AMPLIACIÓN INGENIERÍA DEL SOFTWARE

José María Fernández Gómez

GLOBAL CAMPUS
NEBRIJA

INDICE

1. CASO DE ESTUDIO

En este informe se va a exponer un modelo de migración de una plataforma tecnológica en una compañía, esta plataforma fue desarrollada en PHP basándose en un entorno Symfony hace 4 años por lo que la tecnología está dificultando bastante el proceso de expansión y extensión de las capacidades funcionales.

Se busca como objetivo final mejorar el rendimiento y la eficacia del aplicativo a nivel global por medio de una migración tanto de lenguaje de programación base como de entorno de desarrollo, esto es debido a la antigüedad y a la menor tasa de actualización que posee este lenguaje frente a otros competidores más modernos, como se podrá observar más a delante en la comparación directa dentro del nuevo modelo propuesto.

Por otro lado, se persigue migrar el framework de desarrollo por el simple motivo del cambio del lenguaje de programación base.

El resultado esperado de esta migración es una disminución considerable de la estructura básica del aplicativo (en cuanto a tamaño por líneas de código se refiere) a la par que un rendimiento óptimo a la hora de interactuar con las API REST externas puesto a que el lenguaje PHP no está nativamente orientado a la comunicación con este tipo de interfaces. Como consecuencia de estos avances y mejora del rendimiento se espera eliminar de manera prácticamente total la deuda técnica dentro de esta teniendo la deuda de infraestructura, documental y de código.

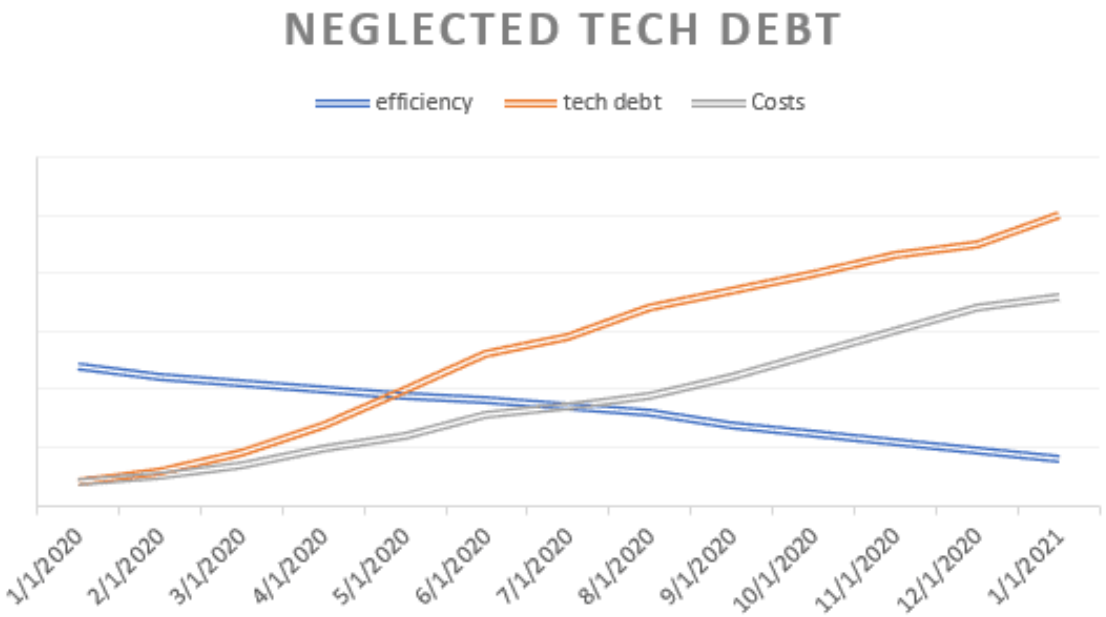
En última instancia se propondrá como por medio de este proceso podremos evitar las deudas de prueba, documentación y de proceso, todas estas siendo subconjuntos de la propia deuda técnica que posteriormente se aclarará, por la misma naturaleza de esta migración que supondrá una refactorización completa del código y por lo tanto una manera de comenzar estos procesos de documentación y testeo desde la base.

2. DEUDA TÉCNICA EXISTENTE

¿Qué es la deuda Técnica?

Según un artículo publicado por la empresa alemana de telecomunicaciones IONOS la deuda técnica se definiría como" *Los errores, carencias y deficiencias deliberadas o involuntarias en el código que se generan por falta de comunicación, dirección de equipo, cualificación o publicación apresurada de productos y que aumentan constantemente debido a la falta de refactorización.* "Esto podría dividirse en dos partes, la deuda técnica que se genera antes del propio lanzamiento del producto, y la que se genera después con el paso del tiempo y es inherente a la tecnología empleada.

Esto es de gran relevancia debido al grave impacto negativo que puede suponer ante una plataforma de las mismas características que nos atañe, ya que a la hora del mantenimiento y la actualización puede verse considerablemente lastrada por esta deuda técnica. A continuación, se desglosará esto mismo en subconjuntos de deuda y las causas actuales que generan estos inconvenientes.



Source: ImpactMakers

1.1. DEUDA DE INFRAESTRUCTURA

Nos encontramos ante uno de los mayores fallos en cuanto al lenguaje base PHP se refiere, al ser un lenguaje que originalmente se dedicó para páginas web de uso personal y relativamente reducido es subóptimo a nivel de escalabilidad, por lo que ahora cuando queremos expandir la página, a nivel de funcionalidades o de capacidad en cuanto a tráfico se refiere, y muchas de las características que esta incluya va a ser muy complicado y pesado a nivel de código hacerlo debido a estas propias limitaciones del lenguaje.

