● **Web Based**:

○ React

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| High Performance | Steep Learning Curve |
| Reusable Components | JSX Syntax |
| Great Developer Tools | Large Bundle Size |
| Large Community and Ecosystem | Lack of Built-in Features |
| SEO-Friendly | Fragmented Ecosystem |

<https://www.devstringx.com/pros-and-cons-of-using-react>

○ Angular

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Component-based architecture that boosts developer productivity and provides a high quality of code | Divided and churning community |
| TypeScript: better tooling, cleaner code, and higher scalability | Migrating legacy systems from AngularJS to Angular requires time |
| RxJS: efficient, asynchronous programming | Angular is verbose and complex |
| The platform-agnostic philosophy | Hard learning curve |
| High performance ensured by built-in tools |  |
| Google support and documentation |  |
| Angular Material components to create aesthetic UIs |  |
| Seamless updates using Angular CLI |  |
| Powerful ecosystem |  |
| Angular Elements: compatibility with different engineering environments |  |

<https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/>

○ Vue

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Lightweight and Performance | Language barrier (Chinese) |
| Simplicity and Learning Curve | Risk of over flexibility |
| Documentation | Smaller number of components and plugins |
| Simple Integration |  |
| Reactivity |  |

<https://thecodest.co/blog/pros-and-cons-of-vue/>

○ Svelte

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Faster loading times | Smaller community |
| Better performance | Limited resources |
| Easy to learn | The steep learning curve for some |
| Smaller bundle size | Lack of mature tools |

<https://cubettech.com/resources/blog/advantages-of-using-svelte-over-other-frontend-frameworks/>

○ Astro

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| accessible | creating particularly complex or interactive projects such as administration dashboards or social networks, Astro is probably not the ideal framework |
| framework is indeed easy to set up and use | The number of resources, tools and functionalities available certainly cannot yet compete with more famous and established frameworks |
| oriented towards a server-first API design and does not include any JS runtimes. |  |
| comprehensive set of tools to cover all the needs of developers |  |
| includes file-based routing, data retrieval, resource management and much more |  |
| possible to use Astro in conjunction with other popular frameworks |  |
| offers support for Markdown. |  |
| Discord channel is full of information and developers willing to help and provide suggestions and advice. |  |

<https://www.devinterface.com/en/blog/astro-everything-about-framework#tos_2>

○ HTMX

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Lightweight | Responses should be HTML |
| Simple to use | Small community |
| Easy to configure | Lack of pre-made components |

<https://labcodes.com.br/blog/en-us/development/pros-and-cons-for-a-htmx-beginner/>

○ Solid

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

○ Remix

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Dynamic contents | Responsiveness |
| Faster data fetching | Nested routes’ nuisances |
| Simpler code | Problems with not having client-side state |
| More resilient apps |  |
| Smaller bundle size |  |

<https://hackernoon.com/remix-framework-review>

● **Mobile**:

○ React native

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| ONE TEAM – TWO PLATFORMS | DEPENDENCE ON FACEBOOK |
| FASTER DEVELOPMENT | PERFORMANCE HIT WITH COMPLEX INTERACTIONS |
| FULLY NATIVE UI | FEW THIRD-PARTY COMPONENTS |
| NATIVE-LIKE PERFORMANCE | NEED FOR NATIVE DEVELOPERS |
| HOT RELOADING | FEATURE LAG |
| UPDATES WITHOUT APP STORE APPROVALS |  |

<https://www.mindk.com/blog/react-native-pros-and-cons/>

○ Android Java

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Cross-Platform Compatibility | Performance Overheads |
| Rich Ecosystem | Memory Consumption and Usage |
| Significant Community Support | Learning Curve |
| Security |  |

<https://amela.tech/java-pros-and-cons-in-mobile-app-development/#Pros_and_Cons_of_Using_Java_in_Mobile_App_Development>

○ Android Kotlin

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Interoperability with Java Code | Fluctuation in Compilation |
| Easy Maintainability | Less Talent for Hire |
| Boosts Team Efficiency | Limited Learning Resources |
| Kotlin is Much More Reliable | Kotlin is Still Not Java |

<https://medium.com/quick-code/pros-and-cons-of-kotlin-for-android-app-development-c4b0f95c1324>

○ Swift IOS **( BEWARE YOU NEED A MAC AND IPHONE TO WORK IN**

**THIS BECAUSE YOU NEED XCODE)**

● **Cross platform:**

○ Ionic

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Convenient for Developers | Long Debugging of Programs |
| Cross-Platform Compatibility | Lower Performance Compared to Native Apps |
| Single Code Base | Limited Access to Native APIs |
| Wide Range of Interface Elements | Rendering and Compatibility Issues on Different Devices |
| Access to Native Device Features | Risk of Data Leakage and Unauthorized Access |
| Convenience of Testing |  |
| Large Community Contributing to Resources and Knowledge |  |

<https://maybe.works/blogs/pros-and-cons-of-ionic-framework#:~:text=The%20Ionic%20Framework%20is%20a,less%20flexible%20than%20native%20ones>.

