PRÉSENTATION DE STAGE

PROGRAMME

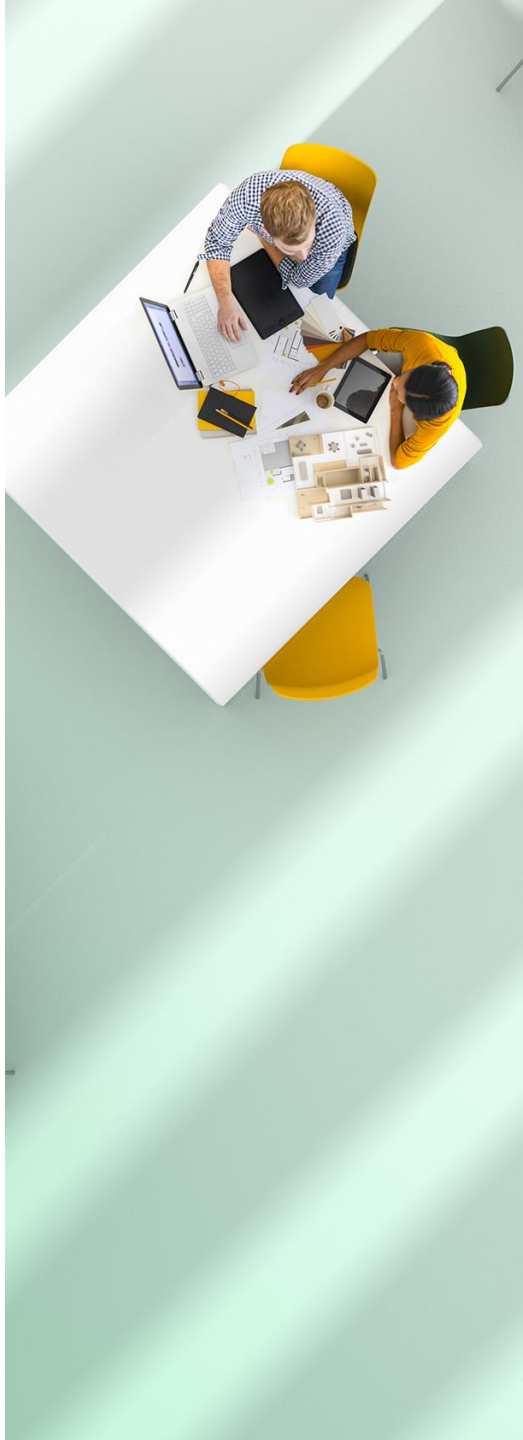
Introduction

Contexte du stage

Mission

Difficulté

Compétences



NITRAM SOFTWARE

CONTEXTE

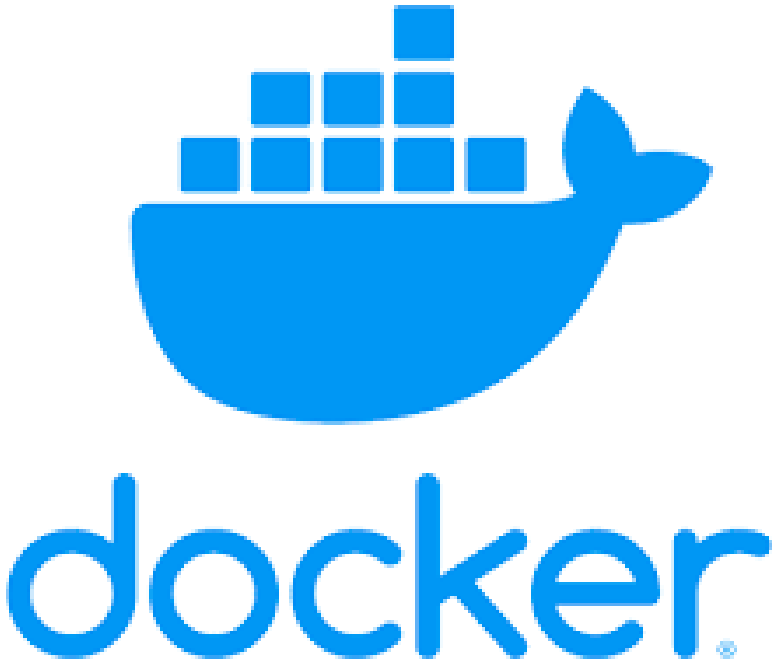
DOMAINE D'ACTIVITÉ: CONCEPTION, DÉVELOPPEMENT, MAINTENANCE LOGICIEL.