El primer ejemplo claro que encontramos es la comunicación con bases de datos. PHP se desarrolló de cara a la comunicación con bases de datos que siguieran un modelo relacional, en particular comunicación con bases de datos de tipo SQL. Esto supone un problema de cara al cambio de paradigma que se está originando con relación al Big Data y a las ingentes cantidades de datos que se generan diariamente y se tienen que almacenar, estos datos suelen ser mejor almacenados en bases de datos de tipo no relacional ya que se prefiere una clasificación de datos con un modelo más ambiguo. Esto quiere decir que no existen relaciones claras y directas entre varias agrupaciones de datos y se tienen que almacenar por medio de criterios más generales y no por campos específicos; todo esto siguiendo el artículo de PHPEarth (2022).

Por otro lado, dentro de este tipo de deudas encontramos uno de los mayores inconvenientes del lenguaje PHP, su falta de seguridad.

Al ser un lenguaje de código abierto todo está disponible a cualquier usuario de internet, tenga o no conocimientos informáticos.

Por otro lado, el código fuente de la página también es fácilmente accesible para cualquier usuario de la plataforma, así que, si a esto le sumamos lo anterior, el hecho de que todo el mundo tenga documentación genera un agujero de seguridad bastante significativo.

En último lugar y también relacionado con el tema de la seguridad, se conocen muchas vías potenciales y vectores de ataque vinculados a las propias características del lenguaje PHP, al ser un lenguaje de scripting y que permite la interpretación de código externo por medio de subida de archivos de extensión php podría darse el caso de brechas de seguridad como RCE (Remote code execution) o LFI (Local File Inclusion) como se explica en el artículo de Zeena Grimoire (2022), por medio de funciones que actúan a nivel de sistema proporcionadas por la parte de scripting de PHP como pueden ser system(), exec(), passtrhu() entre muchas otras.

1.2. DEUDA OBSOLETA/ACCIDENTAL DESCONOCIDA

Aquí nos encontramos en una situación que puede beneficiar a la par que añadir aspectos negativos al uso de PHP, la deuda obsoleta.

Partiendo de la base que PHP es una tecnología que está comenzando a quedarse atrás de manera considerable frente a los otros lenguajes de desarrollo web, podemos remarcar que la deuda obsoleta por el hecho de simplemente usar este lenguaje es bastante alta.

Pero por otro lado hay un aspecto que se ha mencionado ya que puede beneficiar de manera considerable a PHP, esto es la poca disponibilidad de librerías que este posee, al no tener prácticamente dependencias externas estamos minimizando las posibilidades de que alguna de estas se quede obsoletas o entren en un proceso de regresión pudiendo aumentar así la deuda obsoleta de la plataforma en cuestión.

1.3. DEUDA BIT ROT

Este tipo de deuda aparece por dos causas principalmente dentro de nuestro aplicativo, el primero e inevitable es el paso del tiempo, el añadir nuevas funciones a posteriori va aumentando la complejidad del código, y si a esto le sumamos la poca eficiencia que tiene PHP nos hallamos ante una combinación fatal, por otro lado, por el simple hecho de que el tiempo pase se van descubriendo y estandarizando nuevos paradigmas a la hora de codificar y desarrollar aplicativos web; esto como consecuencia tiene una eficiencia menor en comparación a otros aplicativos de la competencia, la complejidad de nuestro aplicativo no crecería ni su rendimiento disminuiría, siempre y cuando no se añadieran nuevas funcionalidades, pero sería considerablemente menos eficiente en comparación a desarrollos nuevos. Esta complejidad es de alta importancia de cara al cómputo total de la deuda técnica, ya que un código de alta complejidad a la larga acabará degradándose más y proporcionando un rendimiento peor, el cual irá aumentando más y más con el paso del tiempo, para poder medir esa complejidad tenemos las métricas de complejidad CC (complejidad ciclomática) o el sistema estandarizado denominado por la O(Big-O).

Como segunda causa principal nos encontramos con un tema ya comentado previamente, y es la antigüedad y las pocas actualizaciones que recibe PHP, al ser un lenguaje que fue desarrollado hace tanto tiempo hay muchas funcionalidades básicas que no son implementables o no están nativamente embebidas en este lenguaje, y al estar recibiendo tan pocas actualizaciones no se espera que la situación mejore, por lo que inevitablemente nos vamos a encontrar un código que para realizar la misma función que uno desarrollado en un lenguaje más moderno va a emplear un número mayor de LDC y sin duda una complejidad mayor, incrementando así de manera notable la deuda por Bit Rot.

1.4. DEUDA DE CODIGO

Otra de las desventajas asociadas al lenguaje PHP es la poca potencia nativa que este posee, al ser un lenguaje originalmente creado en el año 1995 no se tomaron en cuenta funcionalidades que hoy en día son estándares en el ámbito de navegación web, como por ejemplo la implementación de los métodos GET y POST que no se implementaron hasta el año 2002 o las diversas funcionalidades que hoy se necesitan en la capa de sesión descrita por el modelo ISO de comunicaciones como el protocolo RPC, que no fue implementado hasta el año 2008.

Por lo explicado anteriormente nos encontramos con un gran problema a la hora de ir añadiendo funcionalidades, los dos grandes motivos por los que esto sucede es porque, en primer lugar, PHP no se diseñó en el paradigma informático actual, esto refiriéndose a las arquitecturas y funcionalidades que están siendo estandarizadas hoy en día, un ejemplo claro y que afecta directamente al proyecto es la diferencia entre arquitecturas de las APIs, cuando PHP se desarrolló el estándar era usar el protocolo SOAP para estas y el estándar de ahora es la arquitectura REST; por lo que muchas de las funcionalidades que a día de hoy se requieren no son compatibles o por lo menos nativamente óptimas para este lenguaje, y en segundo lugar porque si se da el caso de que estas funcionalidades sí que se pueden implementar finalmente se tendrían que hacer por medio de una cantidad muy grande de líneas de código, cosa que sería nefasta para el desempeño de la plataforma por el hecho de que el motor de interpretación de PHP.

