Lore N.A.: Uma Análise Estratégica e Técnica para a Engenharia de um Microverso Digital

Introdução: Do Terrário Digital ao Instrumento Científico

O projeto Lore N.A. (Lore Neural Agents) representa uma proposta de notável ambição e sofisticação. Transcende a concepção tradicional de desenvolvimento de software para se posicionar como um programa de pesquisa fundamental nos campos da Vida Artificial (ALife) e da Economia Computacional Baseada em Agentes (ACE). A visão central de criar um "universo numa garrafa" — um ecossistema digital autossuficiente para observar o comportamento emergente — alinha-se com as fronteiras da ciência de sistemas complexos. O objetivo não é a criação de uma aplicação com uma utilidade predefinida, mas sim a construção de um instrumento científico cujo principal produto é o conhecimento: a "lore" que emerge das interações imprevisíveis dos seus habitantes digitais.[1]

A tese central do projeto — "A 'lore'... não é algo que nós escrevemos, mas sim o que emerge" — é particularmente inovadora. Ela propõe uma mudança de paradigma, afastando-se da simulação como mera replicação de sistemas conhecidos e aproximando-se da simulação como uma forma de etnografia computacional generativa. O sucesso do projeto não será medido pela sua capacidade de gerar lucro simulado, mas pela complexidade e surpresa geradas pelo comportamento dos seus habitantes. O verdadeiro valor reside nos dados e nos insights que podem ser extraídos da observação passiva deste terrário digital.[2]

Este relatório serve como uma análise estratégica e uma revisão técnica aprofundada da proposta Lore N.A. O seu propósito é fornecer uma validação rigorosa da visão do projeto, fundamentando-a em teorias científicas estabelecidas, e oferecer um roteiro detalhado para a sua implementação. A análise está estruturada para abordar sequencialmente os pilares do projeto. A Secção 1 estabelecerá as fundações filosóficas e científicas, contextualizando Lore N.A. no legado da Vida Artificial e da Modelagem Baseada em Agentes. A Secção 2 refinará a arquitetura técnica proposta, traduzindo os conceitos em padrões de design de software robustos e escaláveis. A Secção 3 aprofundará o design dos habitantes do microverso, os Agentes Neurais, propondo uma taxonomia para as suas arquiteturas cognitivas. A Secção 4 detalhará o ambiente económico e, crucialmente, a "Sala de Controle" para a sua observação. Finalmente, a Secção 5 apresentará um plano de implementação pragmático e faseado,

focado na entrega de um "Microverso Mínimo Viável" que possa começar a gerar os fenómenos emergentes desejados.

Secção 1: As Fundações Filosóficas e Científicas de um Universo Criado

Para apreciar plenamente o potencial de Lore N.A., é essencial situá-lo no seu contexto intelectual. O projeto não surge no vácuo; é a continuação de décadas de investigação em sistemas complexos, vida artificial e modelagem computacional. Esta secção estabelece essa herança, fornecendo a base teórica que valida as premissas do projeto e informa a sua implementação.

1.1 A Herança da Vida Artificial (ALife)

O conceito central de Lore N.A. como um "terrário digital" povoado por "formas de vida artificiais" insere-se diretamente no campo da Vida Artificial (ALife). Definida pelo cientista da computação Christopher Langton em 1986, a ALife é a área de estudo que examina sistemas relacionados com a vida natural, os seus processos e a sua evolução, através do uso de modelos computacionais, robótica e bioquímica.[3] Os investigadores de ALife estudam a biologia tradicional tentando recriar aspetos de fenómenos biológicos em ambientes artificiais para obter uma compreensão mais profunda dos processos fundamentais dos sistemas vivos.[3]

