netguru

Ruby on Rails - a perfect way to kickstart your software business

Quick Guide for CEOs

Date: Nov 6th, 2017

Table of contents

Selecting a technology in which you will develop a web app is a challenge that every product or business owner has to face. If you make the right choice, it will give you a solid base for growth and expansion. If you choose wrong though, it may cost you an arm and a leg.

Ruby on Rails (RoR) is a popular web framework for web application development built on the Ruby programming language. As a full-fledged web framework, RoR offers many components of a successful web project, such as an ORM (Object Relational Mapping) system for business data and logic, routing, and application management out of the box. Still, to decide whether RoR is a good fit for your project, you need to know what makes this framework different from others. To help you build a deeper understanding of RoR, we are going to give an overview of its main strengths and limitations.

Why and when to use Ruby on Rails?

Ruby on Rails performs well in CPU-intensive applications. If you consider building CPU-heavy applications, Rails is definitely a better option than, for example, Node. Rails is also better when the speed of development is critical. With all the modules and generators available out of the box, Rails is very powerful in Rapid Application Development (RAD). Just in few commands, you may have a fully functional prototype that may be amended with additional features later. Fast prototyping is the philosophy of Rails.

1. Best Industry Standards

Ruby on Rails is an opinionated framework which means it guides you into their way of doing things. It promotes the best standards and practices of web development.
The central pillar of the Rails philosophy is the DRY (Don't Repeat Yourself) principle that ensures a clear separation of concerns and maintainability of your application. The framework embraces the principle of 'convention over configuration', according to which Rails defaults to a set of conventions that specify the best way of doing many things.
Rails is also built around the MVC (Model-View-Controller) philosophy that promotes modularity and extensibility of your applications. This means that no matter how complex your application is, it can be easily extended with new features and business logic.
Just to illustrate the level of complexity one can achieve with RoR, take a look at GitHub. This is the largest repository of source code in the world, built on a complex architecture of version control and distributed software development. All this complexity is successfully managed by the RoR framework.

2. Speed of Development

RoR was created with the high velocity of prototyping and application development in mind. Its well-developed system of modules, generator scripts, and an efficient package management system allow scaffolding a complex application in just a few commands. One can achieve rapid application

development thanks to the expressive and concise nature of Ruby language, and also to dozens of open-source libraries for just about any purpose, which the Ruby community calls ‘gems’.
As an added bonus, Rails ships with a default ORM system (ActiveRecord), which helps developers quickly put application and data logic together and deploy a fully functional prototype to be expanded with new features later.

3. Vibrant Community

Rails is an open-source web framework supported by a vibrant community of talented developers. Using RoR in their own projects, Rails developers are interested in the constant improvement of the code base and incorporation of new functionalities. As a result, with Rails, there is no need to reinvent the wheel in your projects.
RoR’s ecosystem contains many “gems”, i.e. pieces of software that can be incorporated into your project. Ruby community takes care of that. Almost any functionality you might need for your web project has already been created. A vibrant community that runs Rails also ensures that the framework is regularly updated, issues are fixed, and security is kept up-to-date with the best industry standards.

Where Ruby on Rails falls short

1. Runtime Speed and Performance

One of the most frequent arguments against RoR is its ‘slow’ runtime speed, which makes it harder to scale your RoR applications. While it’s true that other top environments and frameworks (Node.js or Django) are somewhat faster than RoR, it is unlikely that your application will witness performance bottlenecks, unless it has a user base comparable to such large websites as Twitter.
In most cases, performance issues your RoR application will face will be linked to the server or database architecture and the skillfulness of your engineering team rather than RoR itself. Performance considerations should be still kept in mind, though. Twitter, for example, struggled to improve RoR’s performance that deteriorated after the social network became very popular. Although Twitter did not abandon RoR completely, it had to replace certain internal communication components and server daemons with Scala solutions.

2. Lack of Flexibility

RoR is an opinionated framework with a lot of hard dependencies and modules included out of the box. To kickstart the project, your developers should configure routing, database migrations, and other modules shipped with the framework. These default modules are good if you want to create an application with some standard functionalities, but they might backfire on you if you have something unique in mind. In this case, it may be harder to adjust RoR to your product’s needs. At some point, you will have to make a difficult choice between giving a deep overhaul to your Rails application or using another framework that better suits your needs.

