Beginner's Guide to building API using Ruby on Rails

Abhinav Risal
Senior Software Engineer
Bajra Technologies

Roman KC
Software Engineer
Bajra Technologies

Saturday, June 19, 2021

2:00 pm NPT onwards

Agenda

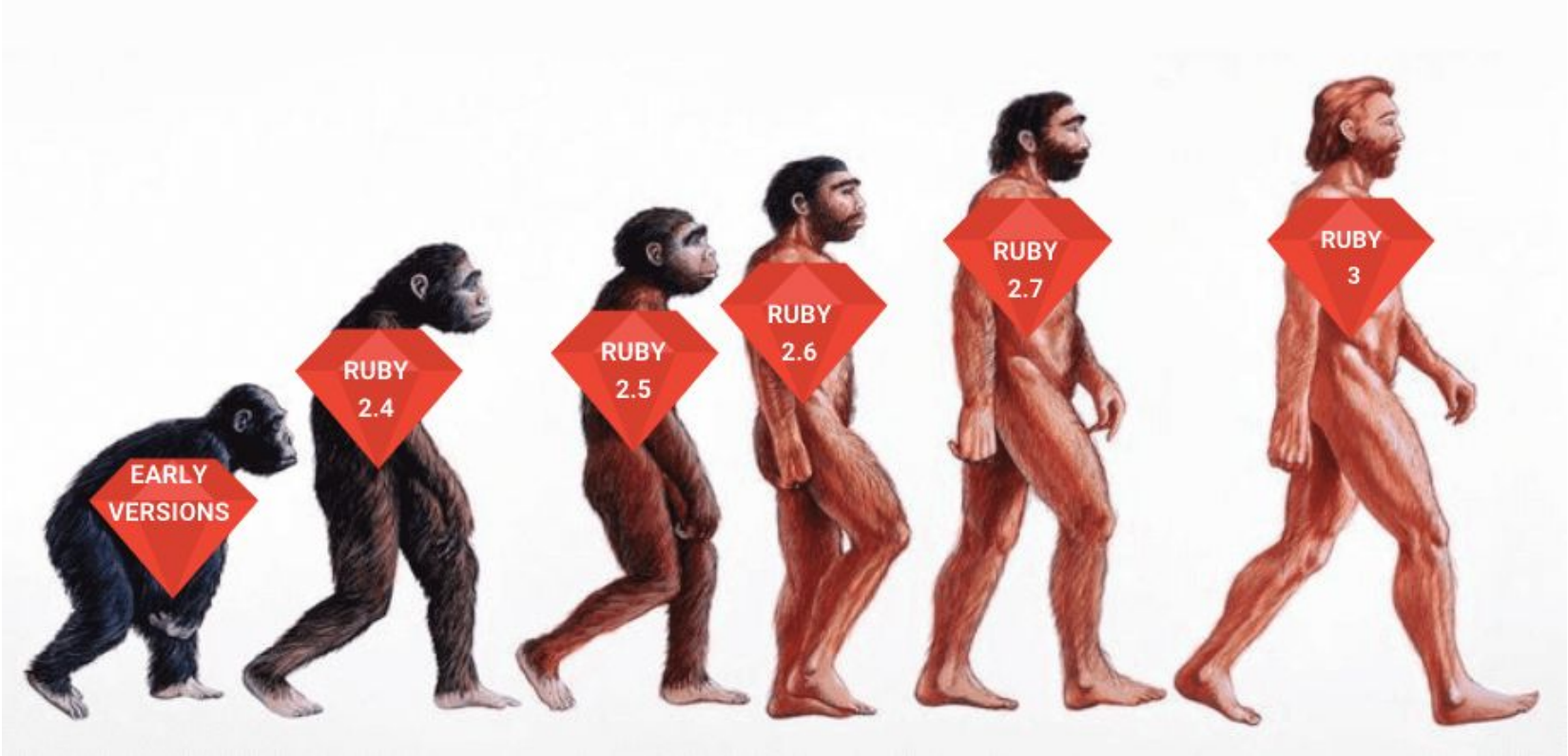
1. Introduction
2. Ruby
3. Rails
4. Live Demo
5. Q & A

Introduction

- **Why this topic ?**
  - Get people acquainted with this framework.
  - Expand the community.
- **Why Ruby on Rails?**
  - Rails propagated the use of the MVC pattern and good development practices, such as the DRY principle.
  - Provides a set of defaults that allows us to get up and running quickly.