Dentro da ALife, existem duas posições filosóficas distintas sobre o potencial do campo. A posição da **"strong alife"** (ALife forte) afirma que "a vida é um processo que pode ser abstraído de qualquer meio em particular".[3] Sob esta ótica, uma simulação suficientemente complexa e precisa dos processos da vida não é apenas um modelo, mas uma instância real de vida sintética. Um exemplo notável é o programa Tierra de Tom Ray, cujo criador declarou não estar a simular a vida, mas a sintetizá-la.[3] Em contraste, a posição da **"weak alife"** (ALife fraca) nega a possibilidade de gerar um "processo vivo" fora de uma solução química. Os seus investigadores procuram simular processos de vida para compreender os mecanismos subjacentes aos fenómenos biológicos, sem afirmar que criaram vida real.[3]

A formulação do projeto Lore N.A. alinha-se de forma inequívoca com a perspetiva da "weak alife". A declaração de missão é clara: "o objetivo não é construir uma aplicação, mas sim acender a faísca da criação e então, como arquitetos e cientistas, nos afastarmos para observar o que acontece". O papel dos criadores é o de "observadores passivos e cientistas de dados", e o sucesso é alcançado "quando o comportamento de seus habitantes se tornar complexo o suficiente para nos surpreender". Esta postura define claramente os critérios de sucesso do projeto. O objetivo não é passar num "teste de Turing para a vida", mas sim criar

uma simulação suficientemente rica para que as suas propriedades emergentes sejam não-ó bvias e gerem dados valiosos e analisáveis sobre a dinâmica socioeconómica. Isto justifica o foco na "Sala de Controle" (os dashboards do Grafana) como a verdadeira interface de utilizador do projeto, o observatório através do qual a história do microverso é lida.
Historicamente, Lore N.A. baseia-se em predecessores intelectuais fundamentais. O mais famoso é o **Jogo da Vida de Conway**, um autómato celular concebido em 1970.[7] Com apenas quatro regras simples aplicadas a uma grelha de células, o Jogo da Vida demonstra como padrões globais complexos e imprevisíveis — como "gliders" (planadores) e "oscillators" (osciladores) — podem emergir e auto-organizar-se sem um designer central.[7] Este é um paralelo direto ao objetivo de Lore N.A. de observar fenómenos económicos complexos, como bolhas, a emergir de regras simples de compra e venda dos agentes. O projeto também se insere numa comunidade de investigação contemporânea e vibrante, com projetos de código aberto como
asal (Automating the Search for Artificial Life), que utiliza modelos de fundação para descobrir novas formas de vida em substratos como Lenia e Boids [10];
Evolve, um simulador que utiliza uma linguagem genética sofisticada para evoluir replicadores complexos [11]; e
project-origin, um simulador focado na emergência da inteligência (Noogénese).[12]

1.2 O Paradigma da Modelagem Baseada em Agentes (ABM)

A metodologia que sustenta a totalidade do projeto Lore N.A. é a Modelagem Baseada em Agentes (ABM - Agent-Based Modeling). A ABM é uma abordagem de simulação computacional "bottom-up" (de baixo para cima) utilizada para estudar as interações entre entidades autónomas, ou "agentes".[2] A característica definidora da ABM é que ela estabelece uma correspondência direta e explícita entre as unidades individuais do sistema a ser modelado e os agentes que as representam no modelo.[13] Os componentes centrais de uma ABM, conforme descritos na literatura, são:

- **Agentes:** Entidades autónomas com atributos (estado interno) e regras de comportamento (instruções).[2] Isto corresponde diretamente aos "Agentes Neurais" de Lore N.A.
- **Ambiente:** O contexto ou espaço no qual os agentes operam e interagem.[1] Isto é o "Mercado do Limbo".
- **Interações:** As regras que governam como os agentes se afetam uns aos outros e ao ambiente.[14] No caso de Lore N.A., estas são as transações económicas e os mecanismos de influência social.

