UDC 004.4`2
DOI https://doi.org/10.32782/2663-5941/2025.1.2/33

Semeniuk V.V.
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

OPTIMIZATION OF LOCAL DEVELOPMENT PROCESS USING DOCKER PHP IMAGE THAT COMES WITH A FULL SET OF TOOLS OUT OF THE BOX: DATABASE AND INTERNATIONALIZATION EXTENSIONS

The article examines the integration of critical PHP extensions within the Docker PHP 7.1 image, highlighting their role in creating a cohesive and efficient development environment. MySQLi, with its support for both procedural and object-oriented programming, enables seamless database interactions while incorporating secure connections and debugging capabilities. It proves essential for managing relational data structures and performing complex database operations. Similarly, PDO enhances flexibility by standardizing interactions across different database systems, facilitating smooth transitions between platforms. Its prepared statement mechanism safeguards against SQL injection, and its abstraction layer supports scalability without compromising functionality. SOAP is explored as a bridge for XML-based communication, enabling integration of external APIs and transforming PHP applications into web services capable of managing diverse data types and complex message structures. The extension's role as an intermediary in synchronized data exchange between independent systems is emphasized. Aerospike is presented as a robust solution for NoSQL database management, excelling in scenarios requiring low-latency processing of large-scale data. Its distributed architecture ensures data redundancy and fault tolerance, supported by advanced replication techniques that optimize performance under high loads. Intl facilitates locale-sensitive operations, automating the formatting of dates, numbers, and messages. It simplifies the development of multilingual applications by removing manual localization complexities and enhancing the user experience through dynamic and culturally relevant data presentation. The Docker PHP 7.1 image unifies these extensions into a pre-configured containerized system, enabling developers to replicate production-like environments locally with consistent behavior across development, testing, and deployment phases. This integration eliminates dependency conflicts, leveraging Docker's volume mapping and networking features to seamlessly connect database containers with the PHP container. As a result, real-time database queries, SOAP interactions, and data formatting are efficiently managed within a streamlined workflow. This setup accelerates iterative cycles, reduces context switching, and ensures an optimized development process for scalable and globally accessible applications.

Key words: *containerization, localization, scalability, interoperability, optimization, adaptability.*

Formulation of the problem. The local development process is a critical phase in software creation, allowing developers to design, test, and iterate applications before deployment. However, common challenges include mismatched environments between development and production, manual dependency configurations, and difficulties integrating essential tools. These issues lead to inefficiencies, increased error rates, and delays in the development cycle.

Optimizing the local development process has become a necessity, particularly in environments requiring rapid iteration and high reliability. The adoption of containerized solutions like Docker offers a promising approach to addressing these challenges. A PHP Docker image equipped with a comprehensive set of tools, including database and internationalization extensions, significantly simplifies local development. By consolidating critical components such as MySQLi, PDO, SOAP, and Intl extensions into a pre-configured container, developers gain instant access to a stable and scalable environment that mirrors production settings.

This approach addresses the difficulties of manual configuration, ensuring seamless database connectivity, secure handling of structured and unstructured data, and culturally relevant localization. Additionally, incorporating advanced debugging tools and pre-configured libraries facilitates a smoother and more efficient development workflow. By locally replicating production-like conditions, developers can focus on building robust and scalable applications, ensuring consistency across all stages of the development lifecycle.

Analysis of recent research and publications. The article introduces a web-based Docker Image Assistant Generation (DIAG) tool designed to optimize the creation of Docker images in the User-PC Computing (UPC) system. Based on a master-worker model, the UPC system utilizes Docker technology to encapsulate application environments into container images. DIAG addresses manual Docker image creation inefficiencies by integrating Angular for dynamic user interfaces, Laravel for server-side logic and data processing through RestAPI, MySQL for structured data storage, and Shell scripts for automating image creation and upload processes.