Assumptions on Ruby / Rails

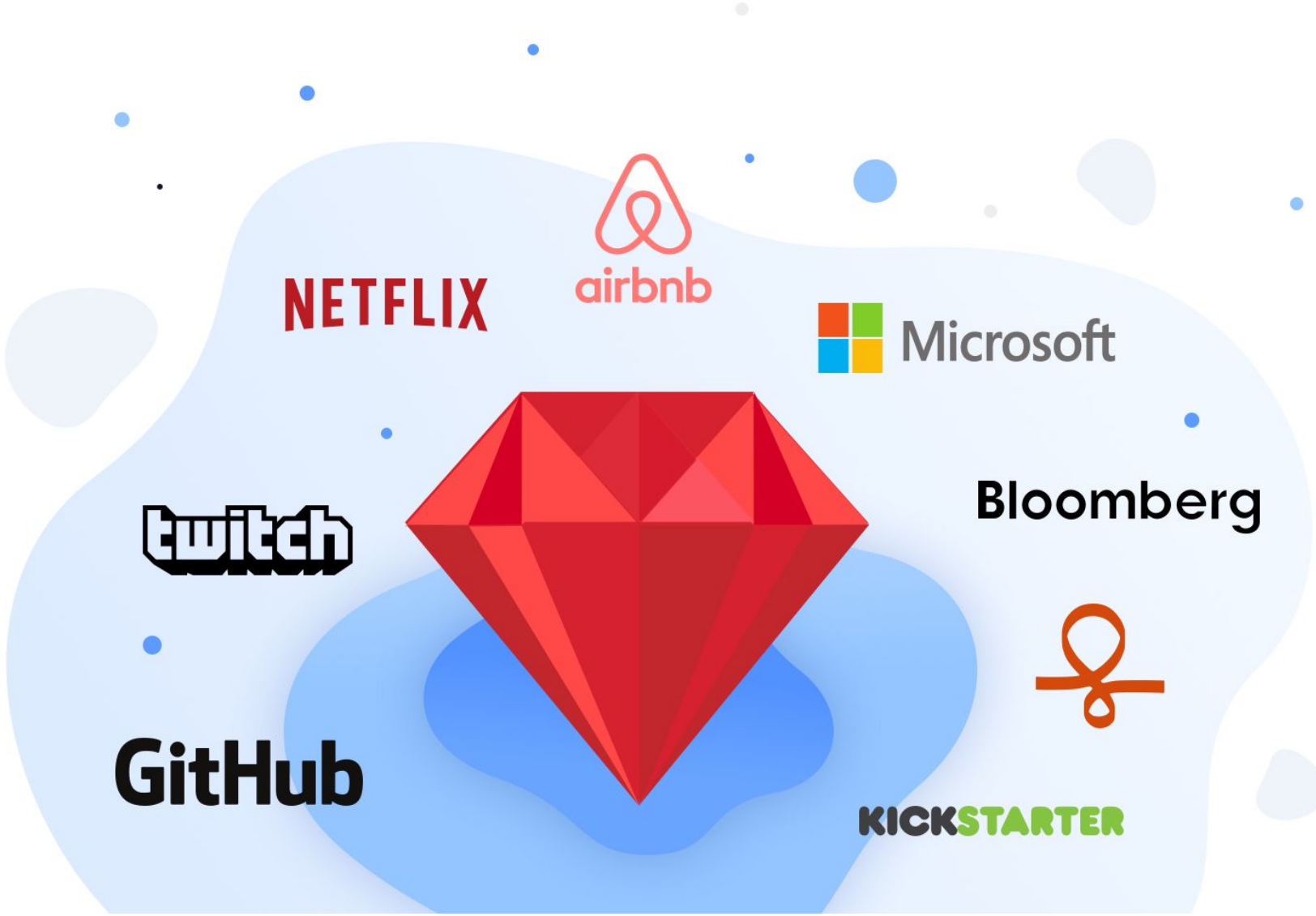
Is Ruby old?