3. High cost of wrong decisions in development

Wrong architecture decisions during the initial stages of your project might cost you more in Rails than in any other framework. Since prototyping with Rails is incredibly fast, an engineering team inexperienced in Rails might make unobvious mistakes that will erode your application's performance in the future. These structural deficiencies will be hard to fix because Rails is an open framework, where all components are tightly coupled and depend on each other. For instance, too much reliance on ActiveRecord makes an application logic tightly coupled with database models, which leads to maintainability problems in the long run. At Netguru, we have patterns that help us prevent these issues from the beginning.

Is Ruby on Rails a Good Choice for Machine Learning?

Ruby is an elegant programming language which found its role in the web development and scripts. With the help of Ruby and Rails framework, developers can build MVPs in a way which is both fast and stable. This is thanks to the availability of various packages called gems, which helps solving diverse problems speedily.
However, looking for the Machine Learning gems, we can conclude that the choice is not that rich. Going deeper, the described solutions are not documented enough. The reason is that they do not provide efficient computation speed and gather a too small community around. All these factors attest to the fact that there are more risks than advantages of using Ruby gems as Machine Learning solutions, and it is not the best choice after all.
Moreover, tools and packages are as useful as the language of development. Ruby is definitely one of the most interesting programming languages. It has many proved purposes, but fast computing is not one of them. Ruby does not match Machine Learning, and we need to look into something better.

What's the Alternative?

Python is also a popular programming language which is often used in Data Science projects:

- It has numerous packages for Machine Learning and other computations. The prime examples are numpy, pandas, keras and tensorflow. These packages are well-documented, which is helpful in starting with new projects and solutions. It also speeds up the process of fixing bugs,
- Its libraries are simply powerful. It means that they comprise many features helpful in complex computations. The development is fast, efficient and stable. It is also common that they use a range of computation speed improvements. All of these advantages make these tools mature and reliable,
- Another important advantage of using Python libraries is a considerable support from the community as the developers can easily find tutorials and tips valuable in a development process. A stable community makes the start threshold lower - it is easier to use new technologies from scratch,
- Python is a developer-friendly language which is easier to start with for Ruby on Rails developers comparing to other, lower-level programming languages. The syntax is intuitive, and it parallels the one in Ruby more than other popular languages.

Tensorflow

We need to choose the best Python library for the Machine Learning purposes. We recommend using Tensorflow, a popular and powerful tool from Google. It provides stable implementation for Python, C++ and many other programming languages. We decide to use Tensorflow for the benefits it provides:

- it has an excellent documentation, a bunch of helpful tutorials and howtos, which helps developers go deep into the Machine Learning solutions,
- it performs all complex calculations “behind” Python - it uses a unique computational engine and leaves Python free of heavy operations,
- it allows building neural nets and other Machine Learning structures like graphs and chains of single operation blocks,
- it allows using Graphics Processing Unit for a much better performance.

RoR as a Web Application for ML

Based on the above, it is a good idea to connect the brilliance of Ruby on Rails framework with Python as a microservice performing Machine Learning computations. This architecture gives us the mix of the best computation efficiency and web application development stability. It minimises the time of building a prototype and provides the best quality of usage.

What are the main benefits of such a combination?

- It’s easy and convenient to connect our app with other microservices. The Rails framework provides many reliable ways of communication between different services. It does not break the integrity with the core services,
- Rails is great for building MVPs. Developers can build a web application fast and pitch it to investors,
- It is a stable solution with a really good documentation. Moreover, there are many famous companies which trusted this framework and built efficient software,
- Active community support also makes this choice smart,
- With the help of gems, Rails packages, developers can quickly build more complex parts of an application.

The proposed architecture of web application using Machine Learning features has both its strong and weak sides. Web application development is stable, and it's possible to use Ruby gems to build a web application fast. It ensures great efficiency of Machine Learning computations thanks to the Python and Tensorflow library. Finally, the connection between both services is fast and safe.

On the other hand, you should consider the downsides as well. It has a bit more complex architecture model at the cost of creating an almost perfect solution.

In the case of Machine Learning ecosystem, it is better to mix different technologies and select the best tools to support them than rely on standalone choices which are not always as good as they seem.

Ruby on Rails for E-commerce

An online store is a good business model. Customers visit your website, browse, put items in the basket, and transfer money to your account. Whereas it might look simple from the outside, many things can go wrong inside your business, especially from the technological perspective.

