Ask HN: What would be your stack if you are building an MVP today?

459 points by nvln 22 days ago | hide | past | favorite | 723 comments

Specifically the backend. I'd love to hear your reasons. Do you keep one eye on what the stack would be post MVP?

1. Old schoolish (VPS - Maybe DO, Django/Flask/Rails/Remix/Next with postgres)

2. Supabase etc with JS/TS on either side of the network

3. Lambda/Cloud functions with Firebase/Dyanamo DB/Cosmos DB etc

4..n. What else and why?

nyxcharon 22 days ago | next [–]

After working with the PETAL (Phoenix, Elixir, TailwindCSS, Alpine.Js, Liveview) stack at $JOB for a while now, I have to say I've never been more productive.

Early on I would still have to lean on Alpine heavily for various JS interactions but with all the features Liveview has been adding and improving on (Hello JS module!) I find myself needing it less and less. Liveview really has been a game changer for me. Tailwind has more or less fixed most of my frustrations with CSS and has worked itself nicely into the company design system resulting in nice re-usable components that can be easily customized for those one-offs. Elixir is a nice language to use and I find myself missing features of it when working with other languages. Phoenix is well structured for projects and the newer generators solve a lot of common issues/features in projects like user auth in a reasonable way.

Depending on the needs of the app I'd look into deploying to Fly.io otherwise I've been using Kubernetes with little issue. I've toyed with Nomad for orchestration but haven't used it enough to give an opinion. I do wander if it's a nice middle ground for those who don't need the full suite of what K8s offers.

> davehcker 22 days ago | parent | next [–]
>
> I second this. I would say that I am pretty advanced with Python (and Django) and same with JavaScript (Vue and Nuxt) and have written applications that got used by multiple users. I saw a sharp rise in my productivity when I knew enough about the frameworks. But Elixir + PhoenixLiveView + Tailwind has been life changing.
>
> I learned Elixir for the sake of the joy of learning a new programming language and I kept playing with it for few random days over 5-6 months. Finally, I took the leap of faith and for our startup I started the switch to Elixir + LiveView with minimal JavaScript hooks and I feel a weird bliss that we are two engineer FTEs and I can add features on a daily basis. Why that's the case, I still haven't ruminated myself, but my guess is 1. I have gotten older, 2. Elixir is beautiful and productive by design-- pattern matching, everything is a process (so the dimensionality of time is not an issue at all) and the code is kind of a right balance of simplicity and complexity, and in my opinionated view, there is "one" right way of doing things. 3. Standard tooling (mix, ExUnit). They have enabled us to write really maintainable code and for our next hires, we are willing to pay for them to learn Elixir than switching to other languages. Of course this is only for true for our web app which is actually a weird beast that interfaces farms, sensors, algorithms, and humans.
>
> > kodah 22 days ago | root | parent | next [–]
> >
> > Awesome experience write up.
> >
> > Question for you and OP, do you find that using languages like Elixir makes it so you have to look for a *certain kind of dev* given that Elixir is a pretty niche community or are folks with no experience able to adapt with relative ease?
> >
> > > nelsonic 22 days ago | root | parent | next [–]
> > >
> > > Not one of the OPs but can happily echo their experience/praise for PETAL and think I can answer your dev search question:
> > >
> > > Anyone with a can-do attitude can learn Elixir. People with a “no thank you, I already know XYZ” mindset you *really* don’t want on your team.
> > >
> > > Almost 20 years ago @pg wrote the “Python Paradox” post: www.paulgraham.com/pypar.html
> > >
> > > The same holds true for Elixir (and Rust) these days. The people you want to hire are those who learn new skills/languages proactively.
> > >
> > > However I can confidently say, from experience of having hired several Python devs/data scientists with zero Elixir experience, that smart+motivated people can learn Elixir in a couple of days and be productive within a week. See: https://www.verytechnology.com/iot-insights/elixir-for-pytho...
> > >
> > > > john_the_writer 21 days ago | root | parent | next [–]
> > > >
> > > > I'd argue the other way. Yes anyone with a can-do can learn Elixir. But there's a difference between learn and LEARN..
> > > >
> > > > I work with a pile of rails devs writing code in elixir. In that I mean you can feel the rails in their code. It "tastes" like rails. A few python devs also write python with elixir syntax.

Finding people who know the language is hard. Finding people who are willing to risk their time on the language is also hard.

By risk. If I spend 2 years writing Elixir and then go looking for a job I'm looking at getting a job with a language that's not popular. So my pool of place that will even look at me is smaller. Hiring managers/gate keepers might not see..

After 12 years of c++ I got a job where I needed to code in Delphi-Script for a year. When I told recruiters what I'd been doing I got "Oh.... Okay um... We'll call you."

I switched to rails for 6 months (same job) and recruiters were calling me non-stop. Working in low popularity langs limits my options. (even if they are great langs)

I'd say picking a language that's not yet popular limits your pool not just to the can-do's, but also to the can get my next job with this on my CV.

pcthrowaway 21 days ago | root | parent | next [–]

Isn't Elixir a little bit like Ruby though? At least superficially. And it supports metaprogramming like Ruby.

So I'd guess the code would end up looking similar on a lot of teams

john_the_writer 21 days ago | root | parent | next [–]

It's a little bit like ruby.. But

It's not object oriented, it's a functional language. It is polymorphic and all data is immutable. (Edit: actually realised its very much not like ruby, when I re-read this)

What I've seen in most projects is that the awesomeness that is threading (tasks and processes) is not really used. Supervisors aren't built as first class citizens.

I see teams eat the cost of NIH, but not get the benefits of crash fast, crash often.

So yes it does end up looking/smelling like ruby/rails but it really really shouldn't.

“A language that doesn't affect the way you think about programming, is not worth knowing.”

doetoe 21 days ago | root | parent | next [–]

I like the quote at the end, I would change knowing to learning though

slindz 21 days ago | root | parent | prev | next [–]

There is development joy found in both, but Elixir and the underlying erlang definitely has it's own idioms.

You can know enough to get going quickly, but there's a healthy layer of power that will only come with time.

john_the_writer 21 days ago | root | parent | next [–]

Took me a year before I found a dev/manager that even scratched at that power. A dev that pushed me to learn it. Before that I was a rails dev writing elixir. :)

I should mention there is no problem not going there. I respect my teammates. Ruby/Rails is a great way to get things done. I would 100% use it as my MVP lang.

kodah 21 days ago | root | parent | prev | next [–]

That nailed what I wanted to know. Thanks!

davehcker 21 days ago | root | parent | prev | next [–]

I didn't want to confess, but yes you're right. Elixir (or the willingness to learn Elixir) is a great filter for me to find the right people. Again, this is my purely subjective opinion and might as well be true for other languages as well.

nyxcharon 21 days ago | root | parent | prev | next [–]

I'll more or less echo what nelsonic said.

No one where I work has come in with Elixir experience. We look for people that are motivated to learn. Elixir has been one of the easier languages in my career to learn, and I think others that I work with would agree with that.

Dowwie 21 days ago | root | parent | next [–]

The degree of difficulty with learning Elixir/OTP depends on what one needs to learn. Phoenix web development has its own domain of learning requirements where as other domains have theirs. Learning Elixir is one thing but learning OTP, and then learning how to make sound decisions about design/architecture, is far more demanding of effort and time. "How do others solve this?" leading to github search tends to fall far short of what one would find with other language ecosystems.

eddsh1994 22 days ago | root | parent | prev | next [–]

Moving from Python/JS -> Elixir at a startup seems very high risk, have the rewards been worth it? I can imagine issues with hiring, unknown unknowns, and less libraries/support in general.

davehcker 22 days ago | root | parent | next [–]

I know-- I was scared like hell. But after the first week and multiple staging releases in less than 10 days, I was on top of it. Yes, hiring is real challenge and am facing it already. However, if I, even as a startup, reach the salary threshold, then hiring is not a problem. I'd say I've yet not found a case where I couldn't find a library for my use case.

innocentoldguy 21 days ago | root | parent | prev | next [–]

The start-up I work for uses Phoenix and Elixir. I've been there around two years now and we haven't experienced any issues with hiring or finding skilled people. If anything, I'd point to Elixir as being one of the major factors of our success.

We are a remote company, so things may not be as easy, depending on your location, if your company demands people be in the office.

scwoodal 22 days ago | parent | prev | next [–]

I also would choose Elixir/Phoenix. Having spent my career in Python/Django, Elixir/Phoenix/Live View is on another level of productivity.

For anyone needing a reason to jump into Elixir, take 40 minutes and watch this talk [1] by Sasa Juric. He communicates things so well as to all the benefits of Elixir/Erlang/BEAM.

Then take the time to read Elixir in Action book (again, Sasa) and you'll be off to the races.

[1] https://www.youtube.com/watch?v=JvBT4XBdoUE

rozap 22 days ago | parent | prev | next [–]

I also went this route, but am keeping the infrastructure simple in order to minimize lock in. It has been super productive but also I've spent years using elixir professionally and unprofessionally so I'm comfortable with it. The nice thing is that the elixir ecosystem is so robust you don't need to get locked into whatever cloud provider solution is in vogue at the moment. Also, we got a pile of cloud credits from different providers, and intend to use them :)

innocentoldguy 21 days ago | parent | prev | next [–]

I too would reach for the PETAL stack you mentioned. I don't code professionally anymore, but when I did I had experience using a lot of different languages (e.g., C#, Java, PHP, Ruby, Elm, Python, Perl, Lisp, Node, Clojure, F#, etc.). Elixir is by far the most productive and joyful language I've ever experienced. It's absolutely beautiful.

afhammad 21 days ago | parent | prev | next [–]

Adding my vote here. Throughout my career I've gone the C#.NET -> RoR -> Clojure -> Elixir/Phoenix route. After spending a year with Elixir, iterating rapidly on a production system, I'd default to it for any MVP or serious project going forward.

LiveView is also very close to becoming default over React for interactive Frontends.

Haegin 21 days ago | parent | prev | next [–]

We use Nomad at $JOB and it's been great. We're a small startup with one senior site reliability engineer doing pretty much all the infra work and I (~CTO) chip in when I can be useful, though when we adopted Nomad it was just me working on infra.

I looked into Kubernetes (I haven't worked with it at all before) and there just seemed so much complexity to handle things I didn't need that it was too much to learn for the benefit. Once I found Nomad it was a lot simpler, but it has everything we need and that's stayed true as we've grown.

I'd definitely strongly recommend it if you're wanting an orchestration system and don't need the complexity Kubernetes brings.

mxgrn 21 days ago | parent | prev | next [–]

For deployment I'm using Docker + Github CI + Digital Ocean basic droplets. Docker support is built into Phoenix, all you need is `mix phx.gen.release --docker`.

What I like about it is that upgrading a DO host is basically a matter of spinning off a new droplet and install Docker on it. Also, setting up deployment for a new Phoenix project takes about 30 minutes now.

bratsche 21 days ago | parent | prev | next [–]

I feel similarly, except for the AlpineJS part. I think PETL is fantastic and I would 100% choose that for my stack.