Is Ruby dead?

BAJRA TECHNOLOGIES

Which companies are using Rails?



BAJRA TECHNOLOGIES

Ruby is an interpreted, high-level, general-purpose, and fully object oriented programming language. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.

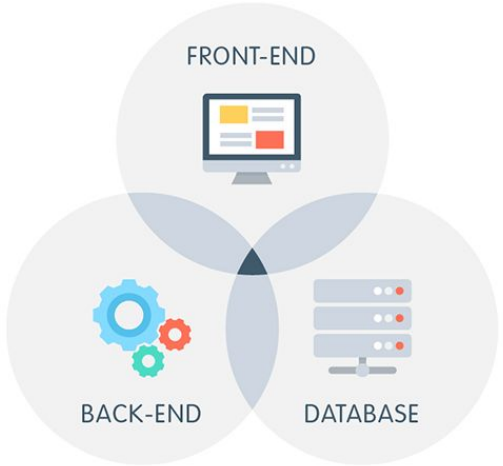
According to Matsumoto, the guiding ideas behind the creation of Ruby was to:

1. Design a programming language that he himself enjoyed coding in,
2. Make it fun to use for developers,
3. Limit the human effort and work associated with using it.

The language has been said to follow the principle of least astonishment, which means that it behaves in a way that minimizes confusion for an experienced user.

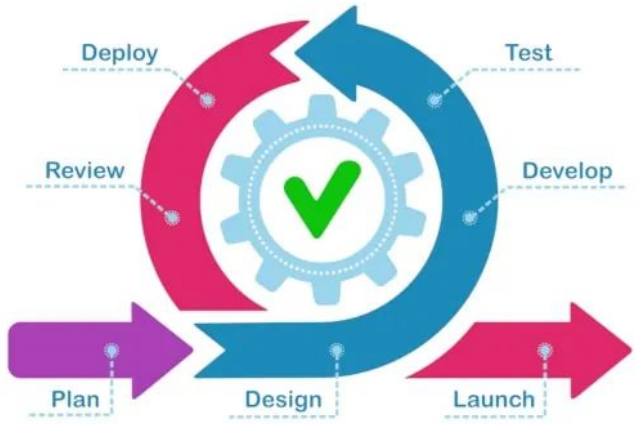
Latest version: 3.0.1 / 5 April 2021

Rails is a ruby gem itself, written in Ruby by David Heinemeier Hansson.



Full Stack

Open Source



Rapid Web Development

Latest version: 6.1.3.2 / 05 May 2021

Rails Philosophy | The Rails Way

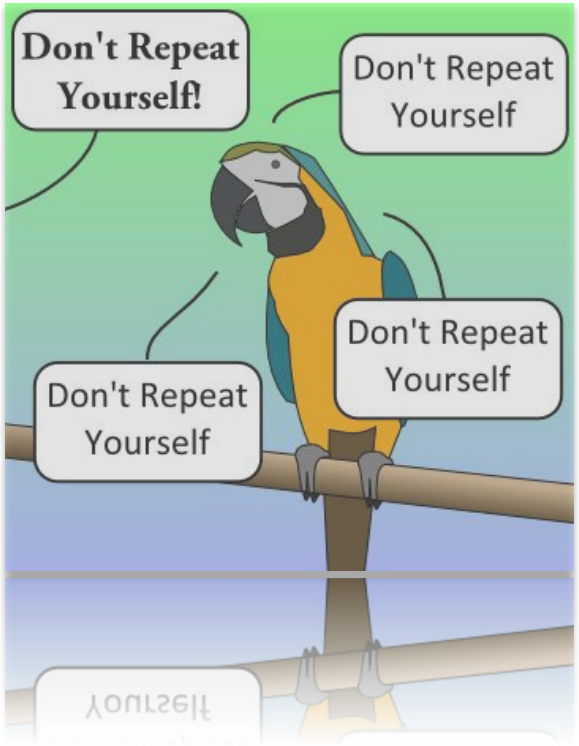
- **Don’t Repeat Yourself**

  By not writing the same information over and over again, our code is more maintainable, more extensible and less buggy.