A key feature of the tool is its ability to modify working code. This capability enables users to alter source code during tasks and update the Docker image directly on worker nodes, reducing transmission loads and enhancing system efficiency. The tool's implementation employs the MVC design pattern, ensuring problem separation while promoting scalability and maintainability. Evaluations included generating Docker images from reverse-engineered Dockerfiles sourced from GitHub, modifying source code for network modelling, and conducting performance tests to assess the tool's task execution across various environments.

The results validated the tool's ability to create and efficiently manage Docker images while supporting real-time updates, emphasizing its value in improving productivity and usability for users with varying experience levels. Future work aims to expand the tool's capabilities and further integrate it into the UPC system for broader applications and enhanced productivity.

In [2], a methodology for installing PHP extensions in Docker images without using PECL, which has been disabled by default since PHP 7.4, is presented. It provides detailed instructions for manually installing extensions like APCu, Redis, Igbinary, and MongoDB using Docker commands and scripts. By leveraging tools like docker-php-ext-configure and docker-php-ext-install, developers can create simplified, modular Dockerfiles tailored to the unique needs of their projects.

The study highlights common challenges associated with manual extension installation, particularly for extensions like MongoDB, which require submodules and multi-stage builds. Practical examples demonstrate how to create a reliable and efficient Docker environment for PHP development, focusing on adaptability, performance, and modular design.

Additionally, notable contributions from scholars such as J. Huang, J. Zhang, J. Liu, C. Li, R. Dai [4], L. Moroz [5], S. Neef, L. Kleissner [6], T. Sanoop [8], J. Watkins [11], J. Zhao, Y. Lu, K. Zhu, Z. Chen, H. Huang [12], M. Laaziri, K. Benmoussa, S. Khoulji, L. Kerkeb [13] and others are acknowledged.

Despite the above-mentioned documentation, the question of optimizing the local development process using the Docker PHP image remains underexplored and requires further study.

Task statement. The aims of this work are to integrate PHP extensions for database management and internationalization within the context of web application development using a Docker image.

Outline of the main material of the study. The MySQLi (MySQL Improved) extension is a key component of the Docker image, designed to facilitate connections to MySQL databases. This extension supports both procedural and object-oriented programming, offering a flexible interface for developers with built-in support for secure connections and debugging capabilities.

MySQLi's adaptability makes it a critical component for managing relational data structures and performing complex database operations. For instance, developers can establish a connection to a database server by providing the hostname, correct username, password, and desired database name. Once connected, structured queries can be executed to retrieve data from the database. Retrieved data can then be iterated over, with each row of the result set processed and displayed to the end user. Upon completing these operations, the database connection is closed. Similarly, in the object-oriented paradigm, the MySQLi class performs the same functions but provides a more intuitive interface through class methods and properties. This approach simplifies error handling by incorporating built-in error reporting mechanisms.

The PDO (PHP Data Objects) extension for MySQL and PostgreSQL offers a convenient method for accessing databases. The flexibility of PDO lies in its ability to standardize interactions with various databases, allowing developers to maintain a single code structure while seamlessly transitioning between different database platforms. For instance, PDO creates a uniform interface that eliminates the need to rewrite large portions of code when switching from one relational database to another. By utilizing prepared statements, PDO enhances security by preventing SQL injection attacks. Furthermore, its error-handling capabilities directly support debugging by providing detailed exception messages that pinpoint the source of problems. PDO's abstraction layer promotes scalability, enabling applications to adapt to varying database requirements without compromising performance or functionality.

SOAP simplifies interactions in web services by implementing client-server communication. This tool acts as an indispensable link in integrating external APIs or presenting PHP applications as web services, supporting a wide range of data types and complex message structures. SOAP enables data communication between different systems, essential in environments where multiple services or applications operate independently but require synchronized data exchange. For example, when a SOAP client sends a request to a remote service, the service interprets the request, processes the necessary data, and responds in a structured format.