Abishek_Muthian 20 days ago | parent | prev | next [–]

Is Liveview CSP compatible? I found out that Alpine isn't and the alternate CSP build of it isn't stable.

arnonejoe 21 days ago | parent | prev | next [–]

I'll second this. Alpine.js is so nice.

gianpaj 20 days ago | parent | prev | next [–]

Fly.io is great!

jrvarela56 22 days ago | prev | next [–]

Personally big on Rails (use it at work every day). But, my last MVP I did with TS/Next/Mantine/Supabase/Vercel.

Reasons:

- I've been using Rails as an API only, so having to grok views felt like a waste. My React skills made me feel that learning how to make complex views in my backend was a waste of time.

- One type of bug I hate is ensuring my API calls have the right schema. With Supabase and TS in the frontend, this is a non-issue: export types from db schema and all frontend requests are typed.

- Supabase is something for sure: I had been eyeing it for a while now, and it did not disappoint. Getting endpoints - even with realtime updates/subs in the frontend - after just defining a schema is magical.

- Auth is super simple with RLS - all this uses my SQL knowledge and makes me double down on it, which seems like a sane choice given it's been around for decades.

- Mantine gives me a large list of prebuilt components. This isn't exclusive to this stack but would highly recommend using something like this instead of just Bootstrap or Tailwind on its own.

Overall I'm happy with the results, feels way faster than my React/Rails workflow. Way less time in the backend (models, migrations, presenters, etc), more focus on tweaking the product.

Con: does not move me in the direction of using Python more. Feel like this is going to be a must-have for AI-powered components.

> bottlepalm 22 days ago | parent | next [–]
>
> Same, Next.js, Vercel, Prisma, Supabase. I push to my GitHub repo and the site is live and deployed in less than a minute.
>
> With the stack you can use the same language, DTOs, libraries, etc.. both server and client side.
>
> Server side rendering ensures a graceful handoff between server rendered state and the resulting app state on the client.
>
> I'm also using MUI which includes all the components you need for a front end app.
>
> I also just really like react/JSX, very strongly typed coupling between the html and JavaScript. Makes it very easy to refactor and pull out new components with confidence.
>
> I really think this is the best stack for an MVP CRUD app right now. Interested to hear of anything better.
>
> > 0xblinq 22 days ago | root | parent | next [–]
> >
> > This stack is pretty incomplete for most MVPs, as unless you're building something trivial like a landing page, you'll probably at some point also need libraries or a custom implementation for:
> >
> > - Validations
> >
> > - Translations
> >
> > - Error and request logging and auditing
> >
> > - Security (CQRS, CORS, CSRF)
> >
> > - Permissions and a way to integrate it with your authentication (does supabase handle this? don't know)
> >
> > - Email sending
> >
> > - Background jobs
> >
> > - Caching
> >
> > - File uploading
> >
> > - Rate limiting
> >
> > - ...
> >
> > Every time somebody suggests "just" Next.js, just "Remix", just "flask" or just [insert minimalist framework] I think they're either building something pretty trivial, or wasting a lot of time on rebuilding or integrating things you get for free with batteries included frameworks.
> >
> > > kiwicopple 22 days ago | root | parent | next [–]
> > >
> > > (supabase CEO)
> > >
> > > We wouldn't be the best fit for all of these, but here are some:
> > >
> > > - Validations: handled already by Postgres and if you use JSONB you can use pg_jsonschema[0]
> > >
> > > - Translations: probably won't ever be part of our stack, but for multilingual FTS we offer the pgroonga extension[1]
> > >
> > > - Permissions: we offer AuthN (email & social logins, MFA, SSO) [2] and AuthZ with Postgres RLS[3]
> > >
> > > - Background jobs: pg_cron [4]
> > >
> > > - Caching: For images we have a smart cache [5]. We'll extend this to database queries over time
> > >
> > > - File uploading: supabase storage [6]

[0] pg_jsonschema: https://supabase.com/blog/pg-jsonschema-a-postgres-extension...

[1] pgroonga: https://pgroonga.github.io/

[2] AuthN: https://supabase.com/docs/guides/auth/overview

[3] AuthZ with Postgres RLS: https://supabase.com/docs/guides/auth/row-level-security

[4] pg_cron: https://supabase.com/docs/guides/database/extensions/pgcron

[5] smart image cache: https://supabase.com/blog/storage-image-resizing-smart-cdn#s...

[6] supabase storage: https://supabase.com/storage

jrvarela56 22 days ago | root | parent | prev | next [–]

I mean, yeah you need more libraries, but I wouldn't list Formik/mobx/react-i18n/Sendgrid when talking at this level of abstraction.

Rastonbury 22 days ago | root | parent | prev | next [–]

That stuff isn't "hard" per se, and some of that stuff isn't even needed for a MVP, they're good to haves or only really required when you've got 100s of users (product dependant of course). If the MVP doesn't get there either iterate product or bin it

0xblinq 22 days ago | root | parent | next [–]

None of these are difficult per se. Multiple of them, working seamlessly together, having them tested, documented and battle tested is a bit more difficult.

Then you think about all that time doing this could have been spent in actual business logic and it makes no sense to me anymore.

gardenhedge 21 days ago | root | parent | prev | next [–]

The post is literally about building an MVP. Remix isn't a minimalist framework - it's a full stack web framework.

moneywoes 20 days ago | root | parent | prev | next [–]

Maybe next js instead of supabase?

burmecia 17 days ago | root | parent | next [–]

Next.js is more about frontend, but Supabase is more on backend. I cannot see any conflicts of the two, they can work with each other flawlessly.

moneywoes 15 days ago | root | parent | next [–]

Meant to say Nest.js sorry

cmbothwell 22 days ago | root | parent | prev | next [–]

This is a really nice stack and what I'm using together with a colleague right now.

Do you use the serverless functions available with Vercel/Supabase? If so, I'm curious how you choose when to go with each provider.

jrvarela56 22 days ago | root | parent | next [–]

I use vercel functions for this side-project bc they worked out of the box. Just least friction tbh not sure if optimal.

jrvarela56 22 days ago | root | parent | prev | next [–]

Ah you comment about MUI reminded me to include Mantine, also big productivity driver.

turbobooster 22 days ago | root | parent | prev | next [–]

Wish I could afford vercel

fullstackchris 22 days ago | root | parent | next [–]

Super generous free similar product is netlify... you get 300 build minutes per month, I've never even gone above their free tier haha

admn2 22 days ago | root | parent | next [–]

Unless something has changed, they explicitly disallow any website that facilitates for profit commerce. Granted I don't think they closely monitor it unless your bandwidth is over a certain threshold. However, Cloudflare Pages is free and unlimited FWIW but I doesn't seem as nice or fast as Vercel.

bottlepalm 22 days ago | root | parent | prev | next [–]

Remember we're talking about MVP. Moving this stack to a VPS would only add a day.

michaelteter 22 days ago | parent | prev | next [–]

> Con: does not move me in the direction of using Python more. Feel like this is going to be a must-have for AI-powered components.

This is why too many companies build on Python... some thought that maybe someday they will need some AI something, and they hamstring themselves from the start with inferior tooling.

foxandmouse 22 days ago | parent | prev | next [–]

I'm glad I came across your comment, I'm planning a 2-week project for learning purposes and I thought rails backend with react/ svelte would be the fastest way to prototype.

My ruby/ rails experience is very limited compared to my JS experience; I really just started experimenting with rails. I'm blown away by the magic, JS frameworks feel like toys in comparison. I also like that Rails is omakase.

Based on your experience, would you suggest trying rails or going the full JS stack route?

jrvarela56 22 days ago | root | parent | next [–]

I was faced with this decision a month ago and started building my models and views in Rails 7. Late night tinkering with views, then trying out ViewComponent I went 'dude wtf am I doing, these apps are so easy to build in React'... Next.js with Supabase has been a breeze. Setting up db with auth and getting realtime updates and object storage with minimal setup (TS fn calls) was mind-blowing.

Caveats: setting up Supabase auth takes a bit of reading docs, using the js helpers correctly and configuring RLS, but after that it just works.

weaksauce 22 days ago | root | parent | prev | next [–]

imo rails without react and leveraging hotwire is going to be the next big thing in web dev. react is a huge increase in complexity for not much appreciable gain. you are not facebook and don't have facebook's needs. rails is nimble by itself.

JeremyNT 22 days ago | root | parent | next [–]

I'm a fan of this too, and it honestly confuses me that people would do a Rails API with React *for a MVP*.

If you're already doing the backend in rails, it's hardly any more work to just do hotwire with server side rendering at the same time, and you can switch to a SPA later (if you really do need to).

0xblinq 22 days ago | root | parent | next [–]

Most people in this thread are not suggesting a stack for an mvp. They're just listing their favorite stacks, that's it.

There's a comment of somebody saying that just Postgres and Go. How on earth is that a good stack for building an mvp?? You'll have to write a shit ton of code or glue together 10 thousand libraries.

weaksauce 21 days ago | root | parent | prev | next [–]

it's also not a ton more work to pivot and turn the rails app with hotwire into an ios/android app with a minimal amount of work. and yeah you can also sprinkle react in if you need some component or two or even do the api that way and drop into a full SPA if you *really* need that. but at that point you are doubling the amount of work on the frontend for little gain in my view.

I liked the idea of react and have used it but i don't feel like it improves the situation enough for 95% of the sites out there to justify it and the enormous extra work there is in getting state/routing/etc to jive with the backend. I'm hoping strada, if it ever comes out, will be a large improvement to turbo native and at least provide a very compelling alternative react native.

kirso 20 days ago | root | parent | prev | next [–]

I would love to learn more about this. As a complete beginner though I tried to find resources for full-stack Rails and unfortunately there is nothing much there that teaches you the new front-end or shows how to build an MVP.

Compared to that JS ecosystem perhaps has too much :)

weaksauce 19 days ago | root | parent | next [–]

Hotwire is pretty new but there are some guides out there. it's a progressive enhancement thing generally so you build it normally and then sprinkle in the extra stuff.

1000 foot overview: https://boringrails.com/articles/thinking-in-hotwire-progres...

a tutorial: https://www.hotrails.dev/turbo-rails

a video tutorial: https://www.driftingruby.com/episodes/hotwire-introduction

probably a good paid option: https://pragmaticstudio.com/hotwire-rails

and then there is the main site: https://turbo.hotwired.dev/

cheers!

kirso 17 days ago | root | parent | next [–]

Thank you!

insane_dreamer 22 days ago | root | parent | prev | next [–]

I'd stick with rails unless you have a complex front-end where UI responsiveness is of the essential (i.e., highly graphical-interactive). React is many levels up in terms of complexity.

pcthrowaway 21 days ago | root | parent | next [–]

Rails and React are doing completely different things in the stack; I don't understand this argument (having worked with both, but never getting *good* at Rails)

React is purely about managing the frontend and user interactions. Many apps are very heavy on this. For a lot of things I've built, I think Rails would have been overkill. Even for ones that use a database, I find node.js/express/prisma much easier to work with (until you get into migrations). Activerecord's story around migrations is the best I've used.