Choosing The Right Technology

The technology that you use in the e-commerce platform development will determine the platform's performance, stability and security, that is the factors that are vital to your customers. If the website crashes every other step of a customer's journey, the loading times exceed 15 seconds, or the payment gate seems to be dodgy, you can't expect many customers to buy products from you. If you don't want this scenario to come true, you should pay attention to a range of factors and requirements that are important when choosing the framework for an e-commerce store.

Flexibility

Trends and technologies are constantly evolving, and your platform should be evolving too. If you want to stay on top of things, you should take into consideration that you might scale your store up some time after the kickoff, or that you will need to implement new features as the business grows. That's why the platform should be flexible enough to accommodate the changes you will want to implement. It should be possible and easy (!) to develop the platform and adjust its structure to your needs. If the framework you choose for the development is not flexible enough, every single change will be a pain for the dev team and yourself.

Stability And Performance

Stability and performance are some of the key factors determining whether users will buy from you or not. People hate to wait for anything, and it also applies to the online environment. If your website is slow, users will simply abandon it. The same goes for stability – when your potential customers can't achieve their goal (buying a product) fast, they will go shop at one of your competitors. The framework should provide a stable and performant ecosystem for your platform, even when the number of users becomes really high.

Payment Integrations

Secure payments are essential for a good shopping experience. When customers submit sensitive personal data, such as a credit card number, they want to be certain that the data will not be collected and processed for purposes other than completing the transaction. Anything that suggests otherwise is bound to prevent them from buying at your store. Users also value convenient payment options. If they proceed to the last step and don't find a payment gate they usually use, it may stop them from finalising the transaction. Therefore, the framework should make it easy to integrate it with many payment processors.

The Number of Ready Solutions

When you start building an online store, it is good to have some out-of-the-box functionalities so that the development team doesn't need to write all modules from scratch. If the framework you choose provides ready-to-implement packages that will enable you to add products, configure the checkout, or integrate payment gates, you will save loads of development time. On top of that, using proven modules that have already worked in a ton of projects gives you a guarantee that they will work from the get-go.

Ease of Deployment

Building an e-commerce platform is a long process that will require much effort from your team. After months of code writing and testing, you want to kick off with the platform fast and with ease. You don't want any problems on the run-up to the commercial launch. That's why the framework should offer an easy way to deploy the application. If all the processes which are part of deployment are automated, the dev team can quickly put your app into production.

Admin Page Offered by The Framework

Whereas the interface of your online store will adjust to user needs, the admin page usually remains in the form the framework offers it. Maybe you don't need your dashboard to be beautifully designed, but it must be easy to navigate, and you need to find what you're looking for easily.

Ruby on Rails' Spree for your e-commerce

Ruby on Rails gives you a reliable framework intended for building online stores: Spree. Spree is the most popular e-commerce solution for RoR, supported by the community and constantly updated to better fit your needs. Does it fulfil the requirements of a good framework?

Mature Codebase

Spree was created in 2007, and since then, over 700 developers have contributed to the codebase. The community actively supports the framework by adding new repositories and creating more and more features that can be directly implemented in the platform. The codebase is mature, and contributors make a lot of effort to provide the highest possible quality. It all translates into the exceptional stability of the framework. That's why it's been used by many companies, for instance, Chipotle Mexican Grill, Second Life, Bonobos, Fortnum and Mason.

Ready Functionalities

Spree offers basic out-of-the-box shop functionalities essential for every e-commerce platform. The features important from a customer's point of view but hard to write from scratch are available in Spree as ready-to-use packages:

- Managing products. You can add products in distinct variants, such as colours, sizes etc. You will be able to upload images for products and for their specific versions separately. The packages also allow managing the stocks of the created products.
- Managing tax zones. This feature is especially important when you run an international business. Spree offers a well-developed system for managing taxes. It allows creating many tax categories (e.g. food, clothes, or electronics) various tax zones (e.g. EU, USA) and also to easily combine zones with categories.
- Managing shipments. You can create different shipping methods and use different delivery methods for different zones, making it easy to integrate the system with local logistics companies.
- Managing promotion campaigns. You can apply a promotion to a specific order if a predefined condition is met (for instance if the total value of the basket is higher than $100, or a specific product was added to the cart). It also facilitates applying promo codes.

Taking into consideration the sheer number of functionalities and the amount of complexity that Spree offers, its performance is very good.

