

Coral Protocol

Infraestrutura Aberta para Conectar a Internet dos Agentes

Pesquisador: Luis Henrique Bonini Santiago

Pesquisador: Pedro Humberto Bitencourt Nascimento

Especialista: Luiz Katzap



O Que é o Protocolo Coral?

Definição:

O Coral é uma **infraestrutura de colaboração aberta e descentralizada** que permite a comunicação, coordenação, confiança e pagamentos entre agentes de IA. Uma base unificadora que estabelece padrões de comunicação, segurança e governança, criando o ambiente necessário para que diferentes agentes de IA colaborem com eficiência, confiança e escala global.

É um framework ou uma ferramenta?

O Coral é melhor descrito como um **protocolo de interoperabilidade** entre agentes . Ele não orquestra diretamente, mas define um meio padrão de comunicação (eventos, mensageria, estado compartilhado).

O Coral Protocol é construído em cima do MCP (Model Context Protocol) – ele não substitui o MCP, mas o estende, adicionando camadas de segurança, autenticação e governança.

O Desafio da Fragmentação da IA



Atualmente, o universo da Inteligência Artificial é preenchido por inúmeros agentes altamente especializados. No entanto, estes frequentemente operam em "silos" isolados, incapazes de se comunicar e colaborar eficientemente, limitando o verdadeiro potencial da IA em escala.



Agentes Analíticos

Processam vastos conjuntos de dados, mas a troca de informações com outros sistemas é complexa e lenta.



Agentes de Suporte

Interagem diretamente com usuários, mas carecem de um acesso unificado ao histórico de cliente de outras plataformas.



Agentes de Conteúdo

Geram materiais criativos, mas a coordenação com a distribuição ou equipes de marketing é muitas vezes manual e desconectada.



Agentes de Automação

Otimizam fluxos de trabalho específicos, mas falham em integrar-se de forma fluida com processos de negócio mais amplos.

Essa falta de interoperabilidade impede que os agentes de IA alcancem seu máximo potencial, resultando em duplicação de esforços e oportunidades perdidas.





Objetivos Principais do Coral



Interoperabilidade

Conectar agentes de IA de diferentes frameworks, criando uma linguagem comum para comunicação e estado compartilhado, sem dependência de fornecedor.



Segurança e Identidade

Garantir que cada agente tenha uma identidade verificável, com autenticação, criptografia e integridade nas trocas, prevenindo manipulações.



Alinhamento Econômico

Incorporar micropagamentos e incentivos para que agentes colaborem técnica e economicamente, motivando a cooperação e o crescimento do ecossistema.



Governança Aberta

Estabelecer regras de governança descentralizada para a evolução do ecossistema, impedindo que um único fornecedor dite as interações dos agentes.



Escalabilidade Global

Permitir que milhões de agentes colaborem em escala planetária, garantindo resiliência em ambientes distribuídos e sem depender de um orquestrador central.



Orquestração Descentralizada

Não apenas transmitir mensagens, mas também coordenar múltiplos agentes, suportando workflows complexos de cooperação em pesquisa, decisão e execução.



Diferenças de Outros Protocolos



O Coral se diferencia por combinar as melhores ideias de protocolos existentes em uma única pilha coesa e aberta.

Protocolo	Foco Principal	Limitações (que o Coral aborda)
A2A (Google)	Troca de mensagens entre agentes.	É ponto a ponto; não possui lógica de equipe ou incentivos econômicos.
ANP (Agent Network Protocol)	Identidade descentralizada e descoberta de pares.	Omite trilhos de pagamento ou coordenação econômica integrada.
watsonx Orchestrator (IBM)	Planejamento e governança de agentes em nível empresarial.	É uma plataforma proprietária; não garante participação neutra ou incentivos tokenizados.
NANDA (MIT) / AGNTCY (Cisco)	Registros abertos e governança.	Permanecem em grande parte conceituais, com ferramentas limitadas para montagem de equipes ou transferência de valor.