But for an MVP you don't really need to worry about that.

halfmatthalfcat 22 days ago | parent | prev | next [–]

Mantine is such a good library, especially when compared to Material, Semantic, etc. I've used it with a couple projects and have been very impressed.

thomasqbrady 22 days ago | parent | prev | next [–]

What's RLS? Row Level Security?

kiwicopple 22 days ago | root | parent | next [–]

Correct. We suggest using Postgres Row Level Security if you use the Supabase APIs + Auth: https://supabase.com/docs/guides/auth/row-level-security

You don't have to use RLS if you BYO middleware, but the RLS approach give you the option to do `frontent <-> backend` (vs `frontend <-> middleware <-> backend`)

ultrasounder 22 days ago | prev | next [–]

Definitely old schools. I am building a MVP right now(kinda building my parachute while jumping off the plane)and I went with Django. And here is why. 1. Very vibrant community of devs and time-tested open-source libraries.If you want a multi-tenancy there is a library for that. IF you want stripe integration there is one for that. If you want "fully built out" services, then we have a plethora of free and paid templates. There are folks here(HN User rlawson)who have built many side projects using Django that they have monitized. if that monetization is your goal, then time is money and Django is your friend. 2. Very easy to deploy. This is a Major major requirement for someone like me(dabbling in webdev). Even after the demise of Heroku hobby dynos, there are things like Railway.app which lets you git deploy app, try it out and then scale it if you want. 3. Lastly, Javascript not necessary- I am a "hobby python dev". I don't have the inclination nor the time to learn yet another Javascript framework. I have an idea, know a bit of Python and I want to iterate it on my idea fast and get to the customer ASAP. Meaning, I can deploy a MVP without having to dabble with JS while jumping through all the different web pack configurations. A big win for me. So, "boring stack" definitely.

clint 22 days ago | parent | next [–]

For context I'm someone who spent nearly 20 years doing almost exclusively python dev, attended the first DjangoCon in 2008, ran the Django community blog during its formation heyday back in the mid 2000's and built tons of Django modules, apps, sites etc, have commits on the Project from way-back, tech edited Django books, etc...

Did rails for a short while in 2010-2012 and absolutely hated it then spent 2012-2020 doing mostly Python/Flask dev for huge fintech companies on AWS.

I now use Javascript/React/Nextjs on Firebase/Google Cloud. So much less hassle, tons more libraries and what seems like an exponentially larger ecosystem. There's still weird shit with transpiling typescript to javascript etc... but overall I enjoy JS dev much more these days.

alexalx666 22 days ago | root | parent | next [–]

Be aware that firebase or aws are only good until you have actual traffic. So in many cases you just loose time you spent understanding their managed infra. I would advice to go with Linode. As a bonus you get to lean new techs that are not tied to crazy managed pricing :)

uncommonGeo 20 days ago | root | parent | next [–]

If the traffic to your app is generating a significant Firebase bill then you have probably moved from MVP to successful app territory. Would you rather start managing more infra or focus on keep building features that add value to the app?

afgrant 21 days ago | root | parent | prev | next [–]

I would be interested in hearing of any instance where Firebase costs created financial hardship.

moneywoes 20 days ago | root | parent | prev | next [–]

Any tutorials for handling next js and a private server (like linode)

squidsoup 22 days ago | root | parent | prev | next [–]

As a long time python dev, I'm now not certain why I would use python for a web app, over typescript on Next, other than familiarity. Sharing types between the server and client, and only having to manage _one_ software ecosystem is great.

davepeck 22 days ago | root | parent | next [–]

Some potential reasons:

1. The data layer, if you want a high-quality ORM and migrations

SQLAlchemy + Alembic (or Django ORM + its built-in migrations) are battle-tested systems that I trust will scale to large teams and *not break my data*.

If you're more of the persuasion to write raw SQL, or to just use a SQL query builder, Python is less of a draw these days (although SQLAlchemy's query builder is quite nice and can be used independent of its ORM).

2. Ties to the data science + machine learning universe

If your back-end intersects with these in ways that are not cleanly separable into services, Python might be a good (or the only) option. Even if you *can* cleanly separate, you're effectively committed to managing Python on the back-end.

3. Stability

For good and ill, the JavaScript ecosystem churns far more rapidly.

4. Familiarity

To your point: there's nothing wrong about optimizing for creature comforts and/or velocity from just having done it before and fired the foot-guns.

I agree that using a single language provides an advantage at the API boundary. Curiously, my experience is that most modern javascript frameworks (like NextJS) don't have a lot to say about how to structure this in practice. Maybe that's fine, but I'd love to see some opinions emerge in the ecosystem.

Amongst other things, I typically want to:

- Share types (typescript `interface`s, etc.) between the front-end and back-end (while avoiding accidentally bundling back-end code into the front)

- Have run-time types (via `zod`, `io-ts`, whatever) for (at minimum) API request structures so I can validate my inputs

- Have a story about how API validation failures are shipped back to the client (and how the client exposes them to the users, e.g. error messages when filling out form fields)

- Differentiate between API types and my underlying data model (the `User` in my database is somewhat to very different from the `APIUser` I ship to my client)

In the python universe, Django, Flask, and FastAPI all have well developed opinions about run-time types, validation failures, and API types vs. data models.

vonseel 22 days ago | root | parent | next [–]

On 1 - I've worked at startups using Django ORM and banking companies using MSSQL stored procs. I don't understand why anyone would ever go the stored proc/raw SQL way unless they absolutely needed the performance. And if you do need the performance, you can always dip down to that level from whatever ORM you are using.

Most of the queries I've done using Django can be optimized at the application code level to be fast and efficient without even touching SQL. There were a few instances at the two Django shops I've worked where we optimized one or two queries by writing raw SQL, but the number of times I've seen that in a codebase can be counted on one hand.

The SQL/stored proc method, on the other hand, doesn't come with any way to do migrations or version your database by default (AFAIK). So now you have to write migration code and come up with some kind of versioning system by hand.

Wish I had more experience with SQLAlchemy/Alembic. I've tried using it on little things here and there but since I've always known Django, that's what I usually go with.

speakfreely 22 days ago | root | parent | prev | next [–]

What to you use for ORM/database migrations with Next.js apps?

christiangenco 22 days ago | root | parent | prev | next [–]

I had a similar journey but with Rails and ended up in a similar place.

React+Nextjs on Firebase makes the deployment and scaling steps *so much easier* than on Rails. I'm very conscious of the platform risk of depending so much on Firebase but gosh darn it they've spoiled me.

FlyingSnake 22 days ago | parent | prev | next [–]

It's funny that only in our industry "time tested"/"battle tested" is called old school and frowned upon. No wonder we see software bloat everywhere.

brookst 22 days ago | root | parent | next [–]

Eh, I think experimental physicists and medical imaging also lean towards newer tech.

marcosdumay 22 days ago | root | parent | next [–]

No area of health work tends towards new tech. It's disallowed by regulation.

randomh3r0 22 days ago | root | parent | prev | next [–]

Eh.. you'd be surprised. Medical Imaging isn't as radical as you may think it is (I spent 7 years doing projects for the exploitation of medical images).

brookst 22 days ago | root | parent | next [–]

I've had the pleasure of annual echo cardiograms my whole life, and moved around a lot, and at least my experience has been visible hardware/software/imagine improvements almost annually.

In any event I wasn't saying all medical imaging is state of the art, I was saying it is another field where people prefer newer tech.

randomh3r0 22 days ago | root | parent | next [–]

Absolutely fair. The vendors have gotten much much better tech "in the room" as it were - CT's are lightning fast anymore, US is clearer and can do more on-the-fly analysis than ever before (e.g. highlighting bloodflow dynamically and capturing tons of data in microseconds), etc. Every modality has grown tremendously so my comment wasn't nearly fair enough in acknowledging that.

Downstream is getting better with items such as AI advancements and more sophisticated mechanisms to transfer data between systems both on-prem and cloud-based, however utilization of all of this tend to be stifled a ton by the standard issue of tech moving faster than policy. My frustration/disgruntlement is due to this issue more than anything.

Thanks for the insight and the reminder that my bubble of experience isn't the world - seriously.

incompleteCode 22 days ago | root | parent | prev | next [–]

Anecdotal, but a co-worker's parent works at NHIS and I've heard they've adopted Kubernetes.

ZephyrBlu 21 days ago | root | parent | prev | next [–]

JS frameworks are also battle tested. "old school" is not necessarily a bonus because the common paradigms have shifted.

Using Django with React is not as easy as a full-stack framework like Next.js. Modern tech stacks are now generally moving towards either full-stack JS frameworks like Next.js, Remix, Astro, etc and/or serverless with things like railway.app, fly.io, Cloudflare Workers, Deno Deploy, Supabase, neon.tech, etc.

Many features of "old school" frameworks are becoming redundant or are being replaced by options with better DX.

FlyingSnake 21 days ago | root | parent | next [–]

I do not want to start a flamewar, but most of the things you mentioned Next.js, Remix, Astro, fly.io etc are not even 6 years old. Django is running production ready code since about 2006 and has more shelf life. I was not downvoting JS but in terms of longevity and robustness, Django has that going for it.

cushychicken 22 days ago | parent | prev | next [–]

Did a big migration from Flask to Django for my main side project for all of these reasons earlier this year.

I love Django. It's criminally underrated.

Side project if anyone's interested in peeping it: https://www.fpgajobs.com

thettiguy 21 days ago | root | parent | next [–]

I'd imagine many are curious like me … are you making any money on this? I have a similar Niche job site I'm thinking of

ultrasounder 22 days ago | root | parent | prev | next [–]

Fellow EE here and love what you do! Loved your post on scraping.Btw, I thought you used Flask. Looks like I was wrong.

cushychicken 22 days ago | root | parent | next [–]

Thanks for the kind words. It's probably time for me to revise that a bit, or write another one. I've learned a lot more yet since I wrote that.

I used to use Flask, but I changed over to Django at the end of 2022. Super happy I did. It wasn't painless, but the benefits have been totally worth it.

heavyheavy 21 days ago | root | parent | next [–]

What were the benefits switching over to Django from Flask?

cushychicken 21 days ago | root | parent | next [–]

Batteries included admin console with easily configurable model views.

Built in migrations. Can't possibly overstate how much this has helped ease deployment. Most of the live debug and prod edits I had to do on the older app were due to db migrations not being shipped with the app.

A baked in ORM with lots of backend optimizations like pagination that keep your queries fast, plus convenience functions for model creation and retrieval (long live get_or_create)

There was a learning curve to all of this, and the complexity of all this in the tutorial scared me away at first, but having seen how the other half lives with Flask, where there are no guardrails for anything, I really appreciate having a framework that solves these kinds of challenges for me.

heavyheavy 16 days ago | root | parent | next [–]

Interesting, appreciate the reply. Wouldn't hurt to add it to the repertoire.

cushychicken 21 days ago | root | parent | prev | next [–]

One more big benefit: you can figure out how to do almost anything with Django with the right combination of Google and Stack Overflow searches. Not so for Flask.

Ok, well, maybe there are *some* answers, but it's always like "Install, configure, and integrate this third party plugin". Ain't nobody who's side-hustlin' got time for that.

noveltyaccount 22 days ago | parent | prev | next [–]

IMO for early stage startups, proven tech is almost always a better bet than the hot new thing. Your goal is to capture market share quickly, not wrangle beta tech into place.

readonthegoapp 22 days ago | parent | prev | next [–]

Is there a multitenancy package?