A ABM é o paradigma correto para este projeto porque é especificamente concebida para explorar sistemas complexos que exibem não-independência de indivíduos, loops de feedback e efeitos emergentes — exatamente os fenómenos que Lore N.A. procura investigar.[2] Uma das suas maiores vantagens é a capacidade de modelar agentes

heterogéneos, ou seja, agentes com diferentes atributos e regras de decisão, o que é uma característica central das "personalidades" propostas para os Agentes Neurais.[13]
Uma limitação frequentemente citada da ABM é a dificuldade em encontrar dados do mundo real para parametrizar e validar o comportamento dos agentes.[2] No entanto, o design de Lore N.A. transforma esta potencial fraqueza numa força fundamental. As ABMs tradicionais tentam modelar uma realidade
existente (por exemplo, o tráfego de uma cidade) e, para isso, necessitam de dados empíricos sobre o comportamento dos condutores, que podem ser escassos ou incompletos. Lore N.A., pelo contrário, não está a modelar uma realidade existente; está a *criar* uma nova. As "leis da física" são o código, e as condições iniciais do universo são definidas pelo script genesis.py. Consequentemente, o projeto não sofre da falta de dados do mundo real para calibração. Ele *gera a sua própria verdade fundamental*. A consistência interna da simulação é o único requisito. O objetivo é observar as consequências de um *dado* conjunto de parâmetros iniciais, não replicar perfeitamente um sistema externo. Isto reforça o papel dos criadores como "observadores passivos" e o foco na análise da história *interna* da simulação, o seu caminho dinâmico.[17] O script
genesis.py não é apenas um inicializador; é o "Big Bang" deste microverso, definindo as constantes fundamentais a partir das quais toda a complexidade subsequente emergirá.

1.3 O Fenómeno da Emergência: O "Produto" do Microverso

O verdadeiro "produto" de Lore N.A. é a observação da emergência. A emergência é o surgimento de estruturas, padrões e propriedades macroscópicas novas e coerentes que não podem ser explicadas ou reduzidas às propriedades dos seus componentes individuais (microscópicos).[18] É o fenómeno em que o todo se torna "mais do que a soma das suas partes".[19]
Exemplos clássicos ajudam a tornar este conceito concreto. O **algoritmo Boids** de Craig Reynolds, de 1986, demonstra como um comportamento de bando complexo e realista emerge de apenas três regras simples aplicadas a cada "boid" individual: separação (evitar colisões), alinhamento (corresponder à velocidade dos vizinhos) e coesão (mover-se em direção ao centro de massa dos vizinhos).[20] Nenhum boid tem conhecimento do bando inteiro; o comportamento global emerge das interações locais. Da mesma forma, no
Jogo da Vida de Conway, estruturas complexas como os "gliders" surgem espontaneamente das quatro regras locais, um exemplo de auto-organização na ausência de um designer.[7]
Os resultados desejados de Lore N.A. — "bolhas económicas", "dinâmica de tribos" — são fenómenos emergentes clássicos. Um conceito crucial para entender como estes fenómenos se podem tornar autossustentáveis é a "causalidade descendente" (downward causation).[19] Este princípio descreve como o sistema de nível macro emergente começa a restringir e a influenciar o comportamento dos agentes de nível micro. Isto cria um poderoso loop de feedback que é central para a história que o microverso contará. O processo pode ser visualizado da seguinte forma:

1. **Causalidade Ascendente:** Inicialmente, as ações individuais dos agentes (por exemplo, comprar um artefacto) agregam-se para criar um estado de mercado (por exemplo, um aumento no preço do artefacto).
2. **Emergência do Macro-Fenómeno:** Este novo estado de mercado — a "bolha" emergente — torna-se parte do ambiente que todos os agentes percebem.
3. **Causalidade Descendente:** Os agentes (especialmente os do tipo "especulador" ou "influenciável") reagem a esta bolha, alterando o seu comportamento *devido à existência da bolha*. O padrão de nível macro está agora a influenciar as decisões de nível micro.
4. **Loop de Feedback:** Esta reação em cadeia pode amplificar ainda mais a bolha ou levar ao seu colapso, criando um ciclo dinâmico.