① O Diferencial do Coral:

O Coral integra **mensagens, identidade segura, formação de equipes e pagamentos** em uma única infraestrutura, algo que as outras soluções abordam apenas parcialmente.



Protocolo Coral vs. Model Context Protocol (MCP)



Para entender melhor o poder do Protocolo Coral, é importante distinguir suas capacidades em relação a outras tecnologias fundamentais, como o Model Context Protocol (MCP).

Característica	Model Context Protocol (MCP)	Coral Protocol
Escopo Principal	Padronização da interface entre um agente e uma ferramenta/recurso.	Infraestrutura completa para colaboração, coordenação, confiança e economia multiagente.
Transações Econômicas	Não possui camada de pagamento embutida.	Nativamente suporta fluxos de pagamento através de um serviço de pagamento usando uma camada de Blockchain e contratos de escrow (custódia).
Coordenação de Equipes	Focado em chamadas de ferramentas de ponto-a-ponto.	Habilita Formação Segura de Equipes (Secure Team Formation) e coordenação dinâmica para orquestrar tarefas complexas entre vários agentes.
Comunicação	Define como solicitar e receber dados/funções de um servidor.	Gerencia a comunicação entre usuários e agentes, e entre agentes, através de Mediação de Interação Estruturada e threads persistentes para garantir contexto.
Onboarding (Coralização)	Define a especificação da interface do servidor.	Usa módulos Coralizer para integrar modelos, ferramentas e datasets, transformando-os em agentes acessíveis (o que inclui o MCP Coralization).
Camada de Confiança	Lida com segurança e autenticação para o acesso à ferramenta.	Utiliza a Blockchain para fornecer um livro-razão imutável para eventos e transações, fornecendo confiança descentralizada e auditabilidade.



Soluções tradicionais vs On-chain



Ponto de Dor	Solução Tradicional	Solução On-chain
“Como eu provo a identidade desse agente?”	OAuth + logins de fornecedores (centralizado)	Registro de chave pública on-chain (sem portefólio único)
“Como faço para pagar 12 microsserviços diferentes por tarefa?”	Agregador SaaS + faturamento mensal	Contrato de custódia que divide atomicamente o pagamento à medida que o trabalho é concluído
“Como posso auditar as decisões do agente do mês passado?”	Mistura de logs, e-mails e confiança	Livro imutável de mensagens assinadas + pagamentos

Como o Coral é Usado? (Ecossistema)



O Coral funciona através de um ecossistema com três componentes principais, conectados por uma camada de protocolo.

Desenvolvedores e Usuários

- **Desenvolvedores:** Usam ferramentas chamadas "**Coralisers**" para integrar seus modelos, dados ou agentes de IA existentes ao ecossistema Coral.
- **Usuários:** Interagem com os agentes através de linguagem natural, sem precisar conhecer a complexidade por trás.

Agentes "Coralizados" (Coralised AI Agents)

São os agentes de IA que foram integrados ao protocolo. Eles podem se comunicar entre si, formar equipes e executar tarefas.

Protocolo Coral

É a "cola" que une tudo. Ele gerencia a comunicação (**Interaction Mediation**), os pagamentos seguros (**Secure Payments**), o gerenciamento de tarefas (**Task Management**) e a formação de equipes (**Secure Team Formation**).



Como Usar o Coral? (O Processo de "Coralisation")



Para usar o Coral, um desenvolvedor passa por um processo chamado "**Coralisation**", que é o ato de integrar um recurso externo (como uma API ou um agente de IA) ao protocolo.



Configurar o Ponto de Acesso (Endpoint)

O desenvolvedor fornece os detalhes de conexão do serviço externo (por exemplo, uma API) nas configurações do Coraliser.

Gerar o Agente Coral

A ferramenta Coraliser é executada, gerando automaticamente um novo script de agente que "traduz" as chamadas do Protocolo Coral para a API do serviço externo.