Easy Payment Integrations

Spree offers a wide range of payment integrations, including the most popular services such as Stripe, PayPal, or SagePay, but also smaller providers (Authorize.Net, Braintree, Moneris, Samurai, Skrill, USA ePay, or WorldPay). It has a highly flexible payment integration model, enabling you to provide your

customers with multiple payment methods during checkout. The biggest advantage is that if gives developers a built-in service for managing payment gates. As a result, they don't need to write code – clicking through it will be enough to make it work.

Documented Admin Dashboard

Even though you might feel a little bit confused when you first see the admin dashboard, it won't take you long to familiarise yourself with it. The dashboard offers numerous functionalities that can be overwhelming at the beginning, but you will definitely appreciate the wide range of options it offers. On top of that, the admin page is solidly documented making navigating through the platform easy, especially during the first use.

Easy Deployment

Just like with any other Ruby on Rails app, the deployment doesn't cause any problems. It's quick and easy. We've tested it on our Docker flow many times, and we haven't come across any major issues.

Things to Consider Before Development

Despite the maturity of the codebase and the stability of the framework, like any other technology, developing in Spree involves some risks, and problems may occur. However, if you know those risks in advance, you will know what to expect and how to handle the problems that might ensue.
The first thing you should consider is the framework's integration with JavaScript. If you're using Ember or React on the front end, it can take a little bit more time to connect the technologies. Spree's out-of-the-box API is not well-suited for rich Javascript applications, and developers need to modify it slightly before connecting the customer's side with Spree. It might get especially complicated with Ember. On top of that, search engines in Ember or React might sometimes refuse to work smoothly with such applications. Secondly, if you want to implement some non-standard functionalities (e.g. replacing the multi-page checkout with a single-page checkout) in your online store, you should know that introducing deep modifications to Spree can take a lot of development time and might require more experienced programmers. Still, introducing deep changes into Spree is possible. Another thing is that payments that need a 3D-secure verification will also involve more development work and testing. It is doable, but problems can occur, and the dev team will have to put more effort into making it all work. Finally, you might encounter some performance issues the performance if you don't use external services such as CloudFront.

Successful Implementations

Ninety Percent

Ninety Percent is a London-based clothing selling their collection online. What sets them apart from other fashion online stores is their business model. The company shares 90 percent of their profits between four charities, and customers are the ones who can vote for the cause they wish to support. The remaining 10 percent go to investors. In the project, we managed to implement the 3D-secure verification successfully. We also had to modify the promotion module and adjust it to our client's needs. Finally, we built from scratch the system that enables customers to choose the cause they wish to support. The project will have its commercial release soon.

Apparel E-commerce Store With Influencers

In another project from the fashion industry, we also had to implement a few dedicated solutions. The platform we worked on engages influencers who offer their products in the store. Apart from handling a complex front end, we had to integrate the Instagram API so that influencers could show their feeds on the website. We also created a system for promoting collections that have already been launched and those that are coming soon.

Ruby on Rails Development Agency - How to Prepare The Setup And Workflow For Your Team

Building an in-house team can be quite a challenge. That's why remote development teams are gaining popularity, and more companies use services of Ruby on Rails development agencies. It's an attractive alternative, but if you want everything to go smoothly, make sure that you prepare yourself before you start cooperation. Communication and transparency are crucial in Ruby on Rails development. Following certain rules will help you solve all potential problems, manage workflow, and keep all issues under control. Before the setup, you should learn a little bit about the technology so that you can understand your team better and streamline the process.

The client: the most important part of the process

The cooperation between a client, and a development agency is both very individual and generic at the same time. From the development's team point of view, the goal of the cooperation is to create a product and possibly maintain it in the future. That's why wrong assumptions, approach, plan, and the execution of that plan can lead to different expectations and results at many stages of product's creation.

Rule #1: Find time to meet with the team

The client will have a very specific vision for the product. But very often, the client forgets to communicate it to the Ruby on Rails agency. So the first thing you should bare in mind is to make time in their calendar for regular meetings with the development team. Meetings should take place at least once a week, for a number of reasons.

- Meetings constitute a good opportunity to present the progress with the product made by the development team in each sprint. From our experience, during meetings, clients will have many questions about developers' work, in accordance with the workflow, layout, and business logic. At such meetings, called stand-up meetings, many edge cases are very often targeted and clarified. A stand-up meeting should take place at least once a week, but twice would be even more beneficial,
- You can be sure that the project goes in line with assumptions and the timeline. That argument is very valid in terms of time and budget,

- Frequent meetings adhere to the lean management approach to creating web products. This approach guarantees delivering more value with less waste in the context of a project,
- The final product can differ very significantly from the client's first idea. This is why the development agency should cooperate closely with the client in order to react to any changes and requests to make sure everyone is on the same page.