1.5. DEUDA DE DOCUMENTACIÓN

Al tratarse de una plataforma con relativamente una vida larga inevitablemente nos encontramos con deudas de documentación, esto debido a que en primer lugar los estándares de documentación con el tiempo van evolucionando y hoy en día se han visto modificados con respecto a la fecha de desarrollo original del proyecto. Estos estándares mencionados vienen regidos por las normas que figuran en el “IEEE 1063-2001”. Aquí podemos observar cómo primero se desarrolló este estándar, pero hasta el 2001 no fue aprobado y tuvo una reedición en el año 2007. Por todo esto mencionado anteriormente se han generado unas deudas de documentación inevitables ligadas al paso del tiempo, pero como se propone un cambio total de código base estas deudas quedarían prácticamente saneadas por medio de una documentación completa y realizada simultáneamente al desarrollo del nuevo código.

1.6. DEUDA DE DEFECTOS Y DEUDA DE PRUEBA

En este apartado veremos las carencias que tiene PHP y Symfony a la hora de poder testear un bloque de código o un programa completo y lo que esto puede suponer a nivel de deuda defectuosa y de deuda de prueba.

Todo este problema gira en torno a la falta de herramientas de depuración de código nativas en PHP, estas herramientas son muy potentes a la hora de poder monitorizar el flujo del programa a nivel infinitesimal e ir observando el comportamiento de cada variable durante la ejecución del programa, por lo que a nivel de eliminación de la deuda defectuosa son increíblemente útiles.

Por otro lado con estas herramientas de depuración se pueden simular bloques de código, dando la posibilidad así de realizar pruebas de componente o de bloque con entradas seleccionadas por el propio usuario, abriendo así el abanico de pruebas a realizar de manera considerable, estas pruebas a realizar podrían ser de tipo caja negra o blanca ya que como se ha indicado previamente se puede profundizar a nivel de variable con estas herramientas, o si no interesara simplemente se le dan valores a la entrada y se observa la salida. Esto último también es bastante relevante ya que gracias a ese control de entradas y salidas se podrían generar pruebas de estrés, de carga, de funcionamiento correcto, de humo…

Por lo que, en conclusión, el simple hecho de que PHP o Symfony no tengan habilitadas herramientas de depuración y gestión de errores de manera nativa incentiva la aparición de deudas defectuosas y dificulta mucho la labor del QA y testing provocando una considerable deuda de prueba.

1.7. DEUDA DE PROCESO, DEUDA DE-LIBERADA Y DEUDA DE DISEÑO.

Estos problemas generados por estas clases de deuda se originan a la hora del diseño previo del producto y el desarrollo y creación de este mismo. Como no se puede especular sobre lo sucedido previamente durante el desarrollo original de este aplicativo simplemente se va a proponer un sistema para tratar de minimizar las posibles consecuencias que aumenten el cómputo de duda técnica a futuro.

Para mitigar la deuda de proceso se va a proponer usar un modelo Agile basado en la metodología Kanban según Julia Martins (2020), distribuyendo las diferentes necesidades y requisitos en historias de usuario y creando diferentes “tarjetas” que se implementaran en un tablero Kanban, cumplimentando así el primer requisito de esta metodología, visualizar los objetivos; mediante este proceso de visualizado y ordenamiento de los requisitos por otro lado se va a tratar de mitigar la deuda de diseño, no excluyendo así ninguna de las ideas o diseños sugeridos por el equipo e implementando estos en el tablero Kanban como potenciales soluciones.

Para seguir mitigando la deuda de proceso se eliminará la multitarea de cara a cada miembro o subequipo dentro del proyecto, haciendo así que cada núcleo de desarrollo se centre en objetivos únicos, mejorando así la productividad de cara a minimizar retrasos y tener que acelerar algún lanzamiento sacrificando calidad, hecho principal que hace que aumente la deuda deliberada. Con esto satisfacemos los dos siguientes requisitos de Kanban, eliminar las interrupciones y gestionar el flujo de trabajo.

Por último, se va a explicar porque se elige Kanban como metodología de trabajo; al tratarse de una metodología ágil que no se divide en procesos tan separables como puede ser Scrum (S. Planning, S. review, S. Grooming …) es más complicado saltarse una etapa o una parte fundamental del proceso, a parte que una de las claves principales de esta metodología se basa en el respeto al proceso, los roles de equipo y las responsabilidades dentro de este.

Con todo esto se pretende minimizar estas deudas a futuro, y quedan plasmadas las políticas básicas a seguir que no solo minimizarán las consecuencias de estas deudas, si no que permitirán trabajar con Kanban de manera óptima y satisfactoria.

3. NUEVO MODELO PROPUESTO

Se va a proponer un modelo basado en el lenguaje de programación Kotlin, este lenguaje se originó por primera vez en el año 2016, por lo que es de los lenguajes de programación más nuevos y con más funcionalidades contemporáneas que existe hasta la fecha.

Por otro lado, se va a emplear el framework Spring Boot, este framework se ha seleccionado ya que contiene un gran número de bibliotecas orientado a la creación e integración de API RESTfuls. Por medio de 4 librerías que subyacen dentro de la biblioteca "`org.springframework`" integrada en el framework podemos montar una API REST básica que se encargue de publicar, y mostrar mensajes, a partes de integrarlos en una base de datos de tipo SQL por medio de solo dos archivos de código y cada uno con menos 15 líneas de código cada uno como se muestra en el artículo escrito por los propios desarrolladores de Kotlin, Jetbrains.

El problema de PHP: Interpretación y librerías

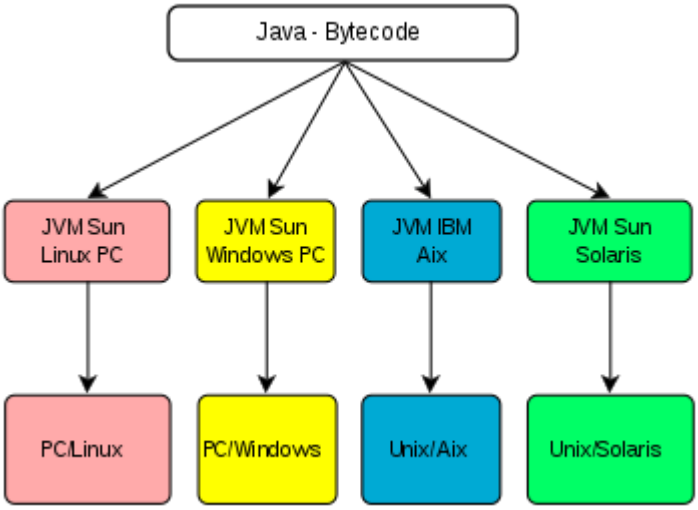
Ahora se procederá a explicar las desventajas de motor de interpretación de PHP, y el problema de dependencias y escasas librerías que este posee.