Aerospike offers flexible support for NoSQL databases to handle large-scale data processing with minimal latency. This extension facilitates seamless connections and operations within PHP applications. Its high throughput and low response times make it ideal for applications requiring real-time data processing, such as financial transactions and user analytics. By employing a distributed architecture, Aerospike effectively ensures data persistence and high availability, particularly under increased workloads and high traffic. This architecture distributes data across multiple nodes, ensuring redundancy and fault tolerance. When a client application interacts with an Aerospike database, the extension efficiently manages data retrieval and storage across nodes, minimizing latency.

Additionally, Aerospike utilizes advanced data replication techniques, synchronous and asynchronous. In synchronous replication, every write operation is immediately propagated to all nodes, ensuring that data remains consistently available in real time. Conversely, asynchronous replication introduces slight delays in data propagation, which can be advantageous in scenarios demanding high throughput and minimal latency. Aerospike also employs quorum-based techniques to determine the minimum number of nodes required to acknowledge a write operation before it is deemed successful. This database is particularly effective for applications demanding fast read-and-write operations, such as in financial services and e-commerce.

The Intl extension provides tools for locale-sensitive operations, automating the formatting of dates, numbers, and messages. Simplifying the development of applications eases the management of multilingual content and removes the challenges of manual localization. For instance, when date values or currency need to be represented for users in different regions, the extension automatically adapts the format based on locale, ensuring an intuitive display of data to users. Additionally, its message translation functionality allows developers to implement dynamic, content-aware translations, improving the overall user experience.

By leveraging the Docker PHP 7.1 image, they described database and internationalization-oriented extensions are consolidated into a unified development experience. This integration highlights the synergy between these components and the Docker environment, optimizing the local development process and improving application design.

The combination of MySQLi and PDO within the Docker PHP image provides flexible support for database connectivity and query execution. The synergy between MySQLi's efficient connection handling for MySQL databases and PDO's standardized interface for multiple database systems allows developers to perform database-oriented operations with minimal configuration and maximum flexibility. For example, the procedural and object-oriented styles inherent to MySQLi, combined with PDO methods in a single environment, enable developers to test and compare approaches.

In Docker, the pre-configured environment ensures that both extensions have access to the same database instance (table 1).

This combined approach ensures that database operations can be performed and debugged in a unified environment, reducing the risk of configuration mismatches.

The SOAP extension, which facilitates XML-based communication, and Aerospike, which provides scalability for unstructured data, complement each other within the Docker image. These extensions enable the creation of hybrid applications capable of handling both structured and unstructured data streams. When a web service sends XML data, the response can be parsed into an Aerospike database, allowing the development environment to smoothly transition between managing real-time service interactions and high-performance data retrieval tasks (table 2).

The Intl extension, essential for applications targeting multilingual and multi-regional audiences, integrates seamlessly within the Docker image to format data based on specific locales, significantly enhancing accessibility. When used in tandem with SOAP services, Intl provides developers with tools to dynamically display culturally relevant information. For example, currency data obtained from a database or web service can be formatted for different regions without requiring manual adjustments.

The Docker PHP image provides a containerized environment where all described extensions function

Table 1

Example of MySQLi and PDO Integration in Docker PHP for Database Operations

```
$connection = new mysqli («db», «user», «password», «database»);
if ($connection->connect_error) {
die («Connection failed:». $connection->connect_error);
}
$pdo = new PDO («mysql: host=db; dbname=database», «user», «password»);
$stmt = $pdo->prepare («SELECT · FROM users»);
$stmt->execute();
while ($row = $stmt->fetch (PDO:: FETCH_ASSOC)) {
echo $row['name'] . "<br>»;
}
$connection->close();
```

Note: author's development

Table 2

Integration of SOAP, Aerospike, and Intl Extensions in Docker PHP for Hybrid Application Development

```
$client = new SoapClient («http: // example. com / api? wsdl»)
$response = $client->getData()
$config = [«hosts» => [[«addr» => «127.0.0.1», «port» => 3000]]]
$aerospike = new Aerospike($config)
$key = $aerospike->initKey («test», «services», 1)
$aerospike->put($key, [«response» => $response])
$aerospike->close()
```