Rule #2: Get acquainted with the tools and crucial project details

In Netguru, we believe that transparency pays off. It enhances the development, it improves relationships in the team, and it also increases the quality of the work with the client. That's why you should be familiar with all the vital information regarding your project. You should have full access to the:

- code repository,
- staging servers,
- domains,
- hosting,
- communication channels,
- tickets in project management tools
- documents such as readmes, plans, and mockups.

Transparency requires security, so all data should be provided in a secure way. For instance, we use 1password to handle passwords, addresses, and confidential data from the list above.

Rule #3: Be present

The process of creating a web application presents multiple challenges, variables, and edge-cases that will occur during cooperation with development agency. Such complex eco-system requires being present in many ways.

- Use staging and beta servers after the development team finishes work on new functionalities. It is very important because the faster you test functionalities, the faster you can spot the discrepancies between the client's vision,
- Slack is a great communication tool. You can ask questions on dedicated Slack channels, so the developer that created functionality or PM can respond,

- In Netguru, we use JIRA. We think it is one of the best tools to track project progress. We know that not everyone has the habit of working with a tool of this kind, but try to make an effort to use JIRA.

The effective collaboration checklist

When working with a RoR development company, you have to manage the work with people, not only with software. Agencies understand that their client's success is their success too and will do everything they can to make this success a reality. In order to work efficiently with an agency you should adhere to a few principles and everyone will be happier.

Rule #5: Prepare yourself for the cooperation

Before the collaboration starts, you should do a few things to make everyone work more efficiently.

- Check whether the company is responsive to the business and marketing changes suggested by the client,
- Check the internal procedures of the agency you're about to choose,
- Check whether the agency has a talent team responsible for delegating the right people to the right job,
- Talk with other clients of the agency you've chosen and ask about their experience.

When you start the collaboration:

- Prepare and provide access to repositories, mockup and graphic designs,
- Put together a list of ideas and user stories,
- Write down the questions about your project as a list,
- Describe the market and identify your target group.

During the co-operation, you need to:

- Specify the requirements for your MVP,
- Set up all the necessary communication channels,
- Always ask questions when you don't understand something,
- Identify the next steps to be taken.

After the launch of the product or at the end of the collaboration:

- Make sure you keep the agency close to hand,
- Check whether the code you've been given is well-documented,
- Ask about maintenance services for your product.

The best practices in Ruby on Rails ecosystem

Every ecosystem follows a set of guidelines. You should know them in order to maximise the gains from collaborating with an agency. In the case of RoR, the guidelines are as follows.

- Use gems. Gems are open-source libraries which you can reuse them in many projects. If you need to implement user registration, image uploads, or automatic e-mail distribution in your application, feel free to use an open-source library. You don't need to write it from scratch,
- Use the right database for your use case. Choose the type of a database appropriate for your problem wisely. You can choose relational databases (RDBMS) such as MySQL, PostgreSQL, MariaDB, Percona, and many more. On the other hand, also non-relational databases are available: MongoDB, Redis, Cassandra, CouchDB, and others. A database you would use for storing users in the system would differ from a database used for real-time storing and processing chat messages for millions of users. Every database was designed with a particular purpose in mind,
- Consider using an external API. Your development team doesn't need to write everything from scratch. They can use an external API to save time and money so that they can focus on solving other, more pressing, issues. You don't need to write software for converting currencies, posting messages on Facebook, or using Google Maps. All major companies in the world have an API available that you can use for a small fee or free of charge,
- Follow the best RoR and programming patterns to keep your software maintainable and easy to develop. The acronyms DRY, KISS, and SOLID should be no mystery to you. Use the right design patterns, keep your software and libraries up-to-date, use useful database features such as indexes, views, Postgis, or full-text search if needed,
- Use continuous delivery tools. In order to save time on delivery process. use tools that make this process as painless and automatic as possible.

Ready to Start Yet?

Even though working with a remote team might have its downsides in comparison to an in-house development team, with the right preparation and management, you won't feel the difference. The crucial principle to follow is to treat all the members as if they were employees of your own company. They need to understand your vision in order to follow and execute it.

If you want to learn more about working with remote teams or you're ready to hire one, drop us a line. We're here for you!