First thing that needs to be asked is, why are you using old software? Is it because you are unable to update your server software? Or is it because the old dependencies that your software is using? Or maybe is it because lack of funds for upgrading? But we will take a look at each of those cases, because each of those represent different threat to business and can have different solution.

In this article I want to focus only on online software because technical depth issue is affecting online software hardest. I will also describe as if it was an outdated PHP issue, but this can actually affect any programming language and any script.

How Did You End Up with Technical Debt?

There are many reasons where you end up with technical depth. One of the reasons may be just not enough money to update, but actually updates should not be very expensive. So if you cannot afford regular updates then maybe your business is not earning enough money. Another thing can be lack of business process for updating everything and the lack of this procedure can affect many areas, not only the one that is outdated at the moment, as later other may emerge. Third scenario that I would imagine is that you received or bought outdated script that needs to be updated now, so the cost of investment will increase by the amount of update cost.

It is very important to find the core reason for outdated software because doing so will allow you to also avoid similar issue in the future So if you want do this correctly, the whole procedure should start with identifying root cause and amending (or implementing) it. As a result your business quality is improved so it can thrive and grow further.

How Old Software Affects SaaS Business Usability?

There are many aspects in which business is affected when using old software. First of such aspects would be of course security. if you are not using most up to date software it is possible that you did not receive newest security updates. Not receiving security updates can have impact both on your users and your business and your workers. In each of those cases data that you are processing may just be stolen. Another impact of unsafe software might be that it just stops working when other dependencies are updated.

Second aspect is usability. Without newest updates you are probably not receiving new features aswell as fixes to old. Such new feature could potentially improve user experience or workers experience, thus impacting positively business outcomes. For example, new feature could speed up processes for your employees thus improving their efficiency, and eventually business itself.

Usability is also dependent on speed of software. And new software often implements performance fixes aswell. If your software is lagging, users are getting impatient, angry and frustrated. As frustration of user is growing, they can decide that he will no longer use this software at all.

Of course, these are just examples, but I think I have convinced you that updating software is very important for business itself.

Is Using Outdated PHP Versions Dangerous for my Website?

The most generic answer to that question would be: **it depends, but in general, yes**.

Worst case is – no matter the error level – when you are getting error messages displayed on the front page or it’s blocking parts of website functionality (this should NEVER happen – because HOPEFULLY you turned off displaying errors on production and this will be visible only in logs! Otherwise it could be a HUGE security issue!). While it does not sound dangerous, it is information you are giving to a potential attacker for free. And it’s not uncommon to observe. In the edge case scenario, if data being shown on the front page uncovers too much, then the hacker does not even need to attack – he just logs in to your admin panel. You should never display in browser backend errors on your production software.

There are a few levels of php errors, with most common being: deprecated, notice, warning and error. In the case least impact case, when updating php to newer version, you will see deprecated errors. This means that in future revisions usage of such functionality will change. You should update as soon as possible because with next version your software will stop working – so without updating script you are unable to update whole software.

A slightly worse case is when you ignored the deprecation message or missed it and

now on update software stopped working at all. In such case, you will mostly receive warnings or errors and you won't be able to continue to use your software. If such a situation happens then you can only revert back to the old PHP version or you need to update script as soon as possible to get website working again.

Ability to revert back software updates is very important in software development. As you can see on the example above that's why at Sailing Byte we always aim to keep such possibility. Least convenient in such case but still acceptable solution should be reverting from backups which you should always have.

Which PHP Version Should I Use?

Simplest answer would be: **the newest one that your script is able to run on.** Preferably, of course, it would be the newest version of PHP available, that is in stable branch. If you ar asking this question, then you probably also should take a look at improvements in newest PHP versions.

Hosting providers mostly do not offer help when migrating old scripts to a new PHP version. Even though they allow you to change the PHP version. But should you use an older version that lacks security support? The obvious answer is – no.

And just a reminder – PHP 5.x is no longer be developed and is not receiving security updates. PHP 6 was never deployed. And even whole PHP 7.x branch will no longer receive any updates. You should always also avoid using version php 8.0 as it is no longer receiving security updates – so at the moment you must aim above 8.0. Acceptable versions of PHP is 8.1 and 8.2, but preferable would be 8.3 or 8.4. You can check official php.net website for. active support and security. support dates, but for your convenience I am attaching a table below.

Branch	Initial Release	Active Support Until	Security Support Until
8.1	25 Nov 2021	25 Nov 2023	31 Dec 2025
8.2	8 Dec 2022	31 Dec 2024	31 Dec 2026
8.3	23 Nov 2023	31 Dec 2025	31 Dec 2027
8.4	21 Nov 2024	31 Dec 2026	31 Dec 2028

So if you are not sure how to update or you are looking for someone to do it for you – hit Contact Form below and let us know. We can help you!

How to Reduce Impact of Updating Software?

First of all, you should **always** have backup and ability to revert. Don't even start any attempts before you are sure that you are able to revert changes.

Next question would be what is the acceptable downtime for the update? If you are running a small website, then probably you can allow yourself to be offline even for few hours. But if you are running big SaaS service you can either consider updating at inactivity times (like at night) or make a blue-green update system with seamless switch behind reverse proxy with fallback mechanism. For most in-development services it will be probably sufficient to have proper automated CI/CD pipeline with implemented minimum downtime update script built in. If you are not sure what I am describing here and you want to do it yourself – just go with the first option.

When downtime is planned, you shoul also inform users that it is planned. You might consider sending emails to them or at least leaving "maintenance" page on the homepage – so they can be reassured that everything is fine and your update is planned.