Note: author's development

Table 3

Example of Dockerfile for Configuring a PHP Image with mysqli, pdo_mysql, soap, and intl Extensions

```
FROM php: 7.1-fpm
RUN docker-php-ext-install mysqli pdo_mysql soap intl
```

Note: author's development

cohesively. Developers can replicate production-like configurations on their local machines, ensuring consistent behaviour across all stages of application deployment. The pre-configured Docker image eliminates potential compatibility conflicts. By leveraging Docker's volume mapping and networking capabilities, database containers can be seamlessly connected to the PHP container, ensuring direct access to required resources. This configuration simplifies testing real-time database queries, SOAP interactions, and data formatting while offering an integrated development workflow that reduces context switching and accelerates iterative cycles (table 3).

This configuration ensures immediate availability of all extensions after the container is built, allowing developers to focus on application development and testing without managing underlying dependencies.

Conclusions. The analyzed extensions integrated into the Docker PHP 7.1 image demonstrate a cohesive ecosystem designed to provide an efficient, secure, and flexible environment for application development. MySQLi and PDO support a wide range of database operations, allowing developers to combine procedural and object-oriented approaches with enhanced flexibility and minimal configuration.

SOAP offers seamless XML-based communication, while Aerospike ensures low-latency operations for unstructured data. The Intl extension simplifies localization by automating the formatting of dates, numbers, and messages, improving accessibility for users in multilingual and multicultural contexts.

The Docker environment consolidates these extensions into a pre-configured containerized system, ensuring compatibility and consistent behaviour across the development, testing, and deployment phases. By utilizing Docker's volume mapping and networking capabilities, developers can efficiently connect database containers to the PHP container, facilitating real-time database queries, SOAP interactions, and data formatting within an integrated workflow.

This streamlined setup eliminates potential dependency mismatches, enabling developers to focus on application development and testing without worrying about underlying configurations. The result is a unified, productive development environment optimized for modern, scalable, and globally accessible applications.

Bibliography:

1. Htet Aung L., Funabiki N., Aung S., Zhou X., Xiang X., Kao W. – C. A web-based Docker image assistant generation tool for user-PC computing system. *Information*. 2023. Vol. 14. P. 300. DOI: 10.3390 / info14060300.
2. Laviale O. Installing PHP extensions from source in your Dockerfile. *Olvlvl*: website. 2019. URL: https:// olvlvl.com/2019–06-install-php-ext-source.html (last accessed: 16.01.2025).
3. Semeniuk V. Automated build for docker-php image. *Docker Hub*: website. 2025. URL: https://hub.docker. com/r/vadymsemeniuk/docker-php (last accessed: 16.01.2025).
4. Huang J., Zhang J., Liu J., Li C., Dai R. UFuzzer: lightweight detection of PHP-based unrestricted file upload vulnerabilities via static-fuzzing co-analysis. *24th International Symposium on Research in Attacks, Intrusions and Defenses*. *Association for Computing Machinery*. New York, NY, USA, 2021. P. 78–90. DOI: 10.1145 / 3471621.3471859.
5. Moroz L. bWAPP latest modified for PHP7. *GitHub*: website. 2018. URL: https://github.com/lmoroz/ bWAPP (last accessed: 16.01.2025).
6. Neef S., Kleissner L. PHUZZ: a grey-box fuzzer for PHP web applications. *GitHub*: website. 2024. URL: https://github.com/gehaxelt/phuzz (last accessed: 16.01.2025).
7. Q-Success. Usage statistics and market share of PHP for websites. *W3techs*: website. 2023. URL: https:// w3techs.com/technologies/details/pl-php (last accessed: 16.01.2025).
8. Sanoop T. XVWA is a badly coded web application written in PHP / MySQL that helps security enthusiasts to learn application security. *GitHub*: website. 2015. URL: https://github.com/s4n7h0/xvwa (last accessed: 16.01.2025).
9. The PHP Group. PHP: register_shutdown_function – Manual. *Php. net*: website. 2023. URL: https://www. php.net/manual/en/function.register-shutdown-function.php (last accessed: 16.01.2025).
10. The PHP Group. PHP: uopz – Manual. *Php. net*: website. 2023. URL: https://www.php.net/manual/en/ book.uopz.php (last accessed: 16.01.2025).
11. Watkins J. PCOV – CodeCoverage compatible driver for PHP. *GitHub*: website. 2023. URL: https://github. com/krakjoe/pcov (last accessed: 16.01.2025).
12. Zhao J., Lu Y., Zhu K., Chen Z., Huang H. Cefuzz: a directed fuzzing framework for PHP RCE vulnerability. *Electronics*. 2022. Vol. 11. No. 5. P. 758. DOI: 10.3390/electronics11050758.
13. Laaziri M., Benmoussa K., Khoulji S., Kerkeb L. A comparative study of PHP frameworks performance. *Procedia Manufacturing*. 2019. Vol. 32. P. 864–871. DOI: 10.1016/j.promfg.2019.02.295.

