Ruby - basics and implementation

Here is a complete step-by-step guide to learning Ruby and setting up the backend for the ft\_transcendance project. It includes detailed examples and resources for each step.

---

1. Learning Ruby Basics

Key Topics to Cover

1. Variables and Data Types:

# String, Integer, Boolean

name = "John"

age = 25

is\_logged\_in = true

# Printing variables

puts "Name: #{name}, Age: #{age}, Logged in: #{is\_logged\_in}"

2. Control Structures:

# if/else

if age > 18

puts "Adult"

else

puts "Minor"

end

# Case statement

case age

when 0..12

puts "Child"

when 13..19

puts "Teenager"

else

puts "Adult"

end

3. Arrays and Hashes:

# Array

colors = ["red", "green", "blue"]

puts colors[0] # Access first element

# Iterate over array

colors.each { |color| puts "Color: #{color}" }

# Hash (key-value pairs)

user = { name: "John", age: 25 }

puts user[:name] # Access value by key

4. Methods:

def greet(name)

"Hello, #{name}!"

end

puts greet("Alice") # Outputs: Hello, Alice!

5. Classes and Objects:

class User

attr\_accessor :name, :age

def initialize(name, age)

@name = name

@age = age

end

def greet

"Hello, #{@name}!"

end

end

user = User.new("John", 25)

puts user.greet # Outputs: Hello, John!

---

2. Setting Up Your Ruby Environment

Install Ruby

On Debian-based systems (Linux):

sudo apt update

sudo apt install ruby-full

Verify installation:

ruby --version

Install Bundler

Bundler manages your project’s dependencies:

gem install bundler

---

3. Create Your Project

Initialize a Ruby Project

1. Create a directory for your project:

mkdir ft\_transcendance-backend

cd ft\_transcendance-backend

bundle init

2. Add dependencies to the Gemfile:

gem 'sinatra' # Lightweight web framework

gem 'pg' # PostgreSQL adapter

gem 'bcrypt' # For password hashing

gem 'dotenv' # Manage environment variables

gem 'faye-websocket' # WebSocket support

3. Install the dependencies:

bundle install

---

4. Build a Basic Web Server with Sinatra

Create app.rb

1. Add a simple Sinatra app:

require 'sinatra'

get '/' do

"Welcome to ft\_transcendance"

end

2. Run the app:

ruby app.rb

3. Open http://localhost:4567 in your browser to see the output.

---

5. Connect to a PostgreSQL Database

Install PostgreSQL

1. Install PostgreSQL:

sudo apt install postgresql postgresql-contrib

2. Create a database and user:

sudo -u postgres psql

CREATE DATABASE ft\_transcendance;

CREATE USER ft\_user WITH PASSWORD 'securepassword';

GRANT ALL PRIVILEGES ON DATABASE ft\_transcendance TO ft\_user;

\q

Add a Database Connection

1. Create a .env file for storing environment variables:

touch .env

Add the following:

DB\_NAME=ft\_transcendance

DB\_USER=ft\_user

DB\_PASSWORD=securepassword

2. Update app.rb:

require 'pg'

require 'dotenv/load'

conn = PG.connect(

dbname: ENV['DB\_NAME'],

user: ENV['DB\_USER'],

password: ENV['DB\_PASSWORD']

)

get '/users' do

result = conn.exec("SELECT \* FROM users;")

result.map { |row| row }.to\_json

end

3. Create a users table:

sudo -u postgres psql ft\_transcendance

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(50),

password VARCHAR(255)

);

---

6. Add User Registration and Authentication

Register Endpoint

require 'bcrypt'

post '/register' do

data = JSON.parse(request.body.read)

hashed\_password = BCrypt::Password.create(data['password'])

conn.exec\_params(

"INSERT INTO users (username, password) VALUES ($1, $2)",

[data['username'], hashed\_password]

)

{ status: 'User registered' }.to\_json

end

Login Endpoint

post '/login' do

data = JSON.parse(request.body.read)

result = conn.exec\_params(

"SELECT \* FROM users WHERE username = $1",

[data['username']]

)

if result.any? && BCrypt::Password.new(result[0]['password']) == data['password']

{ status: 'Login successful' }.to\_json

else

halt 401, { error: 'Invalid credentials' }.to\_json

end

end

---

7. Add WebSocket Support

Install WebSocket

Add this to the Gemfile:

gem 'faye-websocket'

Install:

bundle install

WebSocket Endpoint

require 'faye/websocket'

clients = []

get '/ws' do

if Faye::WebSocket.websocket?(env)

ws = Faye::WebSocket.new(env)

clients << ws

ws.on :message do |event|

clients.each { |client| client.send(event.data) }

end

ws.on :close do |\_event|

clients.delete(ws)

end

ws.rack\_response

else

halt 400, 'WebSocket endpoint'

end

end

---

8. Test Your Setup

Test WebSocket

Use websocat:

websocat ws://localhost:4567/ws

Test Registration and Login

1. Use curl to test the /register endpoint:

curl -X POST -H "Content-Type: application/json" -d '{"username":"test","password":"1234"}' http://localhost:4567/register

2. Test the /login endpoint:

curl -X POST -H "Content-Type: application/json" -d '{"username":"test","password":"1234"}' http://localhost:4567/login

---

Resources for Further Learning

1. Ruby Basics:

Learn Ruby the Hard Way

Ruby Official Documentation

2. Sinatra:

Sinatra Guide

3. PostgreSQL:

PostgreSQL Documentation

4. WebSockets:

Faye WebSocket Documentation

Would you like me to assist with setting up a matchmaking system, tournament brackets, or deploying the backend with Docker?