A diferencia de otros lenguajes clásicos de programación como pueden ser C o Java, PHP no se compila, se interpreta, esto difiere a lo previamente mencionado en que no se genera un archivo ejecutable cada vez que queremos actualizar el código, si no que se ejecuta directamente por el ordenador a través de varios pasos de análisis e interpretación de este código, y aunque esto pueda sonar mejor que el proceso de compilación realmente es más lento ya que el uso de recursos del ordenador es mucho más alto en estos pasaos que en los de compilación, y por otro lado cada vez que se quiera ejecutar el programa se deberá pasar por el proceso de interpretación cosa que es considerablemente más larga que una simple ejecución de un archivo ejecutable resultado del proceso de compilación.

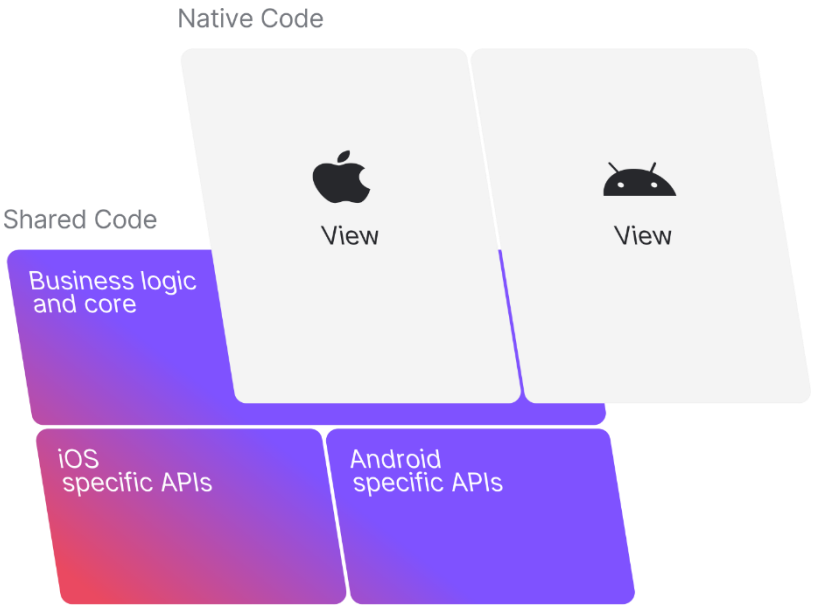
Por último, se va a analizar uno de los puntos más significativos a tomar en cuenta con respecto al uso de PHP, esto es el gran problema de sus librerías. PHP contiene muy pocas librerías a nivel nativo, cosa que no sería demasiado relevante si existieran frameworks de trabajo que poseyeran esas carencias que PHP nos trae nativamente, como puede pasar con JavaScript y nodeJS.

El problema vuelve a radicar en que los mejores frameworks de PHP, como puede ser Symfony, solo incluyen hasta 30 librerías, esto es un inconveniente ya que cada función que no esté disponible en estas limitadas librerías va a tener que ser escrita de manera manual, por lo que va a tener que pasar el proceso de interpretación aumentando aún más el tiempo de ejecución ralentizando muchísimo el despliegue del aplicativo.

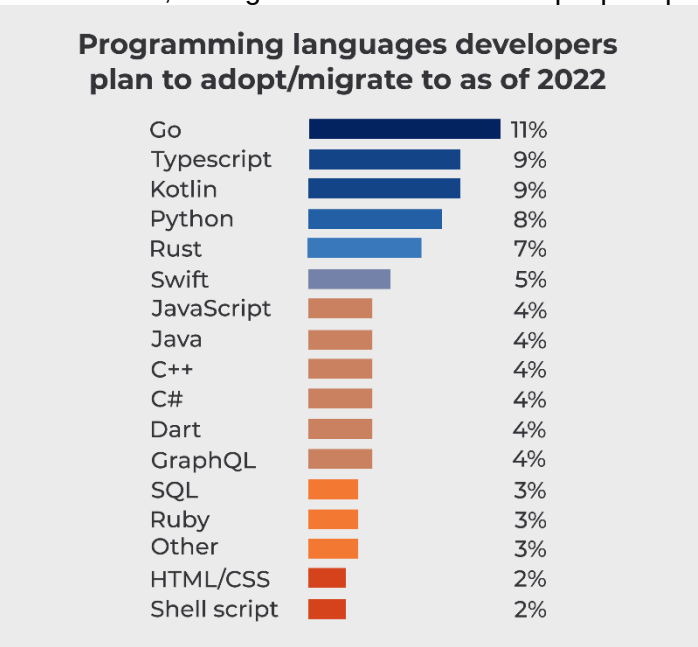
¿Por qué Kotlin?



