocaña, L. A., Viscaino Naranjo, F. A., & Culque Topanta, W. V. (2025). Evaluación comparativa de frameworks backend y adopción educativa de Entity Framework Core. Universidad y Sociedad, 17(14), e5301.

RESUMEN

El presente artículo trata los elementos esenciales que intervienen en la habilidad de comprender y las tipologías textuales en el proceso de enseñanza-aprendizaje de la Lengua castellana, fundamentados desde la teoría, así como desde los documentos normativos colombianos. Los cuales en conjunto y teniendo como base la Constitución Política de 1991, deben garantizar una educación incluyente, donde se debe tener en cuenta el entorno social y cultural de los individuos para poder desarrollar una práctica pedagógica significativa y contextualizada. A lo largo del escrito se relaciona el proceso de enseñanza-aprendizaje, con la formación de la habilidad comprender, desde las diversas tipologías textuales, como elemento dinamizador de dicho proceso en los estudiantes de tercer grado de la Educación Básica Primaria.

Palabras clave: Proceso de enseñanza-aprendizaje, Lengua castellana, habilidad comprender, tipologías textuales.

ABSTRACT

This article deals with the essential elements that intervene with the ability to understand, from the different text typologies in the teaching-learning process of the Castilian language, based on theory, as well as Colombian normative documents, which as a whole and based on the Political Constitution of 1991, must guarantee an inclusive education, where the social and cultural environment of the individuals must be taken into account in order to develop a meaningful and contextualized pedagogical practice. Throughout the writing, the teaching-learning process is related with the formation of the ability to understand, from different text typologies, as a dynamic element of this process in the third grade students of Basic Primary Education.

Keywords: Teaching-learning process, Castilian language, ability to understand, text typologies.

INTRODUCCIÓN

La amplia disponibilidad de Internet ha dado lugar a la popularidad de las aplicaciones de Internet, que facilitan el uso de diversos servicios directamente a través de un navegador web. Esto elimina la necesidad de instalar software adicional o consumir recursos de hardware del dispositivo. Las aplicaciones de Internet son programas de software alojados en servidores remotos, y los usuarios pueden acceder a ellos a través de una interfaz gráfica de usuario que se muestra en una ventana del navegador. La comunicación entre el navegador y los servicios web se facilita mediante interfaces de programación, denominadas comúnmente interfaces de programación de aplicaciones (API). Las aplicaciones web dependen del envío de peticiones y la recepción de respuestas mediante diversos protocolos, entre ellos el Protocolo de Transferencia de Hipertexto (HTTP). Con el tiempo, se han desarrollado numerosos lenguajes de programación para facilitar la creación de aplicaciones de Internet. Sin embargo, utilizar un lenguaje puro exige construir todas las funcionalidades necesarias desde cero. Esto plantea numerosos problemas y requiere mucho tiempo. Por eso, los desarrolladores suelen recurrir a soluciones preexistentes que ofrecen funciones de eficacia probada capaces de mejorar tanto el rendimiento como la seguridad (Choma et al., 2023)

Para mejorar la eficiencia del desarrollo de software, los desarrolladores utilizan marcos de trabajo (frameworks (FW)) que mejoran el rendimiento de las aplicaciones web y les proporcionan una mayor facilidad de uso. Un framework consiste en un conjunto de componentes que ofrecen diversas funciones y capacidades a los desarrolladores. Entre estas funciones se encuentran la gestión de bases de datos, la realización de operaciones en ellas y los mecanismos de autenticación y autorización que mejoran la seguridad de las aplicaciones (Choma et al., 2023)

En síntesis, un Framework es una estructura o modelo de trabajo que los programadores suelen utilizar para facilitar la creación de programas. Su uso acelera el proceso de desarrollo al evitar la duplicación de código, garantiza la implementación de buenas prácticas y mantiene la coherencia del código (Esquivel-Paula et al., 2025)

Un enfoque común es dividir las aplicaciones web en dos partes: el lado del cliente (frontend) y el lado del servidor (backend). El lado del cliente se encarga de enviar las peticiones al servidor, recibir y procesar las respuestas y presentar los datos al usuario en un formato adecuado. Por su parte, el servidor gestiona las peticiones, se comunica con la base de datos, procesa los datos y genera las respuestas. Esta separación permite el desarrollo paralelo de ambas partes de la aplicación, lo que puede reducir el tiempo total de desarrollo del software. A la hora de seleccionar un marco de trabajo, es importante tener en cuenta los requisitos específicos de un proyecto, ya que cada uno de ellos ofrece soluciones ligeramente distintas que pueden influir significativamente en parámetros como el rendimiento, la fiabilidad, la mantenibilidad y la portabilidad. Estos factores contribuyen a la calidad general del producto final (Choma et al., 2023).