- **Convention Over Configuration**

  No More imports!!

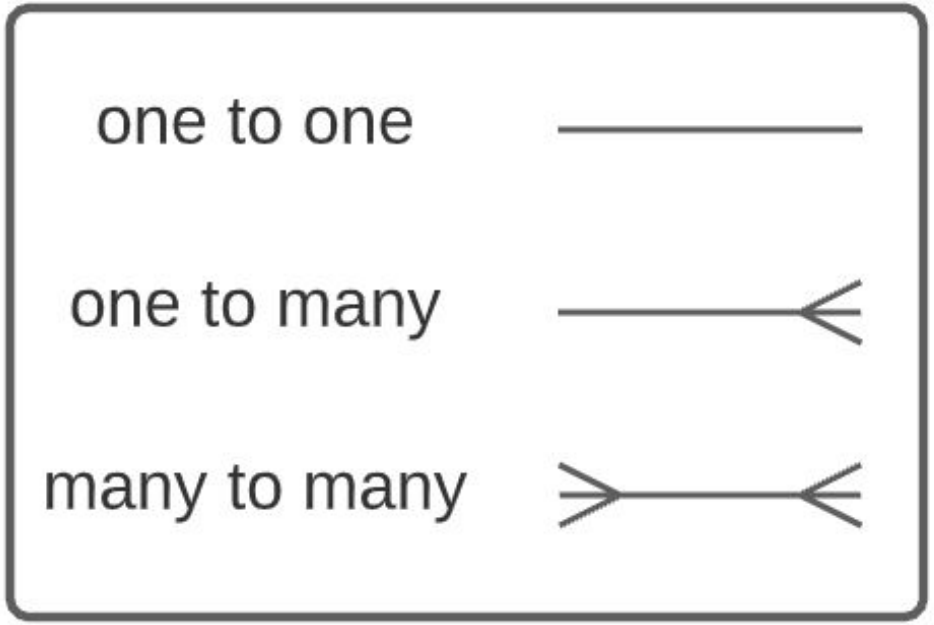
  Our application code and our running database already contain everything that Rails needs to know.



Let’s Build an Application!

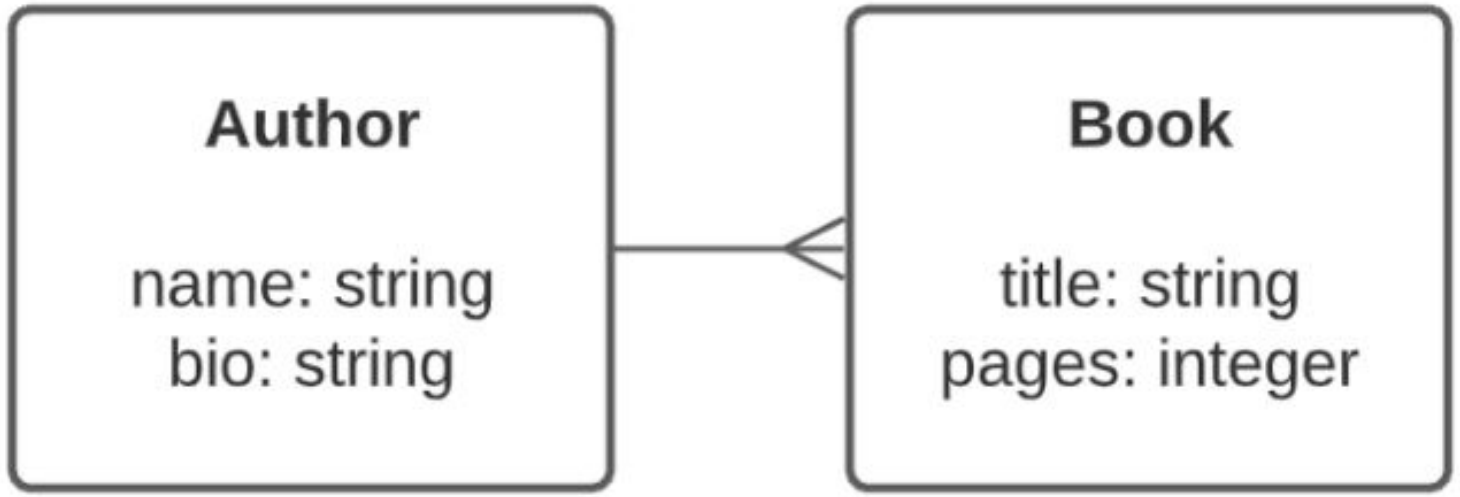
- Application overview
- Planning
- MVC Pattern
- Ruby Gems
- Live Demo