Como se ha indicado previamente Kotlin es un lenguaje de programación contemporáneo que está basado en Java. Uno de los puntos fuertes que comparte con Java es que corre directamente sobre la JVM (Java Virtual Machine). Esto quiere decir que el programa que se esté ejecutando no corre directamente sobre el sistema operativo del sistema huésped, si no que se crea una "capa" adicional que simula un entorno idéntico para todos las máquinas que existen independientemente de su sistema operativo o su arquitectura; por lo que es uno de los mejores ejemplos de multiplataforma, esto porque el programa se va a comportar exactamente igual ya sea ejecutado en un servidor con sistema base Linux, Windows o incluso Android, sistemas que tienen una arquitectura a nivel de hardware diferente. Por otro lado, presenta una ventaja clara frente a lenguajes como Java, no sólo puede ser compilado y ejecutado por la máquina virtual de Java, si no que también puede ser interpretado. Como se explicó previamente un lenguaje interpretado es analizado y ejecutado prácticamente de manera simultánea, y lenguajes compilados como Java no poseen la capacidad de hacer esto, pero Kotlin es un lenguaje que se diferencia del resto gracias a que puede ser compilado e interpretado; a parte presenta otra ventaja frente a PHP en que corre sobre el intérprete de JavaScript, un procesador del lenguaje más moderno y eficiente que el de PHP. Realmente no ocurre un proceso de interpretación puro como pasaba con PHP o JS, si no que se da el proceso de transpilación, que de forma resumida es una especie de traducción entre lenguajes fuentes, que a efectos prácticos es lo mismo que una interpretación en otro lenguaje.



Por otro lado, se elige Kotlin debido a lo simple pero potente que es, la curva de aprendizaje de este lenguaje es bastante favorable, sobre todo de cara a usuarios y/o programadores que ya hayan tenido contacto con otros lenguajes clásicos orientados a objetos como pueden ser Java o C++, con la diferencia de que este tiene una sintaxis más sencilla. Por otro lado, es más tolerante a fallos que los mencionados anteriormente ya que puede reconocer de manera automática valores en variables no deseados y corregirlos por sí solo en vez de lanzar una excepción o error, esto por medio de funciones de comprobación de tipos y reemplazo de valores, estas funciones son nativas del lenguaje y no requieren librerías ni compilables externos.



Como último punto a resaltar se recomienda este lenguaje debido a la gran interoperabilidad que tiene con prácticamente todos los componentes de la arquitectura de un aplicativo web. En primer lugar, tenemos Kotlin server-side que se encarga de la distribución y gestión del backend de manera prácticamente íntegra, ya que permite la propia creación de bases de datos en el propio código del backend y por otro lado permite una gran variedad de integraciones

con APIs externas e innumerables plataformas. Plataformas como Netflix, AWS, Pinterest o Flipboard ya están basando sus sistemas en este nuevo lenguaje, por lo que a la hora de la integración de diferentes servicios externos a nuestro propio aplicativo será mucho más sencillo y eficiente basar la arquitectura en el mismo lenguaje que estas plataformas. También encontramos que Kotlin no solo proporciona efectivos sistemas de desarrollo del backend, si no que contiene una gran cantidad de bibliotecas de manera nativa para que el desarrollo del frontend y de la UI sean óptimos, aquí hayamos 7 librerias que a nivel global son de las más utilizadas a nivel global.

Library	Details
stdlib ↗	The Kotlin standard library included in all projects by default.
kotlinx.browser	The Kotlin library for accessing browser-specific functionality, including typical top-level objects such as document and window.
kotlinx.html	The Kotlin library for generating DOM elements using statically-typed HTML builders.
Ktor ↗	The Kotlin multiplatform library for networking.
KVision ↗	A third-party object-oriented web framework for Kotlin/JS.
fritz2 ↗	A third-party lightweight, high-performance, independent library for building reactive web apps in Kotlin that are heavily dependent on coroutines and flows.
Doodle ↗	A third-party vector-based UI framework that uses browser's capabilities to draw user interfaces.
Compose for Web, a part of Compose Multiplatform ↗	The JetBrains framework that brings Google's Jetpack Compose UI toolkit ↗ to the browser.
kotlin-wrappers ↗	Provide convenient abstractions and deep integrations for one of the most popular JavaScript

3.1. COMPARACIÓN ENTRE PHP Y KOTLIN

Ahora se va a proceder a hacer una comparación directa entre el lenguaje empleado originalmente en el aplicativo, PHP, con el lenguaje dentro del modelo propuesto.

En primer lugar, se muestra una encuesta realizada por una plataforma de recomendaciones de productos llamada Slant.co, en esta encuesta se evalúan criterios puramente subjetivos y no de carácter técnico, a la izquierda se muestran los resultados obtenidos por el lenguaje PHP y a la izquierda los resultados dirigidos a Kotlin, como se puede observar según los usuarios Kotlin es ampliamente el vencedor en cuanto a las cuestiones tratadas se refiere.

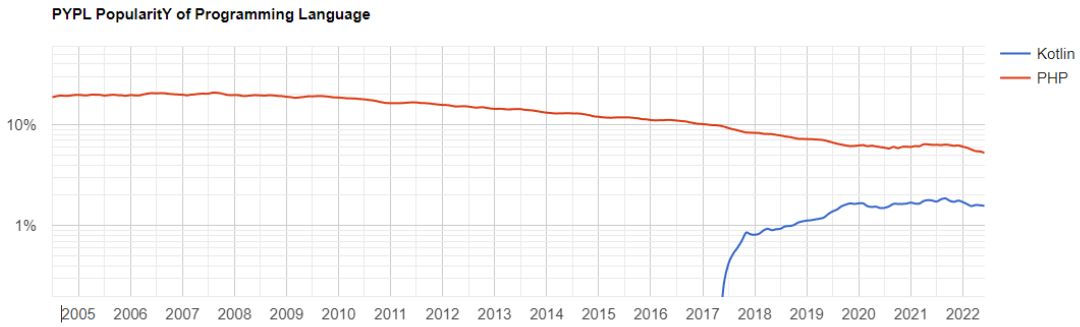
#43	What is the best programming language to learn first?	#9
#72	What are the best (productivity-enhancing, well-designed, and concise, rather than just popular or time-tested) programming languages?	#19
#19	What are the best server side programming languages?	#9
#39	What are the best languages that compile to JavaScript?	#4
#12	What is the best programming language to learn for backend developers?	#14
#14	What is the best programming language to learn?	#9

Slant.co *Programming Languaje*

Aquí se muestra una tabla donde se muestra la popularidad de los lenguajes de programación en el año 2019 según una serie de estudios realizados por la página PYPL, la cual basa sus resultados en el número de búsquedas que han tenido diferentes tutoriales para cada lenguaje de programación. Y se muestra como el lenguaje que más auge ha tenido ha sido sin duda Kotlin.