Las frameworks de backend se han vuelto populares porque facilitan y agilizan el proceso de desarrollo de aplicaciones web. Proporcionan herramientas y bibliotecas que permiten a los desarrolladores centrarse en la lógica específica de la aplicación en lugar de lidiar con tareas repetitivas y complejas, como la gestión de conexiones de base de datos, la implementación de autenticación y la seguridad (Esquivel-Paula et al., 2025).

En la actualidad, las aplicaciones web han alcanzado un alto grado de prevalencia, lo que está asociado a la multitud de herramientas disponibles del lado del servidor utilizadas para su desarrollo. La selección de la tecnología más adecuada que satisfaga los requisitos de software es una de las principales decisiones que hay que tomar para crear una aplicación web plenamente funcional. Este proceso está estrechamente relacionado con la elección de un lenguaje de programación y un marco de trabajo adecuados. Existen numerosos artículos académicos que abordan el tema de la elección de herramientas óptimas para el desarrollo de aplicaciones web con determinados niveles de complejidad y ámbito de operaciones (Choma et al., 2023).

El desarrollo de aplicaciones en la actualidad debe tener en cuenta los grandes volúmenes de información, las especificaciones de software, la seguridad y la accesibilidad (interacción y comunicación). Bien sea una aplicación web o móvil, estas presentan ventajas y desventajas como la dependencia de internet para su funcionamiento, la rapidez, la eficiencia y los costos (Quintero et al., 2024) el diseño se convierte en un elemento central después de tener una visión clara de cómo sería la aplicación. La estructura de las aplicaciones móviles implica algo más que estética, sino también arquitectura de la información, usabilidad y experiencia de usuario (UX.

Elegir un framework de backend depende, en gran parte, de las características requeridas por la aplicación. De acuerdo con algunos autores las pequeñas y medianas empresas se encuentran en un proceso de transformación digital en el que crean sus propias aplicaciones sin una base sólida de conocimientos ni recursos suficientes para gestionarlos de manera adecuada, rentable y solvente, lo cual, puede producir un impacto en sus ingresos, eficiencia y capacidad de crecimiento (Quintero et al., 2024)el diseño se convierte en un elemento central después de tener una visión clara de cómo sería la

aplicación. La estructura de las aplicaciones móviles implica algo más que estética, sino también arquitectura de la información, usabilidad y experiencia de usuario (UX.

Existen múltiples fuentes sobre los criterios que son esenciales para la selección de FW. Una matriz de comparación de proveedores finales es la matriz de Raible adaptada a los proveedores finales. La tabla explica 20 criterios que son relevantes para la visión general de un FW en particular. Explica cada criterio de comparación, evalúa los criterios para cada FW y el método para evaluar los criterio (Kaluža et al., 2019).

El tiempo para el aprendizaje, las características técnicas, el compromiso de la comunidad y la consecuencia de la elección son las características más importantes a la hora de elegir una FW. Se considera que el tiempo de aprendizaje es un factor muy importante, independientemente del FW que se utilice, lo que significa que cada FW requiere una cierta cantidad de tiempo para el aprendizaje. Las características técnicas básicas que deben comprobarse antes de elegir un FW son: rendimiento, compatibilidad y si el FW es adecuado para el tipo de estructura de aplicación utilizada. Es importante el compromiso de la comunidad, es decir, los datos sobre el número de desarrolladores que utilizan un determinado FW y el número de debates relacionados con este asunto (Kaluža et al., 2019).

Un factor importante para la selección de tecnología es el TTM (Time to Market). Una de las características del TTM sugiere elegir tecnologías que tengan la capacidad de integrarse con soluciones de terceros para simplificar el proceso de integración (Kaluža et al., 2019).

Las aplicaciones web modernas se construyen cada vez más sobre frameworks backend que facilitan y aceleran el desarrollo. Estos entornos ofrecen componentes predefinidos que permiten a los desarrolladores enfocar esfuerzos en la lógica del negocio. Históricamente, en los 2000 surgen frameworks como Ruby on Rails en 2004 y Django en 2005 que popularizan el desarrollo rápido basado en el Modelo Vista Controlador (MVC). En la década de 2010 llegan Node.js/Expressy Spring Boot, enfatizando rendimiento y escalabilidad. El ecosistema .NET es otra variante importante que ha ido evolucionado. Lo que comienza como .NET Framework (2002) dio paso a .NET Core y al .NET unificado (2020) de código abierto. Ver tabla 1.