○ Flutter

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Single code for different operating platforms | Weighty and large applications |
| Faster time-to-market | Comparatively low adoption of Dart |
| Lower development costs | Issues with iOS |
| Native-like experience | Limited support of third-party libraries |
| Use of widgets |  |
| Strong community |  |
| Well-developed documentation |  |
| High safety |  |
| Automated testing tool set |  |
| Hot reload |  |
| Add-to-app |  |

<https://www.cogniteq.com/blog/flutter-pros-and-cons-it-good-choice-your-app>

○ .Net MAUI

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Write Once, Use Everywhere | Still in Early Days |
| Leverage C# and .NET | Performance Overhead |
| Native Controls and Experiences | Steep Learning Curve |
| Visual Studio Integration | Platform-Specific Needs |
| Open Source | Scarce Resources |
| Cloud Connectivity | Smaller Community |
| Mature Ecosystem | Stability concerns |

<https://www.vthink.co.in/blogs/navigating-net-maui-pros-and-cons-in-living-colour>

○ Kotlin Multi platform

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| The developed app is 100% native for each platform | Many components are still in Alpha/Beta stage and potentially can be unstable or change in future |
| Easy to use |  |
| UI can be split for each target platform |  |
| Shared logic |  |

<https://www.miquido.com/blog/is-kotlin-multiplatform-the-future/>

● **Server side or MVC frameworks** are a hybrid approach where the frontend

and backend are intertwined into one codebase and there is tight coupling,

which has upsides and downsides depending on the situation or context

○ Laravel

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Easy to Learn and Use | Steep Learning Curve |
| Built-in Features | Performance Overhead |
| Blade Templating Engine | Code Bloat |
| Artisan Command-line Interface | Limited Flexibility |
| Database Migrations | Versioning |

<https://arjunamrutiya.medium.com/laravel-pros-and-cons-of-using-the-popular-php-framework-a604a6d4b03f#:~:text=Its%20ease%20of%20use%2C%20comprehensive,choosing%20Laravel%20for%20a%20project>.

○ Django

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Batteries included - Ready for your MVP | Speed |
| Security | Lack of convention |
| Python inside | Not always the right choice |

<https://www.netguru.com/blog/django-pros-and-cons#:~:text=Django%20was%20designed%20to%20deliver,all%20the%20%E2%80%9Cbatteries%E2%80%9D%20included>.

○ Ruby on Rails

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Time efficiency | Shortage of flexibility |
| A great number of helpful tools and libraries | Continuous evolvement |
| Huge and active community | Performance time |
| Strong adherence to standards | Price of a mistake |

<https://sloboda-studio.com/blog/pros-and-cons-of-ruby-on-rails/>

○ Spring MVC Thymeleaf ( Spring has a lot of projects with the word

spring, so the MVC thymeleaf is important)

|  |  |
| --- | --- |
| **Pros** | **Cons** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

○ Golang with Templ

|  |  |
| --- | --- |
| **Pros** | **Cons** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Pure backend**

● Expressjs

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Minimal and Lightweight | Lack of Structure and Convention |
| Easy to Learn and Use | Overhead |
| Middleware Support | Limited Built-in Features |
| Routing | Steep Learning Curve for Middleware |
| Template Engines | Lack of Strong Typing |
| Database Support |  |
| Scalability |  |

<https://data-flair.training/blogs/expressjs-advantages-and-disadvantages/>

● Fastify

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Performance | not widely used in the industry |
| Maintainability | documentation is not as expansive |
| Flexibility | community support is so-so |
| Ease of development | Library such ss passport.js does not work |

<https://tsh.io/blog/fastify-practical-overview/>

● Spring boot

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Rapid Development | Learning Curve |
| Microservices Friendly | Limited Flexibility |
| Spring Ecosystem Integration | Resource Overhead |
| Auto-Configuration |  |
| Embedded Servers |  |

<https://medium.com/@jayeshwarke011/getting-started-with-spring-boot-advantages-disadvantages-and-use-cases-497b0f04fb86#:~:text=In%20conclusion%2C%20Spring%20Boot%20simplifies,and%20customization%20can%20be%20challenging>.

● .NET

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Rich Development Environment | Learning Curve |
| Language Versatility | Vendor Lock-in |
| Rapid Application Development | Stability Issues for New Releases |
| Cross-Platform Development | Development Environment Limitations |
| Scalability and Performance | Memory Leaks |
| Rich Resources and Libraries | Cost of Licensing |
| Enhanced Security and Reliability |  |
| Flexible Deployment and Simplified Maintenance |  |
| Universal .NET Standard |  |
| Thriving Popularity and Supportive Community |  |
| Active Microsoft Support |  |
| Effective Caching System |  |
| Automatic Monitoring in ASP.NET |  |
| Seamless Integration |  |

<https://waverleysoftware.com/blog/why-choose-net-pros-and-cons/#3>

● Golang Gin

|  |  |
| --- | --- |
| **Pros** | **Cons** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

● Golang CHI

|  |  |
| --- | --- |
| **Pros** | **Cons** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

● Python Flask

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Scalable | Not a lot of tools |
| Flexible | Difficult to get familiar with a larger Flask app |
| Easy to negotiate | Maintenance costs |
| Lightweight |  |
| Documentation |  |

<https://careerfoundry.com/en/blog/web-development/what-is-flask/#advantages-and-disadvantages-of-flask>