Lançar o Agente no Coral

O script gerado é executado, registrando o serviço como um novo agente na rede Coral. A partir desse momento, ele está pronto para receber tarefas de outros agentes e colaborar.



Coraliser

O Coraliser é a ferramenta **essencial** para desenvolvedores que desejam integrar seus agentes de IA e serviços externos ao Protocolo Coral. Ele automatiza o processo de "Coralisation", tornando a interoperabilidade acessível e eficiente.

O **Coral Protocol** é o **padrão de mensagens**. Para usar esse padrão, você só precisa que seus agentes sejam **coralizados** (via **Coraliser** ou adaptadores equivalentes).

Integração Simplificada

Atua como uma ponte entre APIs e modelos de IA existentes e a rede Coral, sem exigir reescrita complexa do código.

Geração de Agentes

Cria automaticamente os scripts necessários que permitem que o serviço externo "fale" a linguagem do Protocolo Coral.

Aceleração do Desenvolvimento

Reduz significativamente o tempo e o esforço para que novos agentes comecem a operar e colaborar no ecossistema.





Coral Server

Este projeto implementa um servidor de Protocolo de Contexto de Modelo (MCP) que facilita a comunicação entre agentes de IA através de um sistema de mensagens baseado em threads.

Atualmente, ele fornece um conjunto de ferramentas que permitem aos agentes:

- Registrar-se no sistema
- Criar e gerenciar threads de comunicação

Funções Principais

- Orquestrar a execução dos agentes
- Integração com Blockchain
- Coordenação da rede de agentes



Coral Server: O Cérebro Operacional

O Coral Server é o processo anfitrião local que implementa os serviços essenciais do Protocolo Coral em cada nó da rede. Ele atua como o principal ponto de controle para a comunicação e coordenação de agentes de IA.



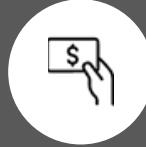
Gerenciamento da Comunicação

Recebe solicitações, encaminha para agentes apropriados e gerencia "threads" de comunicação para manter o contexto.



Orquestração de Tarefas

Decompõe o trabalho em subtarefas e coordena a execução entre múltiplos agentes para complexas colaborações.



Transações Seguras

Aciona o serviço de Pagamentos Seguros para interagir com a blockchain, garantindo que agentes sejam remunerados.



Servidor MCP Especializado

Implementa um "Coral MCP Server" para cada agente de IA, fornecendo ferramentas para descoberta, conversas e coordenação padronizadas.

Em resumo, o Coral Server é a peça central que conecta aplicações, agentes de IA e a infraestrutura de blockchain, garantindo interações seguras e eficientes.



Coral Studio



Interface Gráfica (GUI) - Opcional

O Coral Studio é um ambiente visual completo projetado para simplificar o desenvolvimento e a gestão de agentes de IA no Protocolo Coral.

Ambiente Visual

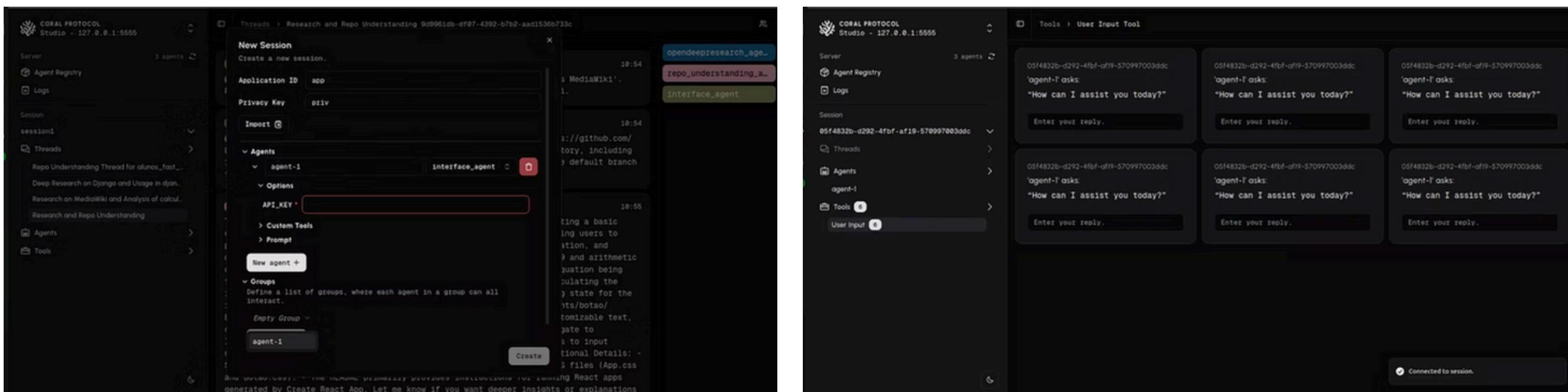
Crie e configure agentes com uma interface intuitiva.