Tabla 1. Herramientas de back-end de código abierto

Herramienta	Lenguaje de Programación	Arquitectura	Sistema de plantillas	ORM	Caracte-rísticas destacadas
Node.js	JavaScript	Event-Driven, Non-Blocking I/O	Ninguno(soporta motores de plantillas como EJS, Pug)	Ninguno(Soporta ORM externos como Sequelize, TypeORM)	Manejo eficiente de conexiones concurrente, gran ecosistema de módulos y paquetes
Django	Python	MVT	Django template language(DTL)	Django ORM	Seguridad robusta, sistema de administración integrado, soporte para múltiples bases de datos
Laravel	PHP	MVC	Blade	Eloquent ORM	Sistema de migraciones de bases de datos, gestión de colas, sistema de autenticación fácil
Ruby on Rails	Ruby	MVC	ERB (Embedded Ruby)	ActiveRecord	Caonvención sobre configuración, sistema de migraciones, soporte AJAX

Herramienta	Lenguaje de Programación	Arquitectura	Sistema de plantillas	ORM	Caracte-rísticas destacadas
Spring Book	Java	MVC	Thymeleaf, JSP	JPA/Hibernate	Integración fácil con servicios de Spring, configuración automática
Express.js	JavaScript	MVC	Ninguno (soporta motores de plantillas como EJS, Pug)	Ninguno (soporta ORM externos como Mongoose)	Framework minimalista, altamente personalizable
Flask	Python	MVC	Jinja2	Ninguno (soporta ORM externos como SQLAlchemy)	Framework minimalista, extensible con módulos externos
Symfony	PHP	MVC	Twig	DoctrineORM	Componentes modulares reutilizables, sistema de bundles
ASP.NET Core	C#	MVC	Razor	Entity Framework	Integración con herramientasde Microsoft, sistema de autenticación y autorización

Fuente: tomando de Esquivel-Paula et al. (2025) .

En el ámbito del desarrollo de aplicaciones web, la eficiencia, escalabilidad y mantenibilidad del software son pilares fundamentales que dependen en gran medida de las herramientas y frameworks que se utilicen. En este contexto, el uso de Object-Relational Mapping (ORM) se ha convertido en una práctica común para facilitar el acceso y manipulación de datos desde el código fuente, reduciendo significativamente la complejidad del desarrollo(Llerena Ocaña et al., 2022).

Uno de los frameworks ORM más utilizados en el ecosistema .NET es Entity Framework Core (EF Core), el cual proporciona una forma robusta, extensible y multiplataforma para trabajar con bases de datos mediante modelos de objetos. EF Core permite aplicar buenas prácticas de ingeniería de software, como la separación de responsabilidades y la implementación de patrones de diseño como Repositorio o Unidad de Trabajo, promoviendo así arquitecturas limpias y modulares (Microsoft, 2024).

Sin embargo, EF Core no es la única solución disponible para el desarrollo de aplicaciones web con acceso a bases de datos. Frameworks como Django (Python), Laravel (PHP) y Spring Boot (Java) también ofrecen mecanismos ORM altamente eficientes y con comunidades de soporte bien establecidas. Por ejemplo, un estudio comparativo entre Laravel y Django realizado por (Espinosa-Hurtado, 2021a) evidencia que Django ofrece una mayor adaptabilidad y eficiencia, especialmente en proyectos con estructuras complejas, gracias a su arquitectura basada en MTV (Model-Template-View) y su sistema ORM nativo.

En entornos Java, Spring Data JPA se ha consolidado como una alternativa sólida para el mapeo objeto-relacional. Según Mejia et al. (2015), los desarrolladores que trabajan con este framework destacan su versatilidad y capacidad de integración con servicios RESTful, además de sus mecanismos avanzados para la consulta dinámica y la validación de datos. Asimismo, otros estudios han comparado diversos frameworks ORM, incluyendo Hibernate, EF Core y Dapper, destacando que, aunque EF Core presenta una ligera curva de aprendizaje, su integración con Visual Studio y su estrecha relación con .NET lo hacen especialmente útil para desarrolladores de ese ecosistema.

