Why Elixir?

Elixir is a modern programming language for the pragmatic Web Engineer who wants to be ready to face tomorrow's challenges.

At RESTAR, we love how Elixir enables our team to build great features and a robust tech platform, while enjoying top class productivity and developer experience. Thanks to Elixir, we can focus on what matters: the business logic, with very little ceremony and bikeshedding.

Let's review quickly what makes Elixir the perfect tech choice for us:

- Productivity: outstanding ecosystem and developer experience
- Reliability: fault tolerance taken to the next level
- Simplicity: do more with less
- Performance without trade-offs

Productivity: outstanding ecosystem and developer experience

Elixir sure is a nice language, but a language does not exist in a vacuum. In order to evaluate it, we need to look at the bigger picture: the tooling and the ecosystem. Despite its relative youth and the fact that obviously there are fewer packages than on more mainstream ecosystems, Elixir really shines here by providing a very high quality ecosystem. First, Elixir itself ships with: a rich standard library, the mix build tool, the ExUnit testing framework, a formatter, documentation, … all of incredibly high quality. Then the rest of the ecosystem, especially the Phoenix framework, provides many amazing libraries, often as polished, well documented and well-thought as Elixir itself. You might not find the same quantity as in other ecosystems, but it makes up for it in quality!

Another great thing about the Elixir community is the overall maturity and pragmatism: people tend to follow guidelines, and there is often a blessed way to do things.

Reliability: fault tolerance taken to the next level

Elixir might be relatively new (10 years), but the underlying Erlang VM is more than 30 years old and is as battle tested as it can be. Far from being a fancy exotic language, Elixir is leveraging solid technology which has been proven in the telecom industry for decades. One of the main selling points of Erlang is: **fault tolerance**. The whole design of the VM has been driven by this need, crucial in the telecom business, and the result is an incredibly robust platform that isolates failures like no other and can remain up and stable under significant loads. Forget about panics, crashing servers or stuck event loops.To know more about this topic, we recommend Saša Jurić's awesome talk: [The Soul of Erlang and Elixir](https://www.youtube.com/watch?v=JvBT4XBdoUE) (https://www.youtube.com/watch?v=JvBT4XBdoUE).

Simplicity: do more with less

Overall, thanks to all the batteries included in Elixir itself and in Phoenix, you can accomplish more with fewer moving parts as you would need in a more classic stack. We believe that there is a huge value in simplifying our stack:

- As developers, we are constantly fighting against complexity. And yet more often than not, our tools are in the way, and are introducing a lot of additional complexity. This is rarely the case in Elixir: keeping it simple makes our lives easier!
- Having fewer dependencies also means fewer possible root causes for errors and an increased stability (remember leftpad)
- It reduces the maintenance overhead

Erlang, Elixir and Phoenix provide powerful building blocks: processes, tasks/agents, channels… These help keep a simple mental model to reason about hard problems such as concurrency, real-time systems (websockets have never been that simple!), and fault-tolerance.

Performance without trade-offs

The original version of our platform at RESTAR was written in Ruby, and was suffering performance issues. While it would have been technically possible to keep optimizing it and/or buy bigger servers, the amount of effort needed was just too important. In Elixir we seldom have to worry about performance at all: it just works ™ . Instead of having to worry about scalability, server costs, or spending precious time on performance tweaks, you can just focus on what matters: adding business value.

That being said, if performance was the only criteria we cared about, we would all be coding in C or assembly. The great thing about Elixir is that it delivers great performance out of the box without needing you to be careful or to sacrifice anything in exchange: no brittle async/await and event loop, no need to manage thread pools or coroutine contexts by yourself, no need for complex caching… You still get to work with a very expressive, high-level, declarative language. It feels like having your cake and eating it!

Conclusion

Elixir has turned out to be an amazing choice for our team and we enjoy working with it a lot. If you are interested, we hope you will join the adventure and have fun with us at RESTAR!