Is it this one?

https://djangopackages.org/grids/g/multi-tenancy/

I know there is a paid one.

ultrasounder 22 days ago | root | parent | next [–]

Yes this is the one. Or one could also use the https://django-tenant-schemas.readthedocs.io/en/latest/. I don't want to plug any paid options for obvious reasons but they are posted here regularly

nicbou 22 days ago | parent | prev | next [–]

Agreed, although I'd really like to throw in type safety, because the lack of it made my life needlessly complicated. Is this feasible with Python?

gmassman 22 days ago | root | parent | next [–]

Type hints in Python are nice, by pale in comparison to truly typed languages. If you want to feel like your language's type system significantly improves your development experience and the safety of your code, don't pick Python.

nicbou 21 days ago | root | parent | next [–]

What would be a nice contender for someone who is otherwise happy with Python? I love that I can use it for darn near anything.

mattficke 22 days ago | root | parent | prev | next [–]

It's not enforced at runtime, but otherwise I feel like typing is in a pretty good place in Python right now. If you're diligent about using Pyright or MyPy as linters it makes life a lot easier.

bm-rf 22 days ago | root | parent | next [–]

Type hints in python also work great with vscode (and probably other IDEs) as it can offer better autocomplete suggestions for things like function parameters.

hooverd 22 days ago | parent | prev | next [–]

The one thing that Django is missing is the ability to attach a User object to a session without a full-fledged RDBMS. I wish it was structured like the Session object which you can in your own backends for.

jsmeaton 21 days ago | root | parent | next [–]

This is unlikely to ever happen though since so much is tied to it like the permissions system and the admin. The ORM itself is central. I wouldn't use Django unless your project depends on an RDBMS.

0cf8612b2e1e 21 days ago | root | parent | next [–]

Even if not using a database, I will always reach for Django over a Flask or FastAPI. Yes, there is more boilerplate when just offering a few memory views, but there is so much ecosystem built around the tooling that the expense is worth it. Just making a form or sending an email from Flask requires a third party library which will never receive the level of maintenance as first-party Django. As an example, the often recommended (still mentioned on the Flask Mega-Tutorial) Flask-Mail package last had a commit in 2014.

jsmeaton 14 days ago | root | parent | next [–]

Fair. I'd probably reach for it too if I had any kind of frontend required, even if that were just HTML and forms. For server-only applications I'd probably go with FastAPI these days and I'm actually looking forward to an all-in-one framework that encompasses FastAPI as the core.

iwebdevfromhome 22 days ago | parent | prev | next [–]

one thing I'd like to see in Django is something similar to InertiaJS [1], just being able to use VueJS for the view layer is amazing and passing data from the backend to the frontend without having to build complex api systems is a godsend. Maybe I haven't done enough research but is there something similar in Django world ?

[1] https://inertiajs.com/

kolanos 22 days ago | root | parent | next [–]

Still a bit raw on the edges, but IDOM [0] looks interesting and has a Django integration [1].

[0]: https://idom-docs.herokuapp.com/docs/index.html [1]: https://github.com/idom-team/django-idom

iwebdevfromhome 22 days ago | root | parent | next [–]

interesting, is this something similar to htmx ?

c17r 16 days ago | root | parent | prev | next [–]

https://github.com/inertiajs/inertia-django

ebalit 21 days ago | root | parent | prev | next [–]

Reactivated [1] looks similar in spirit, but using React on the frontend. I haven't tested it yet but the idea is very interesting.

[1] https://www.reactivated.io/

agumonkey 22 days ago | parent | prev | next [–]

do you still use builtin views/templates ?

ultrasounder 22 days ago | root | parent | next [–]

Yes! Trying to keep it as simple as possible! Might try this whole HTML over the wire after I have made some dough. Not now tho. My customers don’t give a damn about my stack

physicsguy 22 days ago | prev | next [–]

Django, in my previous role we did MVP websites for academics and the speed of getting a site up and running and deployed was just so fast. I’ve never worked with anything else that is as fast, anything like FastAPI or Flask or Express or similar either requires additional libraries to add really basic and common functionality or you have to roll it yourself.

I personally don’t think that if you’re building an MVP you should be worrying that much about how to store users in the database and add RBAC and building a way to add middleware and a storage layer and all of that crap. It’s not worth it, your application is not a special snowflake.

Edit: Should say that with that I mean serving HTML via Django. Usually I would use Bootstrap for layout. Very occasionally I’d add an API for some interactivity but I think for most MVPs interactivity is likely to be a secondary concern until later.

FlyingSnake 22 days ago | parent | next [–]

Came here to say this. I have a side project that I need to spin up quickly and after lots of framework shopping, I settled for Django.

Django and it's batteries included philosophy is perfect for creating full stack applications. The best thing about Django is that almost every problem you face has been answered by the community.

The community tools like DRF/Django-Ninja/Crispy-forms/Django-environ etc are top notch and it's very easy to extend your application.

7/5 would use Django again.

cushychicken 22 days ago | root | parent | next [–]

>The best thing about Django is that almost every problem you face has been answered by the community.

This is something I realized pretty quickly when toying with migrating my side project from Flask to Django last year. Everything I wanted to know how to do, I could google with “Django” as the first term, and figure out how to accomplish. Anything that took longer than 2 mins to figure out how to do was generally a matter of not knowing how to phrase the question.

It’s made me a huge Django advocate.

rlawson 21 days ago | root | parent | prev | next [–]

The Django community is amazing - very friendly, welcoming and a very rich 3rd party app ecosystem.

scrollaway 22 days ago | parent | prev | next [–]

Nowadays, Django has an *amazing* library which is a lot like FastAPI but in the Django ecosystem.

https://django-ninja.rest-framework.com/

It's absolutely wonderful. I would use that in a MVP, and do the F/E in NextJS + Typescript.

physicsguy 22 days ago | root | parent | next [–]

I'm familiar with it, and I don't *love* DRF but the downside with Ninja is that you end up writing a lot more boilerplate IMO. Unless you're strongly optimising for performance (and usually latency and the database are the biggest issues there), you can get away with ModelSerializer and DRF ViewSets a lot of the time for APIs and write very little code.

bastawhiz 22 days ago | parent | prev | next [–]

The biggest benefit to Django, in my opinion, is the orm and admin tooling. Being able to fix things in your app without having to open a repl or open a database tool is absolutely killer.

danudey 22 days ago | root | parent | next [–]

I honestly cannot believe that Rails doesn't have this built in. This was something that was in Django for as long as I can remember; I was looking at it in I think 2006 and it existed at that point in a very similar state to what I've seen lately.

Not a dig on Rails, like "Oh, Rails is so bad they don't even have..."; rather, Rails is extremely capable, and so are Rails devs, so I legitimately don't understand why this isn't there already, 15+ years after Django.

treis 22 days ago | root | parent | next [–]

Rails has Active Admin which is equivalent to what you get with Django. It's just not a core Rails gem but it's one line in your Gemfile to add it.

dieselgate 22 days ago | root | parent | next [–]

Whoah I never knew that - or maybe did but never seriously considered it. It's been my biggest surface-level gripe with Rails compared to Django so thank you very much for bringing it up

Edit: and it' still maintained !

jamie_ca 22 days ago | root | parent | prev | next [–]

There's a handful of very capable and mature admin solutions (activeadmin, rails_admin, administrate, and still people coming up with new ideas, see avo and madmin as newer examples), they just haven't had one become bundled mainline.

Similar situation with authentication, where devise has been a mainstay for over a decade, but it's not always the best fit for everybody so core rails still just has the basic plumbing to build it yourself (with the low-level has_secure_password).

DwnVoteHoneyPot 22 days ago | root | parent | prev | next [–]

To be fair, Django only caught up with Rails in the past 5 years. When I started with Django it didn't have basic stuff that Rails had, such like auto database migrations and the Rails asset pipeline. The database migration stuff really helps when building/iterating. The Django middleware was very finky at the time.

slig 22 days ago | root | parent | next [–]

FWIW, Django has built in migrations since 1.7 which was released in Sep/2014.

jononomo 22 days ago | parent | prev | next [–]

The Django Book is phenomenal. It is just so well-written and so clear that it makes my heart ache: https://django-book.readthedocs.io/en/latest/introduction.ht...

DwnVoteHoneyPot 22 days ago | root | parent | next [–]

Another good one is Tango with Django. https://www.tangowithdjango.com/

iamsanteri 22 days ago | parent | prev | next [–]

I'm trying to learn Django just for this kind of MVP purposes. I also did a small mini project in RoR by following their documentation. Is there anything that is significantly easier to do in RoR vs. Django? Alternatively, is there anything, say, on a larger scale that I can do with RoR that I cannot do in Django if the project grows? These are the questions I've been wrestling with quite a bit now. After learning Vue and React.js and having used and being cognizant of Next, Nuxt, Supabase, Firebase and all that other stuff, I'm looking to become a bit of a better developer by understanding the intricacies of traditional monoliths and how problems are solved on the backend. I'd also like to be able to serve predictions and host ML models in production. Likewise, if my MVPs catch on, I'd like to retain majority of my "IP" by the means of Django or RoR. Any intuition perhaps on my above questions? Thx!

FlyingSnake 22 days ago | root | parent | next [–]

My 2c.

Django shines at creating a nice project scaffold with all the things you need to do rapid development. It has first class support for DB, unit testing, schema migrations, good enough templating system, performant ORM and can be easily debugged. The documentation is top notch and the best thing is the community. Due to its longevity I found that almost all the issues I had were resolved by a simple Google query or visiting the docs.

It's very easy to add functionality like social auth with few lines of code.

I've also not found anything like Django admin in any other framework. I would highly recommended Django for an MVP.

physicsguy 22 days ago | root | parent | prev | next [–]

My view on this is that there is little between them so in that case it's better to look at availability of skills you'd need to take it forwards in the operating region. For e.g. if it's easier to hire Ruby developers then choose Rails.

I see a lot of people talking about Elixir, etc. on this thread, but I'm based in the U.K. and I've never seen a company locally using Elixir so even if it's really cool, it'd be a bad idea for me to choose it absent a very very good reason. If I choose Django then even if I can't find a Django developer, I can likely find people who know Python and the barrier to entry is pretty easy.

nelsonic 22 days ago | root | parent | next [–]

Several companies in the UK use Elixir including the BBC, USWitch and the NHS! The London Elixir Meetup https://www.meetup.com/elixir-london/ has 1,467 members and is great place to meet fellow alchemists and learn interesting things!

physicsguy 22 days ago | root | parent | next [–]

I'm not in London ;)

nelsonic 22 days ago | root | parent | next [–]

Noted. but responding to the "UK" comment; London is often a good guide/proxy for the UK tech scene. But for reference: there are quite a few universities in the UK where students are learning & using Elixir for distributed/embedded computing projects so these people will gradually filter through into the workforce.

Also there are plenty of Ruby/Rails Devs who have embraced Elixir/Phoenix.

But it's a classic network effect problem; adoption drives jobs which drives learning and further adoption. Without a big corporate sponsor like a FAANG co, Elixir/Phoenix doesn't have the mindshare of other languages/platforms.

physicsguy 21 days ago | root | parent | next [–]

In my city, if I restrict it to the "Greater X" area that includes some outlying towns on LinkedIn, I get 7 people that mention Elixir, and 3400 with Python. If I do the same in London, I get 751 people listing Elixir and 113k listing Python.