En resumen, se los frameworks reducen el tiempo de desarrollo mediante plantillas y código reutilizable, estandarizan arquitecturas con patrones comunes y ofrecen seguridad integrada con protección contra XSS/CSRF. Esto beneficia a los desarrolladores con menor carga de trabajo repetitivo, a las organizaciones con aplicaciones mantenibles y rápidas de desplegar y a los usuarios finales con sistemas más fiables, rápidos y seguros.

Por otro lado, en un contexto educativo como el de la asignatura Aplicaciones Web, impartida en el sexto nivel de la carrera de Ingeniería de Software en Ecuador, resulta pertinente evaluar la percepción del estudiantado respecto al uso de EF Core y su impacto en el proceso de aprendizaje y desarrollo, cuestión ésta definida como parte de los objetivos investigativos de este trabajo. Igualmente, en esta investigación se compara los frameworks Django, Spring, Express, Laravel, Ruby on Rails, ASP.NET incluyendo EF Core usando un análisis multicriterio en rendimiento, escalabilidad, curva de aprendizaje, comunidad y costos en búsqueda identificar sus diferencias, innovaciones e implicaciones socio-técnicas.

MATERIALES Y MÉTODOS

La investigación siguió un enfoque mixto cualitativo-cuantitativo. Se revisaron estudios comparativos previos y se recolectaron datos secundarios, benchmarks de rendimiento, reportes comunitarios y encuestas, además de documentación oficial de cada framework. En particular, se consideró el test “Fortunes” de TechEmpower Round 22 para comparar el rendimiento bruto de cada tecnología. Se definieron criterios como rendimiento (throughput/latencia), escalabilidad o sea capacidad de crecer en carga, curva de aprendizaje, comunidad teniendo en cuenta el tamaño y la cantidad de actividades y costos en licencias y recursos. Cada framework se evaluó mediante datos objetivos o cualitativos con la opinión de expertos, y la revisión exahustiva de la documentación según estos criterios.

Adicionalmente se utilizó el método de encuesta estructurada, implementando un cuestionario con 10 ítems basados en la escala de Likert, lo que permitió medir cuantitativamente el grado de acuerdo o desacuerdo de los estudiantes frente a diversas afirmaciones relacionadas con el uso de EF Core.

La estrategia metodológica utilizada se fundamentó en el siguiente proceso: 1. Diseño del instrumento: Se elaboró un cuestionario con afirmaciones relacionadas a la comprensión, uso y percepción de EF Core, con base en los contenidos impartidos en clase. 2. Aplicación del instrumento: El cuestionario fue administrado a los estudiantes del sexto nivel de Ingeniería de Software mediante un formulario digital, garantizando el anonimato y la voluntariedad en la participación. 3. Análisis de resultados: Los datos recolectados fueron codificados y analizados utilizando medidas estadísticas básicas (media, moda, desviación estándar), con el fin de identificar tendencias generales en la percepción estudiantil. 4. Interpretación pedagógica: Los resultados permitieron retroalimentar el proceso de enseñanza-aprendizaje de la asignatura, identificando fortalezas y áreas de mejora en la enseñanza de herramientas ORM como EF Core.

RESULTADOS-DISCUSIÓN

Los resultados reflejan la percepción estudiantil frente al uso de Entity Framework Core (EF Core) como herramienta de desarrollo eficiente en el contexto académico. El cuestionario fue valorado mediante una escala Likert, siendo 1 “totalmente en desacuerdo” y 10 “totalmente de acuerdo”. Ver tabla 2.

Tabla 2. Percepción estudiantil.

Pregunta	Promedio	Máximo	Mínimo	Desviación Estándar	Varianza	Moda	Mediana	Rango	Coef. Variación (%)
Comprendo el propósito principal de EF Core en el desarrollo de aplicaciones web.	3.65	5	1	1.17	1.37	4.0	4.0	4	32.05
EF Core Tools (como Migrations o Scaffolding) me han facilitado la generación y mantenimiento de bases de datos.	3.82	5	1	1.19	1.4	5.0	4.0	4	31.15
El uso de EF Core ha contribuido a reducir significativamente el tiempo de desarrollo de mis proyectos.	3.59	5	1	1.28	1.63	4.0	4.0	4	35.65