Database & Planning



Crow's Foot notation reference

Library Management Application



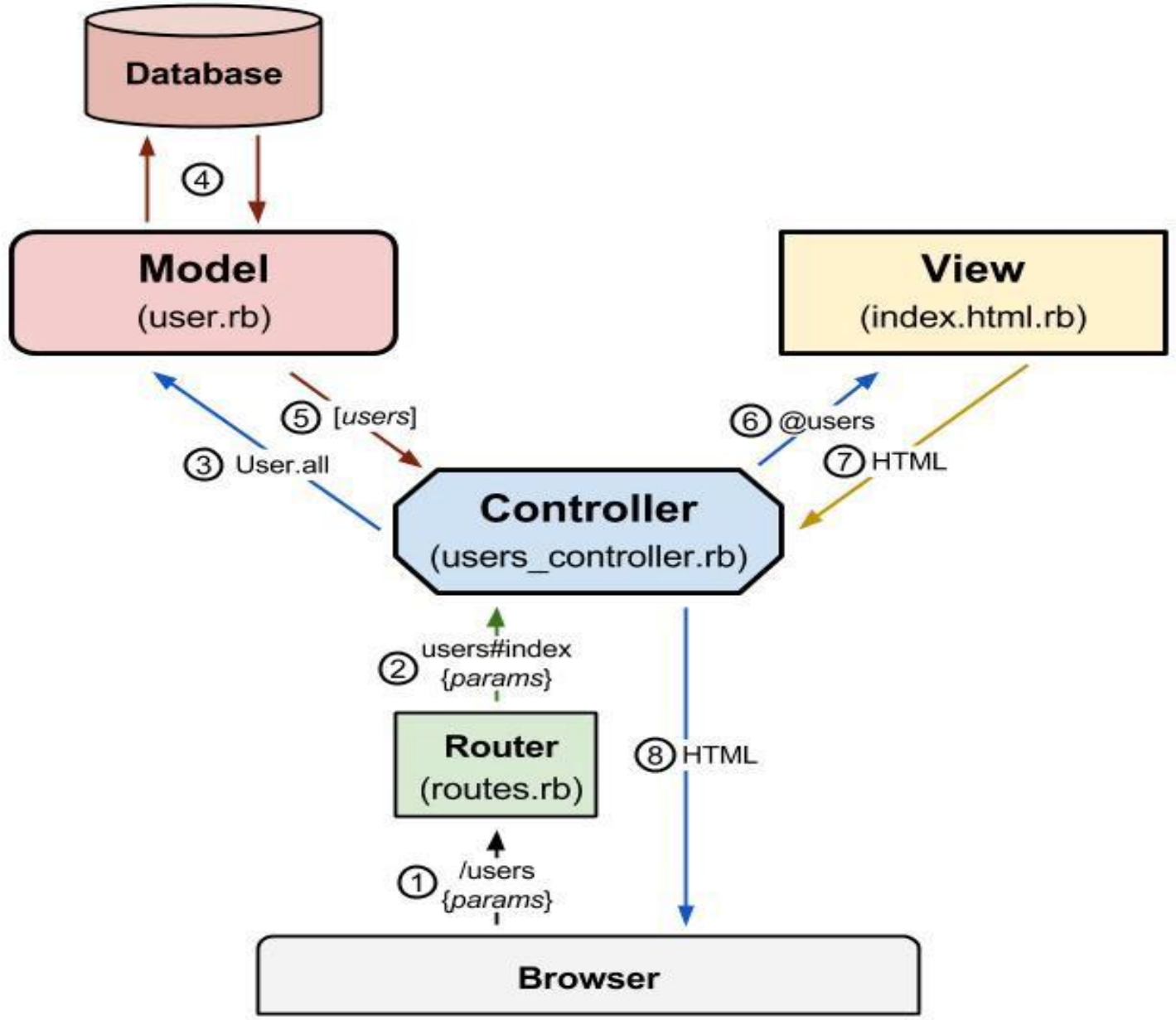
Domain model for our library-API

An author has_many books.