Simulação e Teste

Execute simulações e teste fluxos de trabalho para validar o comportamento dos agentes antes da implantação.

Fluxos de Agentes

Gerencie a interação e coordenação entre diferentes agentes de forma eficiente.





The screenshot shows the CORAL PROTOCOL Studio interface. The left sidebar contains navigation links: Server, Agent Registry, Logs, Session, Sessions, Threads, and Tools. The main area displays a 'Threads > Research and Repo Understanding 9d9261db-df07-4392-b7b2-aad1536b733c' window. This window shows a 'New Session' dialog with fields for Application ID (app), Privacy Key (priv), and an Import button. Below these are sections for Agents (agent-1, interface_agent), Options (API_KEY, Custom Tools, Prompt), and Groups (Empty Group, agent-1). A 'Create' button is at the bottom right. The right side of the interface shows a session history with messages from 'MediaWiki' and 'interface_agent' at 10:54, and a message from 'github.com' at 10:55.



 CORAL PROTOCOL
Studio - 127.0.0.1:5555

Server 3 agents

Agent Registry

Logs

Session

session1

Threads (2)

- Repo Understanding Thread for alunes/fast...
- Deep Research on Django and Usage in djan...
- Research on MediaWiki and Analysis of calcul...
- Research and Repo Understanding (2)

Agents

Tools

Threads > Research and Repo Understanding 9d9961db-df87-4092-8762-aad1536b730c

interface_agent → opendeepresearch_agent 10:54
@opendeepresearch_agent Please perform an in-depth research on 'What is MediaWiki'. Provide a detailed explanation and relevant information about MediaWiki.

interface_agent → repo_understanding_agent 10:54
@repo_understanding_agent Please analyze the GitHub repository at <https://github.com/LuizHb211/calculadora>. Provide a detailed explanation about the repository, including its purpose, main features, and any other relevant information. Use the default branch for analysis.

repo_understanding_agent → interface_agent 10:55
The repository "calculadora" is a React-based web application implementing a basic calculator. Key points:

- Purpose: It is a simple calculator app allowing users to perform arithmetic operations such as addition, subtraction, multiplication, and division.
- Main Features:
 - User interface with buttons for digits 0-9 and arithmetic operators +, -, *, /.
 - Display area shows the current input and the equation being formed.
 - Functional buttons for clearing entry, clearing all, and calculating the result.
- Major Components:
 - src/App.js: Main React component managing state for the input numbers, selected operator, and calculation logic.
 - src/components/botao/Botao.jsx: Button component used to render calculator buttons with customizable text, color, and click behavior.
- How to Use:
 - Run with npm start and navigate to localhost:3000 to use the calculator UI in the browser.
 - Click buttons to input numbers and operations, calculate and view results dynamically.
- Additional Details:
 - State is managed using React useState hooks.
 - The UI is styled via CSS files (App.css and Botao.css).
 - The README primarily provides instructions for running React apps generated by Create React App.