Pregunta	Promedio	Máximo	Mínimo	Desviación Estándar	Varianza	Moda	Mediana	Rango	Coef. Variación (%)
EF Core me ha permitido aplicar con mayor facilidad arquitecturas como MVC o Clean Architecture.	3.53	5	1	1.18	1.39	3.0	4.0	4	33.43
El uso de patrones como Repositorio o Unidad de Trabajo se vuelve más claro al trabajar con EF Core.	3.47	5	1	1.01	1.01	4.0	4.0	4	29.11
Considero que EF Core mejora la legibilidad y mantenimiento del código en aplicaciones web.	3.59	5	1	1.18	1.38	4.0	4.0	4	32.87
La curva de aprendizaje de EF Core es adecuada.	3.47	5	1	0.94	0.89	3.0	3.0	4	27.09
Me siento capacitado para integrar EF Core en un proyecto web completo usando buenas prácticas de arquitectura.	3.47	5	1	0.94	0.89	3.0	3.0	4	27.09
EF Core ha fortalecido mi comprensión sobre la abstracción y separación de responsabilidades en el desarrollo web	3.47	5	1	1.07	1.14	4.0	4.0	4	30.84

Fuente: elaboración propia.

1. Tendencias generales (Promedio y Mediana)

Los promedios oscilan entre 3.47 y 3.82, lo cual indica una percepción moderadamente baja respecto al dominio y la aplicación de EF Core entre los estudiantes. La mediana, por su parte, se mantiene estable en 4.0 para casi todas las preguntas, lo que reafirma que la mayoría de respuestas se concentran hacia la mitad inferior de la escala. Esto podría reflejar una fase inicial de apropiación de conocimientos sobre esta herramienta.

2. Moda y consenso de respuestas

En la mayoría de ítems, la moda se sitúa entre 4 y 5, mostrando que esas fueron las respuestas más frecuentes. La excepción es una pregunta donde la moda fue 3, lo que revela que ciertos estudiantes aún tienen dificultades con la aplicación de arquitecturas complejas como Clean Architecture utilizando EF Core.

3. Dispersión de respuestas (Desviación estándar, Varianza y Coeficiente de variación)

Las desviaciones estándar varían entre 1.01 y 1.28, lo que representa una moderada dispersión en las opiniones de los estudiantes. Esta variabilidad se confirma con los valores de varianza, que oscilan entre 1.01 y 1.63.

El coeficiente de variación (CV) que expresa la dispersión relativa se ubica entre 29.11% y 35.65%, lo cual indica que hay una heterogeneidad significativa en el grado de acuerdo de los estudiantes sobre la utilidad y comprensión de EF Core. Esta diversidad sugiere la necesidad de reforzar el acompañamiento pedagógico y proporcionar más práctica guiada en torno a esta tecnología.

4. Rango y amplitud de respuestas

En todos los ítems, el rango fue de 4 puntos, lo que implica que hubo estudiantes que seleccionaron tanto valores bajos (1 o 2) como altos (5), reflejando diferencias claras en la comprensión individual del tema.

La percepción general sobre EF Core es moderada, con una tendencia a reconocer su valor práctico, pero con limitaciones evidentes en el dominio técnico y conceptual por parte de varios estudiantes. Este hallazgo evidencia que, si bien se está introduciendo el uso de herramientas ORM modernas, aún es necesario fortalecer el desarrollo de habilidades técnicas, especialmente en el uso de EF Core dentro de arquitecturas estructuradas como MVC o Clean Architecture.

Los resultados del análisis multicriterio se resumen en la tabla 3.

Tabla 3. Resumen del análisis multicriterio.

Criterio/ Framework	Django	Spring	Express (Node.js)	Laravel	Ruby on Rails	ASP.NET (Core+EF)
Rendimiento (Fortunes, TechEmpower)	2.0	3.3	4.6	1.0	1.9	19.6 (máximo)
Escalabilidad	Alta	Alta	Media (single-thread)	Media	Limitada	Alta
Curva de aprendizaje	Baja (fácil)	Alta (empinada)	Media-baja	Alta (empinada)	Media-baja	Alta (C# complejo)
Comunidad	Muy grande (Python)	Muy grande (Java)	Muy grande (JavaScript)	Grande (PHP)	Moderada (Ruby)	Muy grande (Microsoft)
Licencia / Costos	BSD, libre	Apache, libre	MIT, libre	MIT, libre	MIT, libre	MIT/Propietario (Core libre)

Fuente: Elaboración propia.

Rendimiento: Las pruebas sintéticas indican que ASP.NET Core lidera ampliamente el rendimiento con ventaja clave para altas cargas. Le siguen Express/Node, Spring Boot y Django, mientras que Rails y sobre todo Laravel resultan más lentos. Este orden refleja la eficiencia de entornos compilados o muy optimizados frente a lenguajes interpretados. Sin embargo, estos valores provienen de casos muy simples; en aplicaciones reales el rendimiento puede atenuarse mediante caché, balanceo y optimizaciones específicas.