LOCALITÉ:(26200) MONTÉLIMAR

MISSION

- docker
- Ruby, Ruby on rails
- Wrapper Http
- Refonte de logiciel de facturation.
- Visio conférance
- Bots d'exchange.

DOCKER



Docker est une **plateforme de conteneurisation** qui permet d'emballer une application avec toutes ses dépendances (bibliothèques, configuration, environnement) dans un **conteneur isolé et portable**. Cela garantit que l'application s'exécute de la même manière, que ce soit sur un poste local, un serveur ou dans le cloud.

WRAPPER HTTP

```ruby
# Gemfile
gem 'httparty'

# app/services/weather_api_wrapper.rb
require 'httparty'

class WeatherApiWrapper
  include HTTParty
  base_uri "https://api.open-meteo.com/v1"

  def self.get_weather(city_lat, city_lon)
    response = get("/forecast", query: {
      latitude: city_lat,
      longitude: city_lon,
      hourly: "temperature_2m"
    })

    if response.success?
      response.parsed_response
    else
      { error: "Impossible de récupérer les données météo." }
    end
  end
end

# app/controllers/weather_controller.rb
class WeatherController < ApplicationController
  def show
    @weather = WeatherApiWrapper.get_weather(48.8566, 2.3522) # Paris
    render json: @weather
  end
end
```

UN **WRAPPER HTTP** EST UNE COUCHE LOGICIELLE QUI SIMPLIFIE LA COMMUNICATION AVEC DES SERVEURS WEB EN MASQUANT LA COMPLEXITÉ DU PROTOCOLE HTTP. PLUTÔT QUE DE MANIPULER DIRECTEMENT LES EN-TÊTES, LES CODES DE STATUT OU LES SOCKETS, IL FOURNIT DES MÉTHODES SIMPLES POUR ENVOYER DES REQUÊTES (GET, POST, ETC.), RECEVOIR LES RÉPONSES ET GÉRER LES ERREURS.

BOTS D'EXCHANGE

```ruby
# bot/exchange_bot.rb
require 'httparty'

class ExchangeBot
  include HTTParty
  base_uri "https://api.binance.com"

  def get_btc_price
    response = self.class.get("/api/v3/ticker/price", query: { symbol: "BTCUSDT" })
    if response.success?
      response["price"].to_f
    else
      raise "Erreur lors de la récupération du prix."
    end
  end

  def trade_strategy
    price = get_btc_price
    puts "Prix actuel du BTC : #{price} USDT"

    if price < 25000
      puts "→ Acheter du BTC 🚀"
      # Ici, tu appellerais l'API de trading de Binance pour acheter
    elsif price > 30000
      puts "→ Vendre du BTC 💰"
      # Ici, tu appellerais l'API de trading de Binance pour vendre
    else
      puts "→ Attente, pas d'action."
    end
  end
end

# Exemple d'utilisation
bot = ExchangeBot.new
bot.trade_strategy
```

Un **bot d'exchange** est un programme automatisé qui interagit avec une plateforme de trading (comme Binance) pour exécuter des ordres d'achat et de vente de manière autonome, selon des règles prédéfinies (par exemple : arbitrage, suivi de tendance, scalping). Il utilise généralement l'**API de l'exchange** pour consulter les prix, gérer un portefeuille et placer des transactions.

DIFICULTÉ

- Durée du stage trop courte.
- Indisponibilité du maitre de stage.
- Manque de taches.
- Clients qui ne répondents pas.

COMPETENCES

- Docker
- Ruby
- Ruby on rails
- Relationnels clients
- Excel