Next thing you should consider is of course testing it out first. Before "going wild" on production, first try to go through the whole process on test site (create one for staging if you don't have any at the moment!). On updating to a new PHP version you definitely first need to check your website on a separate test server. Then you can assess the amount of work needed to bring the full functionality on production website – if any work is needed. Sometimes, if your software was updated accordingly, your website will have no problems when switching to a new PHP version. In the end, test it out throughoutly. If all is good – move on.

Now based on your exeperiences during the update process on testsite, since you probably encountered some issues in there, make sure to have a complete and proper checklist pre-prepared with notes on how to solve potential issues. If there are any bugs or usage of deprecated functions, you should check those closely. It's important to fix all "deprecated" notices. It is information for you, that in the next PHP versions functions will not be present – and the website will simply stop working. Your notes can help you A LOT during updating in production – believe me, I know what I'm saying!

If during the update process, you'll need to update domain settings aswell, be sure that before planned update at least a week before you will reduce TTL settings. This

will ensure that during changes in DNS records setting will be propagated with best possible speed wolrdwide.

Of course, after the update. even if you think everything went successful, you should test out everything yourself. The whole process that your users are going through, cron jobs, sending emails, payments, etc. Also, you should check out if your issue monitoring software is working well and accepting new issues, so when anything is triggered by the user that was not covered by your tests or ommited, you will know immediately. This is one of basic software pieces we are using at Sailing Byte at regular basis.

If you cover all of the above points, you should be good with the update.

Update Order for WordPress and PHP

Remember that you can always find information on supported php versions on official wordpress.org website. But for your convenience, I have recreated the compatibility table below.

WordPress	PHP 8.4	PHP 8.3	PHP 8.2	PHP 8.1	PHP 8.0	PHP 7.4	PHP 7.3	PHP 7.2	PHP 7.1	PHP 7.0	PHP 5.6
6.7	Yes, Beta	Yes, Beta	Yes, Exceptions	Yes, Exceptions	Yes, Exceptions	Yes	Yes	Yes	No	No	No
6.6	No	Yes*	Yes, Exceptions	Yes, Exceptions	Yes, Exceptions	Yes	Yes	Yes	No	No	No
6.5	No	Yes, Beta	Yes, Beta	Yes, Exceptions	Yes, Exceptions	Yes	Yes	Yes	Yes	Yes	No
6.4	No	Yes, Beta	Yes, Beta	Yes, Exceptions	Yes, Exceptions	Yes	Yes	Yes	Yes	Yes	No
6.3	No	No	Yes, Beta	Yes, Exceptions	Yes, Exceptions	Yes	Yes	Yes	Yes	Yes	No
6.2	No	No	Yes, Beta	Yes, Beta	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
6.1	No	No	Yes, Beta	Yes, Beta	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
6.0	No	No	No	Yes, Beta	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
5.9	No	No	No	Yes, Beta	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
5.8	No	No	No	No	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
5.7	No	No	No	No	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes
5.6	No	No	No	No	Yes, Beta	Yes	Yes	Yes	Yes	Yes	Yes

WordPress	PHP 8.4	PHP 8.3	PHP 8.2	PHP 8.1	PHP 8.0	PHP 7.4	PHP 7.3	PHP 7.2	PHP 7.1	PHP 7.0	PHP 5.6
5.5	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
5.4	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
5.3	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes

This table should be helpful for you to craft the planned update process even if you are using very old versions of PHP and WordPress. Your sample path could look like this if you are using wordpress version. 5.5 and php version 7.4:

1. update WordPresspress, for example to version 6.0
2. Increase your PHP version to 8.0 or 8.1
3. Update WordPress to 6.6 or even 6.7
4. Attempt to use PHP 8.3 or 8.4

Update Laravel and it’s PHP Compatibility

Again, for your convenience, I have recreated table based on Laravel documentation so you can see yourself what PHP versions are supported by which Laravel versions.

Laravel Version	PHP 5.6	PHP 7.0	PHP 7.1	PHP 7.2	PHP 7.3	PHP 7.4	PHP 8.0	PHP 8.1	PHP 8.2	PHP 8.3	PHP 8.4
Laravel 5.0	Yes	Yes									
Laravel 5.1	Yes	Yes									
Laravel 5.2	Yes	Yes									
Laravel 5.3	Yes	Yes									
Laravel 5.4	Yes	Yes	Yes								
Laravel 5.5		Yes	Yes								
Laravel 5.6			Yes	Yes							
Laravel 5.7			Yes	Yes							
Laravel 5.8			Yes	Yes	Yes						
Laravel 6.x				Yes	Yes	Yes					
Laravel 7.x				Yes	Yes	Yes					
Laravel 8.x					Yes	Yes	Yes				
Laravel 9.x							Yes	Yes	Yes		
Laravel 10.x								Yes	Yes	Yes	

Laravel 11.x									Yes	Yes	Yes
Laravel 12.x									Yes	Yes	Yes

You should read this table in similar way that we were reading WordPress update table. So essentially update alternatingly Laraval and PHP. Just be sure that you are using compatible versions. Of course, when updating Laravel to the next version, you should always follow the official checklist. In a similar way, you should follow PHP update checklist between each version.

Updating – is it Worth It?

It should be no surprise if I say that updating is not only worth it, but also necessary. But if you understood the article that should be no surprise to you.

Remember, though that here I described everything only in PHP perspective, but there can be many more dependencies. For WordPress that could be plugins. For front end that could be for example React JS packages or JQuery libraries and for the database it could be MySQL or MariaDB versions. Not to mention that you should also take care of updating your linux distro repository packages – so there is quite a lot to cover.

Of course, this might be a little bit too much information to process for non technical person so it’s a good time to ask a person that takes care of your website if you can update the PHP version – preferably to newest one. Also ensure that you have already implemented update process in your business procedures. But if you do not have such a person – do not hesitate to reach us via the contact form below and let’s start working together!