Escalabilidad: Frameworks basados en hilos múltiples (Spring, .NET) suelen escalar bien horizontalmente, aprovechando hardware potente. Django también es considerado altamente escalable para grandes proyectos. En contraste, entornos mono-proceso como Express y, en menor medida, Laravel y Rails requieren arquitecturas adicionales (clusterización, microservicios) para alcanzar escalas similares. Rails en particular presenta limitaciones de escalabilidad en aplicaciones masivas.

Curva de aprendizaje: La facilidad de adopción varía. Estudios reportan que Django es fácil de aprender incluso para principiantes con una curva de aprendizaje casi plana, mientras que Laravel y Spring tienen curvas empinadas (especialmente Spring por la complejidad de Java). Express es relativamente accesible para desarrolladores JavaScript, y Rails suele ser rápido de dominar debido a su filosofía "convención sobre configuración". En general, frameworks con sintaxis declarativa o DSL pueden acelerar el onboarding, mientras que entornos más verbosos requieren más experiencia inicial.

Comunidad: Todos estos frameworks cuentan con comunidades muy activas. JavaScript/Node y Java/Spring lideran en número de desarrolladores globales, seguidas por Python/Django y PHP/Laravel. Rails mantiene una comunidad apasionada, aunque menor. En cualquier caso, abundan foros, tutoriales y paquetes de terceros, lo que mitiga riesgos de tecnología emergente.

Costos: Las soluciones analizadas son open source sin licencias comerciales directas. Django, Laravel y Rails se distribuyen bajo licencias permisivas (BSD o MIT). .NET Core/EF también es libre (con la posible excepción de herramientas opcionales). No hay costos de licencia de software, pero sí se deben considerar gastos en formación, soporte o infraestructura. Como señalan algunos estudios, al ser gratuitos estos frameworks "minimiza el coste de desarrollo".

DISCUSIÓN

El análisis muestra ventajas y desventajas claras de cada opción. En términos de productividad, frameworks con convenciones fuertes (Rails, Laravel, Django) permiten prototipar más rápido. Por ejemplo, Eloquent ORM en Laravel y el ORM de Rails agilizan el acceso a datos sin escribir SQL, mientras que Entity Framework Core brinda a .NET un ORM "ligero y de alta productividad". Esto beneficia directamente a los desarrolladores (menor código repetitivo) y reduce tiempos de entrega. Para organizaciones, esto significa productos al mercado más rápido, aunque con la consideración de que estos entornos suelen demandar más recursos de ejecución.

Por otro lado, al priorizar rendimiento y escalabilidad, .NET Core y Spring Boot sobresalen. Son adecuados en dominios de alta concurrencia (finanzas, IoT, trading), donde cada milisegundo y la capacidad de servir miles de solicitudes simultáneas importan. Sin embargo, su complejidad puede aumentar los costos de aprendizaje y desarrollo.

La comprensión del propósito de Entity Framework Core (EF Core) es muy importante para su correcta implementación en proyectos de desarrollo web. Según un estudio de Myllyaho (2022), EF Core facilita la interacción con bases de datos al permitir a los desarrolladores trabajar con objetos .NET, reduciendo la complejidad del código y mejorando la mantenibilidad del software. Sin embargo, la falta de formación adecuada puede limitar su aprovechamiento óptimo.

Las herramientas de EF Core, como Migrations y Scaffolding, son muy importantes para gestionar cambios en el esquema de la base de datos de manera coherente con el modelo de la aplicación. Según,Microsoft (2024) estas herramientas permiten a los desarrolladores mantener la sincronización entre el modelo de datos y la base de datos, facilitando el desarrollo ágil. No obstante, su uso efectivo requiere una comprensión profunda de su funcionamiento.

La adopción de EF Core puede influir en la eficiencia del desarrollo. Un estudio comparativo entre EF Core y Dapper realizado por Trailhead Technology (2023) indica que, aunque EF Core puede ser más lento en ciertas operaciones, su capacidad para manejar tareas complejas y reducir la cantidad de código escrito puede compensar esta desventaja en términos de productividad.

La integración de EF Core en arquitecturas como Modelo-Vista-Controlador (MVC) o Clean Architecture facilita la separación de preocupaciones y mejora la mantenibilidad del código. Según un artículo de Hutton (Hutton, 2022) EF Core se adapta bien a estas arquitecturas, permitiendo una estructura más modular y escalable.