The scale is just not there to make it worthwhile in evaluating for any company to adopt other than a FAANG or really major employer IMO

iamsanteri 22 days ago | root | parent | prev | next [–]

Thank you for such a great and senseful answer!

mlboss 22 days ago | root | parent | prev | next [–]

If you thinking of anything closely related to ML then python/django is the way to go.

D13Fd 22 days ago | parent | prev | next [–]

This is where I came out a couple of years ago. I love Django and it has worked out perfectly. But I'm far from an expert.

skizm 22 days ago | prev | next [–]

HTML, css, js, (maybe jquery and font-awesome if I want to get fancy) and all static files dumped in an S3 bucket for as long as I can get away with it, then django if I need anything more. Not that I think python/django is the "best", but it is what I know and it's got all the batteries included out of the box. I've never made a sufficiently advanced UI that required react/vue/etc. So I still just stick with html/css/jquery on the front end.

Semi-related side note: I do use TS/react in my 9-5, but the more I use it professionally, the more I'm convinced there is no need for it outside of trying to get a job, IMO.

TekMol 22 days ago | parent | next [–]

Very similar to my approch.

One difference: I am in the process of stripping out jQuery from old projects. So I would not use it in new projects anymore. Plain JS is just awesome these days. And it is easier to reason about event listeners you have set yourself than wondering "Where does this mouseover event come from? Maybe its from jQueries hover function? Wouldn't that rather use mouseenter?".

shpx 22 days ago | parent | prev | next [–]

Instead of an S3 bucket you can use GitHub Pages. You push stuff to the upstream git repo and it gets deployed. It's free (you have to pay for GitHub Pro if you don't want to make the source code publicly available) and comes with Fastly CDN if I recall correctly. You just have to configure your custom domain.

smackeyacky 22 days ago | root | parent | next [–]

I use a GitHub action to sync to an s3 bucket, works extremely well.

my update.yml:

```
name: Update Website
on:
  push:
    branches:
    - production
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@master
    - uses: jakejarvis/s3-sync-action@master
      with:
        args:
      env:
        AWS_S3_BUCKET: ${{ secrets.AWS_S3_BUCKET }}
        AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        AWS_REGION: 'myregion'
        SOURCE_DIR: 'mydir'
```

Corrado 21 days ago | root | parent | next [–]

Just an FYI. You can now use OIDC to authenticate to AWS. This removes the need for hardcoded IDs and KEYs.

clusmore 22 days ago | root | parent | prev | next [–]

In the case of static HTML/JS/CSS, the webserver will make the source publicly available to your browser anyway, so that's not really an issue.

slenk 21 days ago | root | parent | prev | next [–]

Use GH to push to S3 on commit. Done.

mohitgangrade 19 days ago | parent | prev | next [–]

You should look into Cloudflare Pages. I'm using it for all my sites, and it's completely free. Automatic deploys when you push code to GitHub, serverless functions called workers baked in, and built-in analytics. Not to mention, you get high-speed CDN caching for free.

retinaros 21 days ago | parent | prev | next [–]

I wonder why people like s3 buckets. if your website becomes viral it might become expensive

codeonline 21 days ago | root | parent | next [–]

There is an AWS CDN available that can be placed in front of the s3 bucket. Also allows for custom domain names, TLS, globally distributed for low latency ec. https://aws.amazon.com/cloudfront/

sumitb 21 days ago | root | parent | prev | next [–]

True. For static only deployments, I would use Firebase or CloudFlare or Vercel or Netlify etc. You get automatic CDN and generous free quota.

cj 21 days ago | root | parent | prev | next [–]

It's pretty easy and free to slap a CDN like Cloudflare on top of an S3 bucket

MidnightRaver 19 days ago | parent | prev | next [–]

> I've never made a sufficiently advanced UI that required react/vue/etc.

There is no such UI. Users don't want 'advanced' UI.

v3ss0n 22 days ago | prev | next [–]

Hhere is quick survey :

Regarding backend choices:

```
curl "https://news.ycombinator.com/item?id=34530052" | grep -i Django | wc -l
36

curl "https://news.ycombinator.com/item?id=34530052" | grep -i supabase | wc -l
17

curl "https://news.ycombinator.com/item?id=34530052" | grep -i rails | wc -l
28

curl "https://news.ycombinator.com/item?id=34530052" | grep -i node | wc -l
15

curl "https://news.ycombinator.com/item?id=34530052" | grep -i elixir| wc -l
14

curl "https://news.ycombinator.com/item?id=34530052" | grep -i phoenix | wc -l
7

curl "https://news.ycombinator.com/item?id=34530052" | grep -i fastapi | wc -l
7
```

For Frontend Choices :

```
curl "https://news.ycombinator.com/item?id=34530052" | grep -i svelte | wc -l
9
curl "https://news.ycombinator.com/item?id=34530052" | grep -i React | wc -l
34
curl "https://news.ycombinator.com/item?id=34530052" | grep -i htmx | wc -l
15
```

quickthrower2 22 days ago | parent | next [–]

Wow, now curl is doing pretty well in the rankings!

addandsubtract 22 days ago | root | parent | next [–]

C is winning

kayge 21 days ago | root | parent | next [–]

R really ramped up its rankings and seems to be running ahead of the rest ;)

quickthrower2 21 days ago | root | parent | prev | next [–]

. is doing best actually

schrodinger 22 days ago | parent | prev | next [–]

tangent: asked ChatGPT to update your code to use for loops and voila:

Backend Choices:

```
for backend in "Django" "supabase" "rails" "node" "elixir" "phoenix" "fastapi"; do
count=$(curl -s "https://news.ycombinator.com/item?id=34530052" | grep -i "$backend" | wc -l)
echo "$backend: $count"
done

Frontend Choices:

for frontend in "svelte" "React" "htmx"; do
count=$(curl -s "https://news.ycombinator.com/item?id=34530052" | grep -i "$frontend" | wc -l)
echo "$frontend: $count"
done
```

v3ss0n 22 days ago | root | parent | next [–]

Ask chatgpt to collect every possible frontend and backend frameworks form this post.

claar 22 days ago | parent | prev | next [–]

Should include Laravel too, which would also be my answer.

zeeg 22 days ago | root | parent | next [–]

Laravel is just as big as Django and Rails in adoption, definitely should be included

Abishek_Muthian 20 days ago | parent | prev | next [–]

```
curl "https://news.ycombinator.com/item?id=34530052" | grep -i Go | wc -l
891
```

Do you all believe me now when I say Go is an unfortunate name for an otherwise wonderful programming language?

v3ss0n 22 days ago | parent | prev | next [–]

```
curl "https://news.ycombinator.com/item?id=34530052" | grep -i vue | wc -l
14

curl "https://news.ycombinator.com/item?id=34530052" | grep -i laravel | wc -l
10
```

Towaway69 22 days ago | parent | prev | next [–]

But doesn't that ignore the negative comments, e.g. don't use rails! /s

v3ss0n 22 days ago | root | parent | next [–]

for that case we need to build a language model :D

Towaway69 22 days ago | root | parent | next [–]

Grepgpt? ;)

Edit:

apt install grepgpt

Downloading 0% of 5GB #.....

jtwaleson 22 days ago | parent | prev | next [–]

you left out at least vuejs for frontend

v3ss0n 22 days ago | root | parent | next [–]

still collecting frontend parts

tdy721 22 days ago | parent | prev | next [–]

Deno didnt make the list...

dimgl 22 days ago | root | parent | next [–]

For good reason.

heliophobicdude 22 days ago | root | parent | next [–]

Curious! Can you please elaborate?

LeonenTheDK 22 days ago | prev | next [–]

Well that depends a bit on what the MVP needs to be.

Anything web related I'm probably going the Elixir/Phoenix route. I'm no web dev, but this combination makes me actually kind of like doing web-things with how batteries included it is, and the tools it provides to really jump start a project. Then I'd host it "old schoolish" in a VPS or similar since that's what I know best. Might consider being fancy and investigating if Fly is a good fit. If a DB is required I'd use SQLite unless some constraint pushes me into Postgres.

If we're talking command line tool, the correct answer for me would be C# since that's what I know best. The answer I'd want to give though is Go since I want to get deeper into it, but that's not a good business decision.

Desktop application, best I've got in my kit is WinForms with C# and prayers that it never need run on anything but Windows. Maybe Linux with some Wine trickery, but that's not something I've done before either.

mrdoops 22 days ago | parent | next [–]

Agreed Elixir/Phoenix + Tailwind is at the sweet spot of scalability, performance, and bonkers productivity.

Real time? Need to go distributed? Maximize # of features / developer bandwidth? Cutting edge machine learning training & model serving?

Mobile? JSON APIs and GraphQL tooling is great, but https://github.com/liveviewnative is moving fast so this stack almost has the exodia of full stack, one language, and distributed actor model based scalability.

LeonenTheDK 22 days ago | root | parent | next [–]

Honestly I continue to be amazed with each year of developments. The increased robustness of the greater Elixir ecosystem is unreal. The best part to me too is despite how active it all is, progress overall is very measured and sane. It also doesn't develop at such a pace that it feels exhausting to keep track of, despite all the great work.

j-krieger 22 days ago | root | parent | prev | next [–]

What I like about elixir is that it feels like developing with javascript, but you're not delivering megabytes of js to the client

nazka 13 days ago | root | parent | prev | next [–]

> Cutting edge machine learning training & model serving?

I am curious about this. Do you have more infos?

reply

nelsonic 22 days ago | parent | prev | next [–]

Totally agree! For almost all web-based MVPs the PETAL Stack (Phoenix with LiveView & TailwindCSS) deployed to FREE Fly.io is an epic choice for both response times, realtime features, dev speed (time to market). We're currently building our MVP with it and it's fully Open Source so anyone can learn from our journey: https://github.com/dwyl/mvp

stevenhuang 22 days ago | root | parent | next [–]

I really wanted to like LiveView but starting from 0 in the Erlang world meant learning a whole bunch of language semantics and idioms.

In fact I found it perhaps harder to get started than learning Rust, especially because the language server seemed unable to offer as rich completions compared to Rust.

In the end I did manage to understand most of the primitives and concepts in the basic sample app, but it did make me re-evaluate if it is the best choice in terms of technology adoption when it departs from most typical Python/JS/C family style conventions.

nelsonic 22 days ago | root | parent | next [–]

Totally agree that the learning curve can feel steep learning Functional Programming style, Elixir Syntax and Phoenix framework in one go. But if it's any consolation, our company has taken people who only JS or Python and got them fully up-to-speed in less than a week using the open/free tutorials we've written: https://github.com/dwyl/technology-stack HN feedback very much welcome.

stevenhuang 22 days ago | root | parent | next [–]

That looks like a fantastic resource, especially appreciate the update you've provided here https://github.com/dwyl/learn-elixir/issues/102

I'll definitely have to keep at it and play around more, thanks!

veqq 22 days ago | root | parent | prev | next [–]

Thanks for this link, I'll give it a go!