Rank	Change	Language	Share	Trend
1		Python	29.21 %	+4.6 %
2		Java	19.9 %	-2.2 %
3		Javascript	8.39 %	+0.0 %
4		C#	7.23 %	-0.6 %
5		PHP	6.69 %	-1.0 %
6		C/C++	5.8 %	-0.4 %
7		R	3.91 %	-0.2 %
8		Objective-C	2.63 %	-0.7 %
9		Swift	2.46 %	-0.3 %
10		Matlab	1.82 %	-0.2 %
11	↑	TypeScript	1.77 %	+0.2 %
12	↑↑↑↑	Kotlin	1.55 %	+0.6 %

Por último, se muestra un gráfico donde se compara directamente la popularidad de PHP frente a la de Kotlin, y vemos que claramente mientras que este primero comienza a entrar en decadencia, el nuevo lenguaje solo asciende.



Ahora se compararán datos objetivos de manera directa entre ambos lenguajes de programación.

	Kotlin	**PHP**
Año de salida	2016	1995
Numero de releases con manutención	>10	2
Tiempo entre actualizaciones	< 6 meses	1 Año
Integración con servicios REST	Sí, desde el 2008	Nativo
Integración con servicios graphql	Si, con librerías externas	Nativo
Método de ejecución	Interpretación, compilación sobre la JVM	Interpretación
Multiplataforma	Todas las que soporten JVM	Linux, Windows o Mac
Orientado a objetos	Sí	Sí
Programación funcional	Sí	No
Tipado	Fuerte y estático	Débil
Uso de hilos	Si, multithread	No

Como se puede observar en la tabla Kotlin tiene aspectos similares a PHP como pueden ser la programación orientada a objetos o el soporte a APIS externas ya sean de tipo REST o graphql, pero por otro lado tenemos una gran ventaja a la hora del soporte técnico, documentación y cantidad de releases por año en el caso del nuevo lenguaje, cosa que a la hora de desarrollar una plataforma web que debe estar en continuo desarrollo y evolución es extremadamente importante.

Como se puede observar en la tabla, nos encontramos con un gran problema en PHP, el tratamiento de peticiones a través de procesos ¿Qué quiere decir esto? Es un gran problema ya que nos limita de manera significativa cuantas peticiones simultaneas se pueden atender, este límite siendo generado por el número máximo de núcleos que tenga el procesador del sistema servidor. Kotlin supera con creces este problema empleando hilos, los cuales son subprocesos que el propio procesador puede tratar de manera concurrente o empleando algoritmos de distribución de tiempo y/o recursos, haciendo que el tratamiento y procesamiento de estos hilos, y por lo tanto peticiones, se haga de una manera óptima y eficiente, permitiendo así tratar más peticiones por unidad de tiempo que si empleara procesos, como hace PHP.

3.2. COMPARACIÓN ENTRE SPRING Y SYMFONY

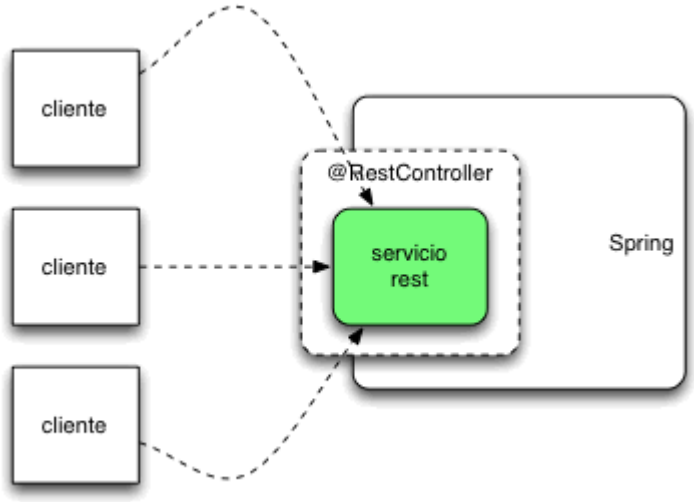
	Spring	**Symfony**
Escrito en	Java	PHP
Slant ranking para backend (lower better)	10	14
Slant ranking para API REST (lower better)	4	13
Lanzado en	2002	2005
Ultima versión estable	Julio 2021	Enero 2022
Multiplataforma	Si	No
Pruebas automatizadas	Unitarias, integración, funcionales	Unitarias y funcionales
Inyección de dependencias	Maven (Estandarizado)	Propio de Symfony
Servidores nativos	Si	No
Orientado a	API REST	Aplicaciones web server side
Serialización de datos	JSON (estándar de API REST)	YAML

En último lugar se va a comparar ambos frameworks de cara al desarrollo, integración y desempeño con respecto a uno de los principales requerimientos del aplicativo final, esto siendo la integración y uso de API RESTfuls. Ambos frameworks fueron desarrollados y son bastante competentes a la hora del desarrollo de estos servicios, pero la comunidad tiene claro que sin duda Spring es mejor a la hora del desarrollo y rendimiento de estos, como se puede observar en el segundo apartado de la tabla comparativa superior. Ahora trataremos de analizar los detalles técnicos y objetivos por los cuales la comunidad ha llegado a tal conclusión. En primer lugar viene tenemos una clara ventaja en cuanto al desarrollo multiplataforma de Spring frente a Symfony, esto viene ligado al lenguaje de programación y no tanto al framework, pero al trabajar en conjunto sigue sumando puntos para esta primera dupla lenguaje/framework, esto es bastante relevante debido a la posibilidad de ejecutar y comunicarse con este tipo de interfaces desde cualquier dispositivo o máquina independientemente de su arquitectura y/o sistema operativo, por lo que el aplicativo en general y sus interfaces (Ya sean API REST en este caso o de otro tipo) son portables, intercambiables y más versátiles que si fueran desarrolladas en un lenguaje dependiente de sistema o arquitectura, como lo es PHP/Symfony.

