Client-based Knowledge Management Systems

Clemens H. Cap

October 2026

1 Introduction

PHP is a most successful language for server-side system construction. It has evolved into a modern language and is employed in a number of important and well-developed systems, most notably Wordpress and Mediawiki. The main disadvantage is that architecture-wise it is developed around a client-server infrastructure and thus has a bit of a prehistoric touch in a world of powerful client nodes and decentralized applications. For many applications, which are or a B2C nature, such as Magento, Drupal, or Joomla there is no need to make them client-capable. However, Wordpress and Mediawiki in particular offer services which are also of value for the individual user. Outside of a B2C setting they can be used for personal knowledge management and they offer a very widely accepted and considerably mature user interface and set of concepts. Moreover, Wordpress and Mediawiki offer elaborate sharing, synchronization and import-export concepts. A use case would thus be: Users employ a Wordpress or Mediawiki installation as a personal knowledge management system, which runs entirely on their mobiles, tablets, and laptops, thus providing perfect data and process sovereignty – at the same time offering a path to push their contents to classical Wordpress or Mediawiki installations, possibly running on servers. In addition to perfect data-storage sovereignty this infrastructure would also offer stable and permanent offline access to the respective knowledge management systems, which is a non-negligible aspect even in times of Starlink Internet access.

2 Research Questions

The basic line of research thus is: How can PHP-based systems, with Wordpress and Mediawiki being the particularly guiding examples, be migrated to

run on deployment environments which are less server-bound. Particularly interesting technologies comprise browser-side and node.js-grade Javascript and more especially WASM.

There is an ongoing activity of porting PHP to WASM and there is an experimental project for carrying this over to provide a browser-based Wordpress installation. Prior art is documented here:

1. `https://wasmer.io/posts/how-webassembly-is-powering-wordpress`
2. `https://developer.wordpress.org/playground/developers/local-development/php-wasm-node/`
3. `https://developer.wordpress.org/playground/developers/architecture/wasm-php-overview/`
4. `https://github.com/WordPress/wordpress-playground`

Further initial probing into the topic can be found at this public ChatGPT dialogue.

`https://chatgpt.com/share/6ab78892-7dbc-83ed-8e03-633acc8d03e1`

The research questions to be tackled within this Thesis center around these questions:

1. How well does the Wordpress-WASM test-implementation really work? Interesting aspects comprise the use of Wordpress plugins, the migration of native PHP plugins to WASM, the overall performance from a users point of view and the startup latency.
2. Which obstacles are likely to be met in a similar project, when the approach of the Wordpress-WASM project is thrown at Mediawiki?
3. To which extent can vibe-coding AI tools support this migration process?

Note: Use of AI-based tools is not only encouraged but forms a central part of the Thesis. The use of Anthropic, OpenAI or similar tools is suggested and the token costs may be subsidized by the Chair.

Requirements: The successful applicant for this thesis topic has – or is ready to acquire before starting – good knowledge of PHP, Javascript and web-solution design.