LeonenTheDK 22 days ago | root | parent | prev | next [–]

I can sympathize, it was a huge jump for me coming from C# land, and I see this sentiment repeated a bit for newcomers so it might be the nature of the beast.

I definitely wouldn't recommend it for an MVP if someone isn't already into it, but then I don't think MVPs with tight business requirements are generally a good place to explore new technology anyway. I think there's a lot of room for nuance there though.

danielvaughn 22 days ago | root | parent | prev | next [–]

Never heard of phoenix but sounds interesting. What's the persistence layer of the stack?

finder83 22 days ago | root | parent | next [–]

Typically Postgres

el_nahual 22 days ago | parent | prev | next [–]

LiveView is great but IMHO it's easily abused. "No JS, ever" is a mistake. IMHO it works best when there's very clear rules with where one should use LiveView vs JS.

sph 22 days ago | prev | next [–]

I'm building an MVP solo.

Rust for the core application, Elixir for the backend, frontend, API, data layer. Javascript is pretty much non-existent, all client-side interactivity is done by LiveView.

Packaged into a podman container and deployed to a Hetzner dedicated server. Storage on PostgreSQL, probably the only thing I would rather not have to manage, but honestly single node is perfectly fine for an MVP. Provisioning is done with Terraform and Ansible.

Future expansion (pretty far away, post launch and post paying customers) is add more geo-distributed worker VPSes, talking to the central coordinator via Wireguard link.

Setup is pretty cheap, fast, scales really well, and it's easy to understand. With additional services (mail, monitoring, etc.) should be less than $100/mo all inclusive, and that should be enough for up to ~100 paying customers.

I have no plans of changing any part of this setup for the next decade. I'm too old to want to get golden handcuffed to turn key solutions and cloud products that just introduce complexity and become VERY expensive, and badly documented, past the demo stage.

Started in November, closed alpha release with users next month, launching Q2 2023.

jstx1 22 days ago | parent | next [–]

If the backend, frontend, API and data layer are in Elixir, what's actually written in Rust?

sph 22 days ago | root | parent | next [–]

At the core there is a custom crawler written in Rust, with the goal of eventually releasing it as an open source standalone application.

But everything else, including scheduling the execution of this crawler, and parsing its output, is done on the Elixir monolith.

ElfinTrousers 22 days ago | root | parent | next [–]

Would that be in Rust for performance reasons? Elixir IME isn't as slow as some other popular languages but it's no speed demon either.

sph 22 days ago | root | parent | next [–]

Performance has nothing to do with it. You can imagine the application being a very specialised version of `wget`, and I like CLI system applications to be easy to distribute, with no dependencies and not requiring an entire VM to run.

Erlang/Elixir is good for networked servers. For CLI apps I either choose Go or Rust, and I much prefer the latter. Since I want to make this component open source, I wanted to keep it separate from the rest of the closed source, proprietary Elixir backend.

There is no wrong choice to be honest, this is mine for my startup, and I don't think I need more justification than "I am productive with it for the task at hand."

danudey 22 days ago | root | parent | prev | next [–]

Rust also has the benefit of eliminating a ton of classes of bugs, which is great for the core of a product that connects to the network, fetches untrusted content, parses it, and (presumably) stores it somewhere.

Not that you can't still have bugs, but when you can get C-levels of speed with Python-levels of safety, why not?

ElfinTrousers 21 days ago | root | parent | next [–]

"Python" levels of safety? Let's aim a little higher in life.

0cf8612b2e1e 22 days ago | root | parent | prev | next [–]

A crawler should be limited by the network not CPU. Outside of the language making it easier to handle multiple concurrent connections, I doubt speed would be much of a consideration.

kfrane 22 days ago | root | parent | next [–]

I'm not sure, it might be. If you're not careful, CPU can become bottleneck when you have a 10Gbit network card.

lairv 22 days ago | root | parent | prev | next [–]

How does Elixir interacts with the Rust code ?

sph 22 days ago | root | parent | next [–]

The Rust app is kept as a standalone application. Elixir spawns the process, passes it an internal URL where it can post its output, and forgets about it, since each run might take a few minutes to hours to complete.

beckler 22 days ago | root | parent | prev | next [–]

Via native implemented functions aka NIFs.

0x008 22 days ago | parent | prev | next [–]

But do you code in vim?

TigerTeamX 20 days ago | root | parent | next [–]

Emacs.... Just kidding, of course vim

newaccount2021 22 days ago | parent | prev | next [–]

I love Rust and program in it a lot...but I would never use it for an MVP

Rust is the tool you use when you are done prototyping, not when you are just beginning

earthling8118 22 days ago | root | parent | next [–]

Hard disagree. Rust lets me move much more quickly in many scenarios. I wouldn't want to prototype in Python for example. In my experience I'll spend too much time debugging and dealing with typing issues if i did that. Even with type hints. Having solid tooling that I can trust to help me maintain quality while moving fast and a really good ecosystem makes Rust a good choice

smaddox 22 days ago | root | parent | prev | next [–]

Depends what you're prototyping. Rust is hard to beat for prototyping an interpreter, for example.

atmosx 21 days ago | root | parent | prev | next [–]

Ah… part of the process is having fun. If ppl enjoys rust, then rust is fine :-)

apocalyptic0n3 22 days ago | prev | next [–]

For an MVP, I'd choose what I'm most familiar with and can be fastest with. That would mean:

1. Laravel

2. An Ubuntu VPS in either Digital Ocean or Linode

3. A managed database in one of those services, likely Postgres

That would get me to market the quickest. I have no issues with the application being in Laravel/PHP and after getting to market, I'd work on making the infrastructure scalable. I wouldn't expect overnight success, so a single VPS and managed database would let me keep costs low while I pick up a few customers and then work to handle the scalability in the background (likely AWS with a combination of EC2/ECS/Fargate, RDS, SQS, SES, CodeDeploy, ElastiCache, etc.)

mooreds 22 days ago | parent | next [–]

> For an MVP, I'd choose what I'm most familiar with and can be fastest with.

That is absolutely the takeaway here. Use the technology you are familiar with and/or can be fastest with. MVPs are risky enough, don't add in "oh, gosh, I have to learn the tech (libraries, deployments, monitoring, database access, etc)" as well.

The only case where I'd pick a new tech for an MVP is when there is an existing open source or free project that I could use that would obviously get the project shipped faster by providing extensive pre-built functionality.

For example, I once used Sharetribe https://github.com/sharetribe/sharetribe even though I was only an intermediate ruby programmer because, after time spent evaluating it and other solutions, it had functionality that could get us shipped faster.

From "git init" to our first beta customer was 1 month of time. Then to our first paying customer was 1 more month. One developer (me). To be fair, my co-founder had done a ton of market development before I started coding, so the initial market/feature discovery was done; that's a huge part of any MVP.

sanjayio 21 days ago | parent | prev | next [–]

Why opt for managed db?

apocalyptic0n3 21 days ago | root | parent | next [–]

In my experience, I'm far more likely to run into database bottlenecks early on than web server bottlenecks (largely through hastily and poorly written queries or unchecked Eloquent N+1 problems). I'd opt for the managed DB so that there's limited chance that the database will affect other processes on the webserver (including serving requests, handling background and scheduled jobs, etc.)

It also makes it easier to scale within Linode/Digital Ocean if you find yourself needing to do that before the more complicated AWS infrastructure is ready.

adrianthedev 22 days ago | prev | next [–]

Nothing really beats Rails. Use something like Jumpstart (jumpstartrails.com) and Avo (https://avohq.io) and you scaffold a full consumer-ready app in literally a few hours.

The thing that bugs me the most with Next.JS and the whole JAMStack movement is that, yeah, you get from "git clone" to deployed on Vercel in two minutes, but if you need to create real app features like a sturdy admin, accounts, authorization, proper asset management, CI/CD, it takes a whole lotta time. I'm not even touching the most common app features. I'm speaking from experience, from building a Next.JS app for about a year.

Most Next apps out there are incomplete with the worst freaking user experience. Insert any other JS framework or hosting provider in the place of Next.JS and Vercel.

Of course, this is a generalisation, not all Next apps are that bad.

Using Rails is like a cheat-code in dev term.

lcnPylGDnU4H9OF 22 days ago | parent | next [–]

Thanks for making Avo! I'd been building something similar in the application I'm currently working on and it was nice to have something that already figured out much of the hard stuff to replace my shitty implementation.

adrianthedev 22 days ago | root | parent | next [–]

Great to hear! Let me know if you run into issues.

quickthrower2 21 days ago | parent | prev | next [–]

If you are going to pay for a starter pack, you could get the same sort of thing for NextJS, right?

justsomeuser 22 days ago | prev | next [–]

- Server: Node.js + SQLite

I know JS very well, so writing HTTP handlers is quite fast.

Node runs on V8, which is probably the fastest runtime for dynamic code.

SQLite makes development easier as it's just a file, gives you ACID.

- Frontend: React/Mobx/Tailwind SPA hosted on firebase hosting.

I think the concept of JSX (write your HTML with JS) is good as it gives you a real language instead of a restricted templating DSL.

Tailwind for the fast iteration speed.

Firebase hosting for the simple CLI and fast CDN.

- OS and hosting: Docker running on Google Container Linux

Docker so that the OS level dependencies of my server are defined somewhere.

Container Linux as it auto updates and has all the GCP logging and monitoring built in.

GCP for the incremental disk snapshots for simple backup of the SQLite state.

If I had to scale the service I would add more CPU cores and faster disk. I would also move the parts that need scaling to Go/Rust, and design the code to make use of the cores.

A few principles I use when choosing tech:

- Avoid distributed state (network boundaries) when possible (SQLite instead of SQL server, function calls instead of micro services).

- Use tools for their primary purpose. No shoehorning. Issues arise when you try to use something for what it was not designed exactly for. If you have >3 tools in your stack that you are shoehorning, things are more likely to break in the future.

- Things should still work in 10 years with minimal updates. Lindy effect. Bet on older tools if possible as they are more likely to be around and maintained.

- Good enough vs optimal: stop trying to find the perfect tech. Web tech is sometimes messy and imperfect. Opinion over what is right changes.

unsupp0rted 22 days ago | parent | next [–]

> I think the concept of JSX (write your HTML with JS) is good as it gives you a real language instead of a restricted templating DSL.

I never got this reasoning. In Vue the idea is:

* Write html templates in HTML

* Call javascript from the template to do javascript things (i.e. computed properties and methods)

This has always made the most sense to me, conceptually.

Sure Vue gives you a DSL for looping template elements (v-for) or showing template elements (v-if), but almost all the typical javascript action happens inside typical javascript calls.

nojvek 21 days ago | root | parent | next [–]

JSX shines if you use TS. Mostly because tsx has excellent code completion, type checking and refactoring tools.

It eliminates entire classes of human errors.

ccorcos 22 days ago | parent | prev | next [–]

I'm keen on this approach for my next project.

Can you share any Dockerfiles / scripts you use to get going with this?

justsomeuser 21 days ago | root | parent | next [–]

Sorry I do not, although I have been meaning to publish the "skeleton" template repo's I have locally.

There are two template repos I use: `web-ui` and `server`

The both have a `sh` dir with common commands in .sh files (watch tailwind, watch esbuild, browsersync to serve and live reload during dev).