Por otro lado, encontramos una mejora en el ámbito de la optimización, esto por medio de la configuración, ya que el entorno Spring, al trabajar muy estrechamente con la JVM, permite configurar esta misma a muy bajo nivel de cara a la optimización en cuanto a consumo de recursos de la máquina servidor o en cuanto a mejorar rendimiento y potencia pura de esta, esto supone una gran ventaja frente PHP ya que este no posee esa capa de abstracción que proporciona la JVM y se interpreta directamente sobre el sistema operativo por medio de su intérprete, cosa que es bastante más complicada de

configurar.

De cara al desarrollo de estos servicios REST, Spring parte con una gran ventaja ya que está bastante orientada de manera deliberada al desarrollo de estos, y encontramos muchas ayudas que en el entorno Symfony no encontramos como el uso de plantillas preprogramadas o de anotaciones propias características de estas interfaces, en el artículo de Javin Paul que se deja en las referencias se profundiza más en este tema.



Por último, la parte en cuanto a integración con servidores embebidos dentro del propio framework es mucho más potente en Spring, concretamente dentro del entorno Spring boot, ya que soporta de forma nativa el desarrollo y testeo de servidores basados en arquitecturas reputadas y considerados estándares de mercado como puede ser Tomcat o Nginx, esto supone una pequeña ventaja sólo ya que la dupla PHP/Symfony también puede hacer uso de estos servicios, pero la integración y desarrollo entre ambos es más complicada ya que no son dependencias directas y naticas como lo pueden ser en Spring boot.

RANKED IN THESE QUESTIONS

COMMON QUESTIONS

#10	What are the best backend web frameworks?	#14
#4	What are the best web frameworks to create a web REST API?	#13

Source: slant.io

4. CONCLUSIONES

Por medio de este caso de estudio se ha buscado la forma de reducir la deuda técnica existente en el proyecto del servicio web referenciado durante todo el escrito y proponer una solución coherente, moderna y que permita la escalabilidad y expansión funcional a futuro.

Como ha quedado demostrado, tras en análisis técnico que se ha realizado a lo largo de este informe, el proyecto del aplicativo actual posee una grave deuda técnica y el potencial de desarrollar a futuro aún más de esta, el motivo principal, cosa que también ha quedado plasmada en este parte, está profundamente ligado al lenguaje de programación base y al framework junto al que se programó.

Como modelo alternativo se ha propuesto un lenguaje mucho más moderno y orientado a lo que se busca específicamente para este aplicativo, una integración con interfaces de aplicación de tipo REST, a parte de un correcto funcionamiento del lado del servidor y del cliente.

Las principales ventajas se van a exponer ahora de manera resumida a forma de conclusión de cara a una justificación unánime y certera del nuevo modelo propuesto.

1. Escalabilidad: Queda reflejado en la última parte del informe como por medio del cambio de lenguaje de programación base el aplicativo podría soportar una mayor cantidad de usuarios simultáneamente debido al buen uso de recursos y técnicas (Multithreading) que este nos proporciona de manera nativa frente a su competidor.
2. Multiplataforma: Como se ha indicado en el funcionamiento del nuevo lenguaje propuesto, este es uno de los mejores que existen a día de hoy en el mercado orientado a la multiplataforma y la portabilidad, por la independencia que este tiene del sistema operativo y la arquitectura gracias a la capa de abstracción que proporciona la JVM, cosa que PHP no posee.
3. Modernidad: Kotlin es un lenguaje muy joven en comparación a sus competidores, cosa que deja mucho margen para más desarrollo e integraciones funcionales de este, pero teniendo la seguridad que nos proporciona la base que tiene de Java, un lenguaje más que reputado y asentado.
4. Integración de servicios REST: El nuevo framework propuesto tiene como objetivo estar orientado a desarrollos web como Symfony, con la diferencia de que nativamente se ideó para la integración sencilla y nativa de interfaces de aplicación de tipo REST.
5. Tendencia de la nueva tecnología: Si observamos las tendencias de ambas duplas lenguaje/framework observamos que claramente PHP/Symfony está en decadencia en cuanto a uso frente al claro auge que está teniendo Kotlin/Spring, por lo que a futuro integraciones con otros servicios externos serán mucho más sencillas si se emplea el mismo entorno.

Por estas razones y todo lo plasmado a lo largo de este informe se propone este cambio de sistema, esto no quitará de que no se necesiten mantenimientos regulares ni actualizaciones futuras, pero si garantizará un desarrollo más sencillo de estos y un rendimiento inmediato y a futuro próximo notablemente mejor del aplicativo y sus servicios.

5. BIBLIOGRAFIA

Recursos:

Kevin Cox *WHAT IS TECHNICAL DEBT?* Recuperado el 7/06/2022 de:

https://www.impactmakers.com/blog/avoid-technical-debt-in-aws/

PHPEarth,2022 *MongoDB vs. MySQL.* Recuperado el 1/06/2022 de:

https://docs.php.earth/faq/db/mongodb-vs-mysql/

Zeena Grimoire, 2022.*Remote Code Execution PHP.* Recuperado el 1/06/2022 de:

https://vulp3cula.gitbook.io/hackers-grimoire/exploitation/web-application/rce

IONOS ,2020. *Deuda técnica: las posibles consecuencias de ahorrar en el desarrollo de software.* Recuperado el 1/06/2022 de:

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/deuda-tecnica-explicada/

Jetbrains Kotlin, 2022. *Create a RESTful web service with a database using Spring Boot* Recuperado el 1/06/2022 de:

https://kotlinlang.org/docs/jvm-spring-boot-restful.html#run-the-application

Rodríguez Pablo, 2021 *La deuda técnica, un lastre para las tecnológicas: un estudio señala que los informáticos pierden casi un día de trabajo a la semana para solventarlas.* Recuperado el 1/06/2022 de:

https://www.xataka.com/pro/deuda-tecnica-lastre-para-tecnologicas-estudio-senala-que-informaticos-pierden-casi-dia-trabajo-a-semana-para-solventarlas