A book belongs_to an author.

BAJRA TECHNOLOGIES

MVC (Model-View-Controller) pattern



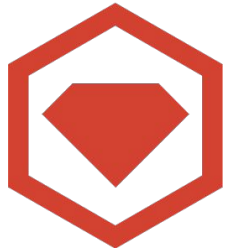
BAJRA TECHNOLOGIES

MVC (Model-View-Controller) pattern - continued

- The **Model** contains only the pure application data, it contains no logic describing how to present the data to a user.
- The **View** presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.
- The **Controller** exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events.

Ruby Gems :

Package manager for the Ruby programming language that provides a standard format for distributing Ruby programs and libraries.



1. **Web framework**: Rails
2. **Code quality**: Rubocop, Guide, Overcommit
3. **Debugging**: ByeBug, Better_errors
4. **Testing**: Rspec, Capybara, Factory Bot, Faker
5. **Authentication and Authorization**: Devise, Ruby-JWT , CanCanCan, OmniAuth
6. **Uploading Files**: Carrierwave, MiniMagick
7. **Search**: Elasticsearch
8. **Admin panels**: Activeadmin

Live demo contents

1. Initializing rails API
2. Model
3. Migration
4. Validation
5. Controller
6. Routes
7. Seeding the database

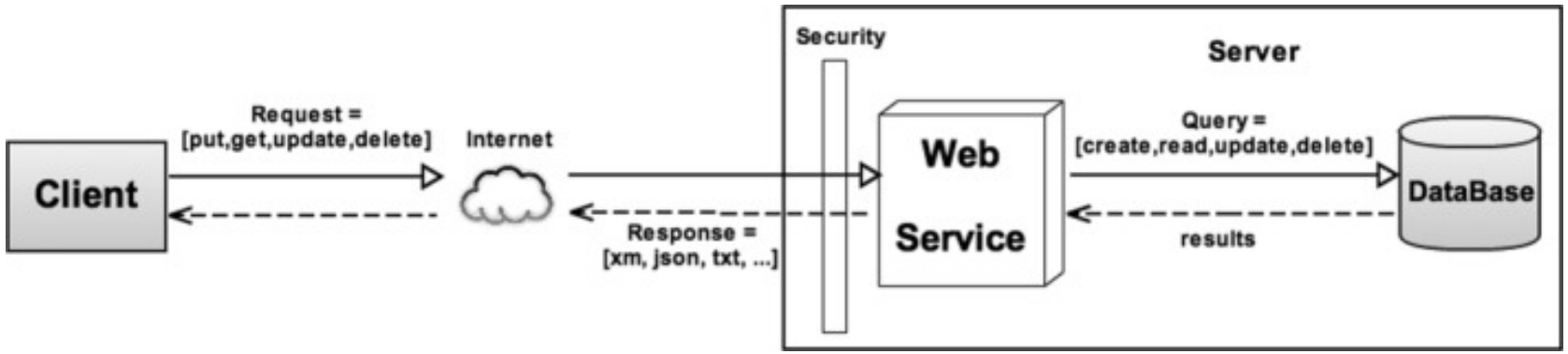
REST VS CRUD

In plain terms, **REST** stands for **Representational State Transfer**, an architectural style designed for distributed hypermedia, or an Application Programming Interface. You've probably heard the latter referred to as an API.

CRUD is an acronym for: CREATE READ UPDATE DELETE

REST is a robust API architecture.
CRUD is a cycle for keeping records current and permanent.

CRUD principles are **mapped** to REST commands to comply with the goals of RESTful architecture.



Using HTTP Methods for RESTful Services

The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently. Of those less-frequent methods, OPTIONS and HEAD are used more often than others.

HTTP Verb	CRUD
POST	Create
GET	Read
PUT	Update/Replace
PATCH	Update/Modify
DELETE	Delete

Questions?

BAJRA TECHNOLOGIES

Acknowledgements

Thank You