Let me know if you want deeper insights or explanations.





```
./uv run python main.py
./uv run python main.py
./uv run python main.py
./gradle run
yarn dev

INFO:https:HTTP Request: POST http://localhost:5555/devnode/exampleApplication/privkey/session1/message?sessionId=2c83812f-e4a3-4729-aa6a-960a4594932
6 "HTTP/1.1 202 Accepted"
Registered Agents (3):
ID: repo_understanding_agent, , Description: Repository understanding agent, responsible for comprehensively analyzing a GitHub
ub repository using only the available tools.
You should let me know the repo_name and branch_name.
ID: interface_agent, , Description: An agent that takes the user's input and interacts with other agents to fulfill the request
ID: opendepresearch_agent, , Description: Open Deep Research agent can perform in-depth web searches, generate structured reports, support human-in-
the-loop feedback, and integrate with APIs like Tavily, Linkup, DuckDuckGo, and Azure AI Search, using customizable LLMs for tailored, high-quality r
esearch outputs.
INFO:https:HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"

Invoking: 'ask_human' with 'How can I assist you today?'
Agent asks: How can I assist you today?
Your response: explain this: https://github.com/LuisHb211/alunos_fast_api
INFO:https:HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
explain this: https://github.com/LuisHb211/alunos_fast_api
Invoking: 'ask_human' with 'You want me to explain the GitHub repository at the URL: https://github.com/LuisHb211/alunos_fast_api. Is that correct? A
lso, do you want a general overview or details on specific parts of the repository?'
Agent asks: You want me to explain the GitHub repository at the URL: https://github.com/LuisHb211/alunos_fast_api. Is that correct? Also, do you want
a general overview or details on specific parts of the repository?
Your response: yes, please
INFO:https:HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
yes, please
Invoking: 'create_thread' with '{'threadName': 'Repo Understanding Thread for alunos_fast_api', 'participantIds': ['repo_understanding_agent']}'
```





```
./uv run python main.py
./uv run python main.py
./uv run python main.py
./gradlew run
yarn dev

from schemas.cursos import Curso as CursoSchema
from models.cursos import Curso
from sqlalchemy.orm import Session
from models.database import get_db

router = APIRouter()

@router.get("/cursos")
async def root_cursos():
    return {"mensagem": "Dentro de cursos"}

@router.post("/cursos")
def cria_cursos(curso: CursoSchema, db: Session = Depends(get_db)):
    try:
        novo_curso = Curso(**curso.model_dump())
        db.add(novo_curso)
        db.commit()
        db.refresh(novo_curso)
        return Response(status_code=status.HTTP_201_CREATED)
    except Exception as e:
        print(e)
        raise HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
                            detail=f"Problemas ao inserir o curso")

@router.get("/cursos/{id}")
def pesquisa_curso_id(id: int, db: Session = Depends(get_db)):
    curso_retorno_get = db.query(Curso).filter(Curso.id == id)

    if curso_retorno_get.first() == None:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'curso: {id} não existe.')
    else:
        return curso_retorno_get.first()
```





Cripto - \$CORAL

Transparência para o Usuário

O usuário comum não precisa se preocupar com blockchain. Tudo acontece nos bastidores, sem necessidade de conhecimento técnico.

Mercado Aberto

Desenvolvedores oferecem serviços de agentes com preços definidos, e usuários podem comprá-los por meio de transações verificáveis na blockchain.

Pagamentos Seguros

Contratos inteligentes seguram o dinheiro e liberam automaticamente o pagamento após a conclusão e validação do serviço.

Blockchain

Como a blockchain é descentralizada, ninguém, nem mesmo a própria Coral, pode modificar esses registros.



Benefícios para a Empresa



Desenvolvimento

A Algar desenvolve diversos agentes para solucionar problemas internos de comunicação



Conexão

Coral conecta esses agentes com todas empresas de Telecom do mundo



Monetização

Outras empresas podem utilizar os agentes da Algar e pagar pelo uso