You config Google Container Linux with a `cloud-config.yaml`, which can take a bit of time to tailor at first, but after that every project uses the same config with small changes. I use Caddy to terminate HTTPS (it will auto generate and renew certs).

If you have a contact I can message you when I put them on Github.

ccorcos 21 days ago | root | parent | next [–]

Interesting. Thanks for the tips. I'm ccorcos@gmail — would appreciate the notification when/if you put on GitHub. Thanks!

jcuenod 22 days ago | parent | prev | next [–]

What do you use for auth?

justsomeuser 21 days ago | root | parent | next [–]

Just a normal create-user/login form. Or Firebase Auth if Google and other sign in's need to be supported.

jcuenod 21 days ago | root | parent | next [–]

In the former case, are you managing signups, password resets...?

justsomeuser 21 days ago | root | parent | next [–]

Yes, I write those parts myself.

logifail 22 days ago | prev | next [–]

> What would be your stack if you are building an MVP today?

Many years ago a CIO told me he didn't really care what technologies I picked for an upcoming MVP backend, as long as I was already comfortable with them and would therefore be productive quickly.

He also told me we would almost certainly need to rewrite everything *at least once*, so not to dwell too long on the initial stuff.

Build the MVP, build it quickly, get it in front of users. It just needs to work, you're not going to get marks, or customers, for backend coding style.

JohnBooty 22 days ago | parent | next [–]

```
He also told me we would almost certainly need to
rewrite everything at least once, so not to dwell too
long on the initial stuff.

Build the MVP, build it quickly, get it in front of users.
It just needs to work, you're not going to get marks, or
customers, for backend coding style.
```

1000% agree. Overengineering things is the death of MVPs. But man this is hard to do in many real world situations.

Stakeholders and managers want you to keep adding incremental features. Never had a situation where they were like "okay, cool MVP, see you in a few months after you tear it down and rebuild it 'for real'."

That "build a fast crappy MVP" mentality takes serious buy in from stakeholders from day one. That mentality might be common in SV and other startup communities but boy is it tough to find elsewhere.

taeric 22 days ago | root | parent | next [–]

Agreed, but it is also easy to see the tension in your second paragraph. You can plan to rewrite everything at least once, but you will never get that actually in the plan.

My only somewhat clear path out of it is to make sure you built it so it will be easy to rewrite. Either do the least that you can, or keep an eye for improvements, but don't necessarily act on them.

throw1234651234 22 days ago | parent | prev | next [–]

That CIO is brilliant. That's the kind of wisdom you get with experience. Always good when stakeholders ask for additional features because the MVP isn't completely throw-away and then my team doesn't know the language. Then they learn the language, but we need to hire new devs and grow, and no one in the market knows the language. I always tell the new hires that we have a lot of learning opportunities - Go, Ruby, Python, Elixir - you name it. All the fresh college grads can't wait to jump in and contribute within 2 hours of getting hired. My grand-grand father always said "If you can't pick up a new language in a day, you aren't a good programmer and don't belong on Hacker News." He also said using C#/Java/Python is "sus booomer ** fr".

noveltyaccount 22 days ago | parent | prev | next [–]

This is excellent advice. The "worse is better" manifesto says similarly:

> The lesson to be learned from this is that it is often undesirable to go for the right thing first. It is better to get half of the right thing available so that it spreads.

https://www.dreamsongs.com/RiseOfWorseIsBetter.html

thecodemonkey 22 days ago | prev | next [–]

I would pick whichever stack that I would be most productive in.

A Laravel app hosted with Laravel Vapor (AWS Lambda) with a MariaDB database. Would allow me to get up and running quickly, at low cost and without having to worry about scaling for a long time.

Using Tailwind and VueJS or AlpineJS for the frontend.

MobileVet 22 days ago | parent | next [–]

This. Your fastest stack is not my fastest stack. If you want to learn the ‘fastest’ frameworks, that is a totally different decision than getting an MVP out the door. That is an educational one… which is totally valid just not in an MVP sense.

The goal of the MVP is ascertaining product market fit, everything else is waste. Use what you know and optimize later. If your MVP can handle 1m calls a second, you have failed (unless it was natively supported by the framework)

rat9988 22 days ago | root | parent | next [–]

I think it's a very reductive view. One cannot try every stack to find its fastest stack. This is why people ask about other people's experiences. Someone might have a better solution, and a convincing argument, so you could try it and become more productive.

xyzzy_plugh 22 days ago | root | parent | next [–]

I have a relatively terrible answer that I would recommend to no one but it works for me. That's the fastest stack for me.

The question wasn't "what is the fastest stack" or "of all stacks which is the fastest for you" but rather akin to "what is the fastest stack *for you*". Which is often the one that you are productive in.

It's almost always not worth learning a new stack to prototype something unless the goal is to learn the new stack.

shireboy 22 days ago | parent | prev | next [–]

That’s usually my take, but I still worry about a few things:

* which stack will still be around in 1/2/5years?

* which stack Will other teammates or future devs be productive in.

I’m still searching for a very light, productive open source stack that is well accepted and if not future proof at least we’ll backed.

RussianCow 22 days ago | root | parent | next [–]

For an MVP, none of those things matter. Just rewrite it in a different stack later if you realize that the one you picked doesn't fit your requirements long-term. Worrying about that stuff before you have users/customers is just a waste of time and energy.

didgetmaster 22 days ago | root | parent | next [–]

But does that really happen? It seems that we have a lot of bloated, buggy, inefficient code out there because it was initially built using something that was 'quick and dirty' for the MVP and was never rewritten properly once it caught on.

I have clients that still use Excel spreadsheets for their database instead of using a real one just because their data was initially stored there and they never changed. New features were added incrementally over time and it became costly to break everything for a complete rewrite. So they limp along forever because management won't let them do it right until it absolutely breaks.

RussianCow 22 days ago | root | parent | next [–]

Yeah, it's definitely a cultural shift from the way most software is built today, but it's a better way to do it in most cases. I was also assuming a startup environment in my comment; established companies can generally afford to do more work up front to make the foundation more robust, and they are (slightly) more likely to have a better idea of what their customers want ahead of time.

But to answer your question more directly, it does happen, it's just uncommon. Where I've seen it done successfully, the rewrites have been piecemeal, not all at once, so that definitely helps with the buy-in factor.

hot_town 22 days ago | parent | prev | next [–]

you should check out https://wasp-lang.dev then. It's probably one of the fastest stacks out there for React + ExpressJS at the moment

verisimilidude 22 days ago | prev | next [–]

It really depends upon the product, right?

Let's assume we're talking about a web-based business app.

If the MVP already requires some complex business logic, then I'd probably choose an old-school stack. I personally prefer Rails. But I'd probably pick .NET. Above all, my metro area is full of .NET people. It'd be much easier to find/hire collaborators in .NET. I'm comfortable and productive enough with .NET on the technical level.

However! If I could get away with just putting up a couple Lambda functions, then that's absolutely the path I'd choose. It's an MVP. Let's get this boat rowing ASAP!

In short, product and market influence the best technical approach.

andyjones11 22 days ago | prev | next [–]

Elixir, Phoenix, LiveView + Postgres

Main reasons are:

- Can build entire app (with SPA-type experience on the FE) in a single codebase

- No need to build any HTTP APIs

- Periodic tasks and managing in-memory state is super easy in Elixir. Eventually you might want to stick state somewhere which survives restarts (eg redis) but for the sake of moving quickly Elixir makes this really easy

- Newer versions of Phoenix/LiveView support html components so building out UIs now feels as nice as some of the tools in the FE ecosystem

> kcartlidge 22 days ago | parent | next [–]
>
> I like Phoenix/LiveView so it's a nice option.
>
> But I'd probably choose Blazor over it as every one of those bullet points also apply to it, it's C# (with which I'm very familiar), and if it takes off there are masses of devs available in most markets as opposed to handfuls currently for Elixir/Phoenix.
>
> Coupled with something like DO droplets and SQLite/Postgres.
>
> > pcthrowaway 21 days ago | root | parent | next [–]
> >
> > I have no experience with either (though I have perused some Elixir docs a while ago). I'd much rather jump into a Phoenix/Elixir project though, just based on what I've heard and know about it.
> >
> > I wouldn't apply to a C# job, I have no interest in it.
> >
> > I think the only disadvantage is really that Elixir devs would be newer to Elixir. But I think the applicant pool would skew towards having broader experience.

> danieka 22 days ago | parent | prev | next [–]
>
> Depending on your use case you could consider using SQLite. For example storing auth info in a central database and then having one database per customer.

aantix 22 days ago | prev | next [–]

I built Call Stacking (https://callstacking.com/), a modern Ruby on Rails debugger, with Jumpstart Pro and 1.5 hours a night over the course of 4 months.

We have four kids, host an exchange student, a new puppy. Our schedules are *full*.

Nothing compares to the productivity of Ruby on Rails. Especially when coupled with a high-end template like Jumpstart Pro (https://jumpstartrails.com/).

> mooreds 22 days ago | parent | next [–]
>
> I used something similar (https://bullettrain.co/) in the past. Have you taken a look at that? Bullet train used to charge but now are apparently open source.
>
> > aantix 22 days ago | root | parent | next [–]
> >
> > I haven't personally used it, but it looks like a really solid choice as well.
> >
> > I know of Andrew Culver through Twitter and he seems like a genuinely nice person.

> adrianthedev 22 days ago | parent | prev | next [–]
>
> Next time check out Avo. It integrates perfectly with Jumpstart (or any Rails app) and helps you create a back-office app in no-time.
>
> https://avohq.io
>
> > aantix 21 days ago | root | parent | next [–]
> >
> > I've checked out Avo several times before. I don't need it quite yet.
> >
> > It looks beautiful Adrian.

> roflyear 22 days ago | parent | prev | next [–]
>
> When I click examples why don't I see code?

sourdesi 22 days ago | prev | next [–]

I've been working on a side project that aims to provide a really simple UX to spin up full-stack web apps with CI and scalable infra by default. The idea is that you would simply enter a domain name you want to purchase and then the tool would do the following:

1. Purchase the domain name using AWS Route53 (perhaps you'd need to setup an IAM role for the website to access your pre existing AWS acct)

2. Setup frontend, API, and CDK infra repos. All in TypeScript.

3. Frontend react, backend Api gateway backed by lambda functions. Have a graphql endpoint that talks to an Amazon auroradb. CI for backend and frontend is defined by a CDK package that is also deployed to the AWS account that also sets up the networking and database for you.

4. Have a basic layer of application code for setting up user authentication and storing user data in the aurora db.

Basically my idea is it kinda sucks that everyone has to do all of this setup yourself if you want to start a new project. Or you have to rely on no-code tools like Squarespace etc which may not be what most engineers are looking for. Having something that can go from simply entering the domain name you want to scaffolding out a fully functional full stack web app with CI and serverless infra defined by code that can scale from day 1 seems both incredibly useful and doable.

Curious to hear other people's thoughts on this!