Javin Paul, Why *Spring is the best framework for developing REST APIs in Java?* Recuperado el 7/06/2022 de:

https://medium.com/javarevisited/why-spring-is-the-best-framework-for-developing-rest-apis-in-java-784590e484a4

Slant.io, *Spring boot vs Symfony* Recuperado el 7/06/2022 de:

https://www.slant.co/versus/158/3758/~spring-boot_vs_symfony

Información referenciada:

Deuda Técnica:

Acervo Lima, *¿QUÉ ES LA DEUDA TÉCNICA EN EL DESARROLLO DE SOFTWARE Y CÓMO GESTIONARLA?* Recuperado el 1/06/2022 de:

https://es.acervolima.com/que-es-la-deuda-tecnica-en-el-desarrollo-de-software-y-como-gestionarla/

Tanya Perez, 2020, *Deuda de Diseño - ¿La conoces?* Recuperado el 1/06/2022 de:

https://medium.com/tanyaperezblog/deuda-de-dise%C3%B1o-la-conoces-bf3024483a06

Scrum Manager,2021, *Deuda técnica*, Recuperado el 1/06/2022 de:

https://www.scrummanager.net/bok/index.php?title=Deuda_t%C3%A9cnica

Rajiv Srivastava, 2022, *Understanding Technical Debt for Software Teams* Recuperado el 1/06/2022 de:

https://www.squadcast.com/blog/understanding-technical-debt-for-software-teams

Insitech, N/D, *¿Qué es la deuda técnica y qué la origina?* Recuperado el 2/06/2022 de:

https://go.insitech.com.mx/low-code-deuda-tecnica-que-es-como-gestionarla-y-como-reducirla/

Pablo Rodríguez, 2021, *La deuda técnica, un lastre para las tecnológicas: un estudio señala que los informáticos pierden casi un día de trabajo a la semana para solventarlas*, Recuperado el 2/06/2022 de:

https://www.xataka.com/pro/deuda-tecnica-lastre-para-tecnologicas-estudio-senala-que-informaticos-pierden-casi-dia-trabajo-a-semana-para-solventarlas

Desventajas de PHP:

Geeksforgeeks,2021, *Advantages and Disadvantages of PHP*, Recuperado el 2/06/2022 de:

https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/#:~:text=PHP%20lacks%20debugging%20tools%2C%20which,the%20features%20of%20Java%20language.

One, 2022, *¿Cuál es la última versión estable de PHP?*, Recuperado el 2/06/2022 de:

https://help.one.com/hc/es/articles/360002576038--Cu%C3%A1l-es-la-%C3%BAltima-versi%C3%B3n-estable-de-PHP-

Programación.net, 2021, *¿Por qué elegir PHP?* Recuperado el 2/06/2022 de:

https://programacion.net/articulo/por_que_elegir_php_143

Prasanna, 2021, *PHP Advantages and Disadvantages | What is PHP Language? Merits and Demerits of PHP*, Recuperado el 2/06/2022 de:

https://www.aplustopper.com/php-advantages-and-disadvantages/#Disadvantages_of_PHP

Stackshare, 2022, *Kotlin vs PHP*, Recuperado el 2/06/2022 de:

https://stackshare.io/stackups/kotlin-vs-php

Wikipedia, 2022, *PHP*, Recuperado el 2/06/2022 de:

https://es.wikipedia.org/wiki/PHP#Historial_de_lanzamiento

Frameworks PHP:

Claire Brotherton, 2021, *Los Frameworks PHP más populares para usar en 2022*,

Recuperado el 3/06/2022 de:

https://kinsta.com/es/blog/frameworks-php/

IONOS, 2017, *Symfony: un framework PHP adaptable con una comunidad fuerte*

Recuperado el 3/06/2022 de:

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/symfony-utiles-bibliotecas-php-para-tu-proyecto-web/#:~:text=Otra%20de%20las%20debilida-des%20de,ORM%2C%20desempe%C3%B1a%20un%20papel%20importante.

Junta Andalucía, N/D, *Symfony*, Recuperado el 3/06/2022 de:

https://juntadeandalucia.es/servicios/madeja/contenido/recurso/263

Documentación:

Everybody Wiki, 2021, *IEEE 1063*, Recuperado el 4/06/2022 de:

https://en.everybodywiki.com/IEEE_1063

O Pauli, 2021, La *importancia de la documentación de software* Recuperado el 4/06/2022 de:

https://techlandia.com/importancia-documentacion-software-sobre_538552/

IEEE,2007, *IEEE 1063-2001*, Recuperado el 4/06/2022 de:

https://standards.ieee.org/ieee/1063/1554/

Complejidad:

Makeitreal, 2019, *Complejidad (Big-O)*, Recuperado el 4/06/2022 de:

https://guias.makeitreal.camp/algoritmos/complejidad

Michael Pratt, 2013, *La Complejidad Ciclomática en PHP*, Recuperado el 4/06/2022 de:

http://www.michael-pratt.com/blog/15/La-Complejidad-Ciclomatica-en-PHP/

First Mark, 2018, 3 *Main Types of Technical Debt and How to Manage Them* Recuperado el 4/06/2022 de:

https://hackernoon.com/there-are-3-main-types-of-technical-debt-heres-how-to-manage-them-4a3328a4c50c

Kanban:

Julia Martins, 2020, *¿Qué es la metodología Kanban y cómo funciona?* Recuperado el 4/06/2022 de:

https://asana.com/es/resources/what-is-kanban

Kanbanize, N/D, *Qué es Kanban: Definición, Características y Ventajas,* Recuperado el 4/06/2022 de:

https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

Kanbantool, N/D, *¿Por qué utilizarla metodología Kanban?* Recuperado el 4/06/2022 de:

https://kanbantool.com/es/metodologia-kanban