La implementación de patrones de diseño como Repositorio y Unidad de Trabajo en conjunto con EF Core puede mejorar la gestión de datos y la organización del código. Un artículo de Teoman (2021) destaca que, aunque EF Core implementa internamente estos patrones, su uso explícito puede ofrecer beneficios adicionales en términos de abstracción y testabilidad.

La legibilidad y mantenibilidad del código son aspectos críticos en el desarrollo de software. Según un artículo de Jones (2015), EF Core permite a los desarrolladores trabajar con modelos de datos más cercanos al dominio del negocio, lo que mejora la comprensión y facilita el mantenimiento del código.

La curva de aprendizaje de EF Core puede variar según la experiencia previa de los estudiantes. Un estudio de Amershi et al. (2019) sugiere que la integración de herramientas modernas como EF Core en el currículo académico puede ser beneficiosa, pero requiere una planificación cuidadosa para asegurar una transición efectiva.

La capacidad de integrar EF Core en proyectos completos es indicativa de una formación sólida. Según un artículo de Quezada-Sarmiento et al. (2020), la educación en ingeniería de software debe enfocarse en proporcionar experiencias prácticas que permitan a los estudiantes aplicar herramientas como EF Core en contextos reales (Llerena et al., 2022).

La abstracción y la separación de responsabilidades son principios fundamentales en la ingeniería de software moderna. En el contexto educativo, su comprensión se ve fortalecida cuando los estudiantes trabajan con tecnologías que lo exigen explícitamente. EF Core permite estructurar el código utilizando patrones como Repositorio y Unidad de Trabajo, que promueven dicha separación, lo cual ha sido reconocido en investigaciones sobre diseño de software en entornos académicos. Otros autores destacan que la incorporación de frameworks modernos en proyectos estudiantiles mejora la comprensión de la arquitectura multicapa y su aplicación en contextos reales. Se ha encontrado además que el uso de patrones de diseño junto con ORM en proyectos universitarios incrementa significativamente la capacidad de los estudiantes para identificar y separar las responsabilidades de cada capa del sistema.

Recomendar una herramienta como EF Core implica reconocer su eficacia en escenarios reales. La percepción de los estudiantes al respecto está alineada con estudios como el de Espinosa-Hurtado (2021b) quien concluye que los frameworks ORM modernos como EF Core y Django ofrecen ventajas claras en términos de rendimiento y facilidad de uso, especialmente para quienes se están formando en entornos universitarios. Igualmente se han identificado que el uso de ORM modernos reduce el tiempo de desarrollo y mejora la mantenibilidad, dos características clave para proyectos estudiantiles. Finalmente, la evaluación comparativa de frameworks realizada por Mejia et al. (2015) confirma que la eficiencia de EF Core en la manipulación de datos lo convierte en una alternativa recomendable frente a otras soluciones como ADO.NET o Dapper, dependiendo del nivel de experiencia del usuario.

CONCLUSIONES

Al concluir el trabajo se puede afirmar que cada framework ofrece ventajas específicas, ASP.NET Core (+EF Core) sobresale en rendimiento extremo; Django y Rails facilitan el aprendizaje rápido; Express y Spring Boot brindan equilibrio entre velocidad y escalabilidad.

La elección de un framework debe basarse en criterios de rendimiento, curva de aprendizaje, comunidad y costes, alineados con los objetivos y recursos del proyecto. Incorporar en el currículo proyectos colaborativos que simulen entornos de producción permitirá a los estudiantes experimentar las exigencias de escalabilidad y mantenibilidad desde etapas tempranas.

El uso de Entity Framework Core en contextos académicos permite a los estudiantes aplicar de forma práctica conceptos como la abstracción, la separación de responsabilidades y la implementación de patrones de diseño como Repositorio y Unidad de Trabajo. Esto se traduce en una mejor comprensión de arquitecturas modernas como MVC y Clean Architecture.

Los resultados muestran que, aunque los estudiantes presentan un nivel medio de dominio sobre las herramientas de EF Core, reconocen su utilidad en términos de eficiencia y organización del código. Esto sugiere que el uso de EF Core como herramienta formativa es adecuado, siempre que se complemente con metodologías activas de aprendizaje y proyectos prácticos.

En comparación con otros frameworks ORM, EF Core se posiciona como una solución robusta y adecuada para el desarrollo de aplicaciones web dentro del entorno educativo. Su integración con .NET y su compatibilidad con buenas prácticas de desarrollo hacen de EF Core una herramienta altamente recomendable para formar futuros ingenieros de software con criterios sólidos de eficiencia, mantenibilidad y escalabilidad.