> zdwolfe 22 days ago | parent | next [–]
>
> Neat, I am working on something similar with almost all the same tech you mentioned (CDK, R53, TS, APIGW+Lambda), and a Dockerized dev env with everything set up. It's meant to be a boilerplate I copy for new side projects. DM me on twitter (same username) if you'd like to compare notes or collaborate.
>
> > moltar 22 days ago | parent | prev | next [–]
> >
> > Hey sourdesi, this is _exactly_ my stack too, and I am also dabbling with a starter. Would also love to discuss and collaborate. Please send me an email (see profile).

noodle 22 days ago | prev | next [–]

Rails on postgres with React in the frontend is my default swiss army knife. I know Rails well and it still gets stuff done fast, comparatively speaking. If the problem space I work in requires something different or solves the problem better than Rails, I'd switch.

Edit: post-mvp, I'd stick with Rails still unless there are specific difficulties with it. But I've yet to have major issues building a large Rails team/project, including present day.

gls2ro 22 days ago | prev | next [–]

If it is a SaaS 100% Ruby on Rails

Why:

- has almost everything that I need usually in mature gems - battle tested in production

- speed of development

> lta 22 days ago | parent | next [–]
>
> I did so last year, and I cannot be happier. My only regret is that I wasn't aware of htmx/hotwire at the time. I went with Vue, and I regret it immensely.
>
> Rails is the fastest development platform I've tried so far, it is predictable, well crafted, structured yet flexible. You can't go wrong with it.
>
> Every now and then I try something new on a side project or I have to work on some other codebase for my customers, and nothing so far has tempted me to move out of rails for my serious projects
>
> > 0xblinq 22 days ago | root | parent | next [–]
> >
> > I'm using Laravel with Hotwire and it's fantastic. Now I want to cry every time I see the mess of overengineering I have to deal with in other projects when they're built with Redux, React, etc, etc.
> >
> > > omnimus 21 days ago | root | parent | next [–]
> > >
> > > Why Hotwire instead of Livewire if i may ask?
> > >
> > > > 0xblinq 18 days ago | root | parent | next [–]
> > > >
> > > > Mostly due to Turbo transitions between pages and being able to persist elements across pages. I know Livewire V3 will solve this but at the time we took the decision it wasn't available.
> > > >
> > > > Also something brought up was a conversation regarding Alpine vs Stimulus. At first sight Alpine looked a lot easier, but Stimulus seemed it would scale better and be easier to maintain at the end, plus we expected it to have less problems with Turbo than Alpine given those were made to work together.
> > > >
> > > > An important difference we found, is that Stimulus can "react" to value changes (https://stimulus.hotwired.dev/reference/values) while with Alpine it was not clear how a component, or some external code would trigger an update by changing a "prop" or an "attribute" of the element. We know we could use stores and events, etc but that's what I meant with stimulus being easier to scale long term.
> > > >
> > > > > omnimus 18 days ago | root | parent | next [–]
> > > > >
> > > > > Interesting. I had the feeling that framework like that has to be so tied to the backend framework that Hotwire has to have lots of troubles outside Rails. I would have never even thought of trying Hotwire with Laravel. Will try. Thanks
> > > > >
> > > > > > 0xblinq 17 days ago | root | parent | next [–]

Yes, that was my initial impression years ago too.

But it's very progressive. For example, you can just use Turbo to get the "SPA like" navigation between pages (no full page reloads) and that's for free, just including it will bring in that behaviour, plus caching when navigating back, link preloading (so when you click the content it's already there), etc. All of this very easy to control/configure via html data attributes.

Integrating the "Turbo frames" feature is also pretty easy, just wrap content in <turbo-frame> custom tags and the library will do the replacement without page reloads when you submit a form. Similar situation with "Turbo stream", etc.

This is an excerpt from the main documentation site at https://turbo.hotwired.dev/handbook/introduction :

"...You don’t need any backend framework to use Turbo. All the features are built to be used directly, without further abstractions. But if you have the opportunity to use a backend framework that’s integrated with Turbo, you’ll find life a lot simpler. We’ve created a reference implementation for such an integration for Ruby on Rails...."

Same story with Stimulus. You can add it to any framework, it's just a frontend library.

Having said that, there are some "helpers" that you can have in the backend that will make things more idiomatic and avoid some boilerplate, etc. And Laravel being such an amazing framework with such a great community, has a great integration library here: https://github.com/tonysm/turbo-laravel

I couldn't be happier with this stack. With Laravel + Hotwire + Tailwind + Laracasts I feel I'm unstoppable.

mindcrime 22 days ago | root | parent | prev | next [–]

I'm vaguely aware of htmx, but have never used it. Had not heard of hotwire until just now, but that looks *really* interesting. Thanks for the pointer!

Would you care to say any more about your experience with Vue? I've heard a lot of good things about Vue and had it on my mental "things to learn one day" list for a while. Would be really interested in hearing any counter-points from somebody who hasn't enjoyed working with it.

jmuguy 22 days ago | root | parent | next [–]

I'm not sure what their particular beef with Vue is but I've been using it for several years now with Rails and its mostly been great.

They had a similar issue as Python, albeit not nearly as bad, with their upgrade from version 2 to 3. It required some rewriting. They also introduced a new core concept with the "composition" API, which completely changes the way Vue apps are written. Luckily once you made a few required changes you could continue writing apps like you used to, ignoring the new composition API. I'm assuming at some point we'll be forced to change, which will suck.

One thing we do, which I don't think is as common when using Rails with a heavy frontend like Vue or React, is that Rails is *not* in API mode. That is Rails still handles routing, pages are still rendered in ERB, and then each page is its own little Vue app. So we can use Rails, Ruby, etc to initially hydrate the pages and inject stuff by calling to_json on it. This also means we can let Rails and Devise handle login and session, which I absolutely hate doing with a pure JS frontend.

Routing and session management are something I think Rails is extremely good with. This also means that if a section or sections of the site are pretty simple (password reset for instance) we can just render the page in ERB, no JS required.

Regardless, I still like Vue and it has a pretty decent ecosystem. If I had to go back and start again I'd probably just go with React. It clearly has the most support (and most jobs) on the front end.

moxplod 22 days ago | root | parent | prev | next [–]

First I heard of HOTWIRE. I have been using PJAX for a decade now, which is a similar concept.

I use .net mvc with a razor templating engine. But this can be used with any backend.

It makes it super easy for me to maintain all my UX in server-rendered HTML templates. I get a clean SPA with super high development efficiency with MINIMAL javascript.

The best part was I could hire any developer and they know how to work in basic HTML/JS/CSS.

Edit: Reading more, I might need to spend time looking into HTMX/Hotwire as a replacement for PJAX at some point.

rahoulb 22 days ago | root | parent | next [–]

I love Hotwire and it's reinvigorated my love of Rails.

0xblinq 22 days ago | root | parent | prev | next [–]

Check out also Unpoly. It’s probably the most “batteries included” of this kind of tools, although not as popular.

evolve2k 21 days ago | root | parent | next [–]

Can you expand further?

What has you bother to introduce unpoly when Hotwire is included by default? Aka, what are the main things missing/better that drive you to use unpoly.

0xblinq 21 days ago | root | parent | next [–]

I was replying to @moxplod because it seems they didn't know about these alternatives to pjax. They're mentioning HTMX, etc so I've added Unpoly to the list of similar things.

I'm not saying you should replace Hotwire with anything else - I wouldn't do it given it's the default or defacto solution.

But all of these tools (pjax, htmx, unpoly and hotwire) work perfectly well outside of Rails too. So if you're not using Rails and you're, for example, using Django which doesn't have a default solution then you can pick one of them, and I think Unpoly is a very nice one.

kcartlidge 22 days ago | root | parent | prev | next [–]

Worth looking into Hotwire. It works well with .NET MVC.

treis 22 days ago | root | parent | prev | next [–]

IMHO, part of this is that people have accepted the poor experience of non-realtime applications. But there's a huge UX improvement if the application responds faster than users can input actions. Doesn't matter for all applications but if your users are going to spend significant time inputting data it makes a huge difference.

Also IMHO, a flexible type system like Typescript makes development faster than without it. You can refactor faster, it catches silly mistakes, and you don't have to write as many tests.

Rails does a lot of great things for you, but IMHO ultimately it's stuff you don't really need. If there's any chance that the application will grow beyond a few developers I think it hurts more than it helps.

aantix 22 days ago | root | parent | next [–]

>If there's any chance that the application will grow beyond a few developers I think it hurts more than it helps.

Simply not true.

I've consulted on multiple teams across products built with Rails. Products that supported hundreds of millions of requests and generated a similar levels of revenue.

Rails scales - programmer productivity, traffic. It scales.

RangerScience 22 days ago | root | parent | next [–]

One thing I've taken to pointing out, and it seems like you have way more experience with which to back it up - is that it's not just about scaling *to* millions, it's also about scaling *from* zero.

Thoughts? :)

treis 22 days ago | root | parent | prev | next [–]

>Simply not true.

It's not true that I think it hurts more than it helps?

boredtofears 22 days ago | root | parent | prev | next [–]

> part of this is that people have accepted the poor experience of non-realtime applications

my experience has taught me the exact opposite: people have accepted the poor experience of real time applications (client side crashes that bring down the entire page, half-baked routing that is essentially just rebuilding the browser navigator, inconsistent client vs server rendering processes). developers tend to completely stick their head in the sand when these issues are occurring. there's complete classes of problems that simply go away when you're not building an SPA.

> Also IMHO, a flexible type system like Typescript makes development faster than without it. You can refactor faster

in the long run, yes, but in the short term you're probably not going to get your mvp out faster because you chose TS

treis 22 days ago | root | parent | next [–]

> people have accepted the poor experience of real time applications (client side crashes that bring down the entire page, half-baked routing that is essentially just rebuilding the browser navigator, inconsistent client vs server rendering processes).

People complain about that stuff all the time.

>stick their head in the sand when these issues are occurring. there's complete classes of problems that simply go away when you're not building an SPA.

Well, yes. If you don't actually build a real time application you won't get the benefits of a real time application.

>in the long run, yes, but in the short term you're probably not going to get your mvp out faster because you chose TS

Once there's even a moderate level of complexity (i.e. 2-5 devs for 1+ months) the type system helps you catch issues. Even if it only saves you one 4 hour debugging session you come out ahead.

That assumes you sit down and write out your MVP without any significant refactoring. If you do have to make changes then TS will save you significant time there.

It also assumes that you're not writing tests that duplicate what a type system does. If you do write those tests then again I think you're slower.

Rails does have the advantage of there generally being one "Rails way" of doing things. That can short circuit a lot of design discussions and other sorts of bikeshedding. But you can also just not do that and, IMHO, come out better.

boredtofears 21 days ago | root | parent | next [–]

> Well, yes. If you don't actually build a real time application you won't get the benefits of a real time application

A rails application can be just as "real time" as any SPA, I don't think that definition really means anything. You get a distinct set of problems that come with SPA's that don't exist in a traditional server side rendered app that have nothing to do with how "real time" the app feels.

weaksauce 22 days ago | root | parent | prev | next [–]

with hotwire you get the benefit of having that SPA with quick responsiveness but with the much, much reduced complexity in stack with all the jankiness that that brings with it.

sureglymop 22 days ago | root | parent | prev | next [–]

I used svelte and it's been super simple and efficient.

streblo 22 days ago | parent | prev | next [–]

Which gems do you most commonly use?

455 more comments...