Семенюк В.В. ОПТИМІЗАЦІЯ ПРОЦЕСУ ЛОКАЛЬНОЇ РОЗРОБКИ ЗА ДОПОМОГОЮ ОБРАЗУ ДОКЕРА PHP, ЯКИЙ ПОСТАВЛЯЄТЬСЯ З ПОВНИМ НАБОРОМ ІНСТРУМЕНТІВ ІЗ КОРОБКИ: РОЗШИРЕННЯ БАЗ ДАНИХ ТА ІНТЕРНАЦІОНАЛІЗАЦІЇ

У роботі розглядається інтеграція критичних розширень PHP в образ Docker PHP 7.1, підкреслюється їхня роль у створенні згуртованого та ефективного середовища розробки. MySQLi, завдяки підтримці як процедурного, так і об'єктно-орієнтованого програмування, забезпечує безперебійну взаємодію з базою даних, одночасно включаючи безпечні з'єднання та можливості налагодження. Це виявляється необхідним для керування реляційними структурами даних і виконання складних операцій з базами даних. Так само PDO підвищує гнучкість шляхом стандартизації взаємодії між різними системами баз даних, сприяючи плавним переходам між платформами. SOAP розглядається як міст для зв'язку на основі XML, що дозволяє інтегрувати зовнішні API і перетворювати додатки PHP у веб-сервіси, здатні керувати різними типами даних і складними структурами повідомлень. Підкреслюється роль розширення як посередника в синхронізованому обміні даними між незалежними системами. Aerospike представлено як надійне рішення для керування базами даних NoSQL, яке відмінно підходить у сценаріях, що вимагають низької затримки обробки великомасштабних даних. Його розподілена архітектура забезпечує надлишковість даних і відмовостійкість, що підтримується розширеними методами реплікації, які оптимізують продуктивність за високих навантажень. Intl полегшує операції, що залежать від локалі, автоматизуючи форматування дат, чисел і повідомлень. Це спрощує розробку багатомовних програм, усуваючи складнощі ручної локалізації та покращуючи взаємодію з користувачем завдяки динамічному та культурно релевантному представленню даних. Образ Docker PHP 7.1 об'єднує ці розширення в попередньо налаштовану контейнерну систему, що дозволяє розробникам локально копіювати середовища, схожі на робочі, із узгодженою поведінкою на етапах розробки, тестування та розгортання. Ця інтеграція усуває конфлікти залежностей, використовуючи відображення томів і мережеві функції Docker для безпроблемного з'єднання контейнерів бази даних із контейнером PHP. У результаті запити до

бази даних у режимі реального часу, взаємодії SOAP і форматування даних ефективно керуються в рамках спрощеного робочого процесу. Це налаштування прискорює ітераційні цикли, зменшує перемикання контексту та забезпечує оптимізований процес розробки для масштабованих і глобально доступних програм.

Ключові слова: *контейнерізація, локалізація, масштабованість, сумісність, оптимізація, адаптивність.*