Aunque los estudiantes reconocen su utilidad, la puntuación media alrededor de 3.5 sobre 5 y la alta dispersión indican brechas en su dominio técnico y conceptual. Es muy importante acompañar la teoría con prácticas intensivas y estudios de caso para asegurar la apropiación de EF Core y los patrones de diseño asociados.

REFERENCIAS BIBLIOGRÁFICAS

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). ***Software Engineering for Machine Learning: A Case Study***. Microsoft. https://www.microsoft.com/en-us/research/wp-content/uploads/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf

Choma, D., Chwaleba, K., & Dzieńkowski, M. (2023). THE EFFICIENCY AND RELIABILITY OF BACKEND TECHNOLOGIES: EXPRESS, DJANGO, AND SPRING BOOT. ***Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska***, ***13***(4), Article 4. https://doi.org/10.35784/iapgos.4279

Espinosa-Hurtado, R. (2021a). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. ***CEDAMAZ***, ***11***(2), Article 2.

https://doi.org/10.54753/cedamaz.v11i2.1182

Espinosa-Hurtado, R. (2021b). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. ***CEDAMAZ***, ***11***(2), Article 2.

https://doi.org/10.54753/cedamaz.v11i2.1182

Esquivel-Paula, G., Quisaguano-Collaguazo, L., Caluña-Guaman, A., & Llambo-Alvarez, S. (2025). Frameworks del lado del Servidor: Caso de Estudio Node JS, Django y Laravel. ***593 Digital Publisher CEIT***, ***10***(1), Article 1. https://doi.org/10.33386/593dp.2025.1.2729

Hutton, R. (2022). Entity Framework vs Repository Pattern vs Unit of Work. ***Medium***. https://medium.com/@robhutton8/entity-framework-vs-repository-pattern-vs-unit-of-work-9fa093bd59e4

Jones, M. (2015). ***Dapper vs Entity Framework vs ADO.NET Performance Benchmarking***. https://exceptionnotfound.net/dapper-vs-entity-framework-vs-ado-net-performance-benchmarking/

Kaluža, M., Kalanj, M., & Vukelić, B. (2019). A COMPARISON OF BACK-END FRAMEWORKS FOR WEB APPLICATION DEVELOPMENT. ***Zbornik Veleučilišta u Rijeci***, ***7***(1), 317-332. https://doi.org/10.31784/zvr.7.1.10

Llerena Ocaña, L. A., Viscaino Naranjo, F. A., Culque Toapanta, W. V., Llerena Ocaña, L. A., Viscaino Naranjo, F. A., & Culque Toapanta, W. V. (2022). Desarrollo de software con Net Core. ***Revista Universidad y Sociedad***, ***14***(2), 85-89. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2218-36202022000200085&lng=es&nrm=iso&tlng=es

Mejia, H., Tuesta-Monteza, V., & Sánchez Acosta, C. (2015). Análisis comparativo de frameworks para el desarrollo de aplicaciones web en java. ***INGENIERÍA Ciencia Tecnología e Innovación***, ***2***(1).

https://www.researchgate.net/publication/343614953_Analisis_comparativo_de_frameworks_para_el_desarrollo_de_aplicaciones_web_en_java

Microsoft. (2024). ***Introduction to Performance—EF Core***. https://learn.microsoft.com/en-us/ef/core/performance/

Myllyaho Forsberg, M. (2022). ***An evaluation of .NET Object-Relational Mappers in relational databases Entity Framework Core and Dapper***. UMEA University. Department of Computing Science.

https://www.diva-portal.org/smash/get/diva2:1687513/FULLTEXT01.pdf

Quezada-Sarmiento, P. A., Elorriaga, J. A., Arruarte, A., & Washizaki, H. (2020). Open BOK on Software Engineering Educational Context: A Systematic Literature Review. ***Sustainability***, ***12***(17), 6858. https://doi.org/10.3390/su12176858

Quintero, L. M. P., Barona, D. J. R., León, K., & Torres, F. C. (2024). Desarrollo de un aplicativo móvil con Node.js para la venta de productos agrícolas en MiPymes. ***Revista Temario Científico***, ***4***(2), Article 2. https://doi.org/10.47212/rtcAlinin.2.224.3

Teoman, E. (2021, abril 6). Combining repository pattern and unit of work using EntityFramework Core. ***Borda Technology***. https://medium.com/borda-technology/combining-repository-pattern-and-unit-of-work-using-entityframework-core-db5c2d9a19cd