Portanto, o projeto não se limita a observar padrões estáticos; trata-se de testemunhar todo o ciclo de vida da emergência, desde a sua génese em interações locais até à sua maturação como uma força a nível do sistema que molda os próprios agentes que a criaram. Esta é a narrativa, a "lore", que o microverso irá gerar autonomamente. Um desafio fundamental do design será visualizar este loop de feedback de forma eficaz nos dashboards de observação.

Secção 2: A Arquitetura de uma Realidade Simulada: Do Monólito ao Micro-Ecossistema

Esta secção traduz a visão conceptual de Lore N.A. num projeto arquitetónico rigoroso e alinhado com as melhores práticas da indústria. Valida a pilha tecnológica proposta e mapeia as "Camadas" abstratas do projeto em padrões de design de software formais, fornecendo uma estrutura clara e robusta para o desenvolvimento.

2.1 Validação da Pilha Tecnológica Fundacional

A escolha de uma pilha tecnológica moderna e contentorizada é ideal para os requisitos de isolamento, escalabilidade e persistência de Lore N.A. A seleção de ferramentas é robusta e apropriada para a tarefa.

- **Python (com FastAPI):** Uma excelente escolha para a API ("Nave") e para a implementação dos próprios agentes. O ecossistema Python é rico em bibliotecas de ciência de dados (como a LeIA, para análise de sentimento [23]) e de desenvolvimento web.[24] O FastAPI, em particular, é conhecido pela sua alta performance, validação de dados automática via Pydantic e facilidade de uso, o que o torna ideal para construir a API com que os agentes irão interagir.[24]
- **PostgreSQL:** A escolha perfeita para o "livro-razão" do microverso. A sua natureza relacional, o suporte para tipos de dados avançados como JSONB (ideal para armazenar a "memória" do agente ou estados complexos) e a sua integridade

transacional (conformidade ACID) são essenciais para manter um registo persistente e fiável de todos os agentes, artefactos e transações.
- **Docker/Kubernetes:** A "bateria isolada do Rick" é um pilar não-negociável para um projeto desta complexidade. A contentorização de cada componente — API, base de dados, executores de agentes, dashboards — garante a consistência do ambiente, simplifica a implantação e permite a escalabilidade horizontal à medida que a complexidade da simulação aumenta.[24] Esta abordagem isola falhas e permite que diferentes partes do sistema sejam atualizadas de forma independente.
- **Git & VSCode:** Ferramentas padrão da indústria que representam as melhores práticas para o controlo de versões e desenvolvimento de software profissional.

2.2 Uma Arquitetura em Camadas para um Mundo em Camadas

A estrutura conceptual do utilizador, com as suas cinco ideias de negócio interligadas, pode ser formalizada utilizando o padrão de **Arquitetura em Camadas** (ou Arquitetura N-Tier). Este padrão organiza uma aplicação em camadas horizontais distintas, cada uma com um conjunto específico de responsabilidades, promovendo a separação de preocupações (separation of concerns) e a manutenibilidade.[27] As "Camadas" de Lore N.A. podem ser mapeadas para este padrão formal da seguinte forma:

- **Camada 1 (Mercado Efêmero) -> Camada de Dados/Persistência:** Esta é a camada fundamental que gere o estado dos "átomos" do universo: os produtos e artefactos. É diretamente implementada pela base de dados PostgreSQL, que armazena e recupera os dados.[27]
- **Camada 2 (Ateliê Algorítmico) & Camada 3 (Caixa Coletiva) -> Camada de Lógica de Negócio/Domínio:** Estas camadas contêm as regras de negócio e os fluxos de trabalho centrais da simulação. Elas definem como os agentes interagem com os artefactos base para criar produtos e serviços de maior valor agregado. Esta lógica reside na aplicação FastAPI.[27]
- **Camada 5 (Logos Prime) -> Camada de Serviço/Infraestrutura:** Esta camada simula as restrições operacionais do "mundo real", como tempos de entrega, custos de frete e falhas. Atua como um serviço especializado que outras camadas invocam para executar ações, introduzindo constrangimentos realistas no sistema.
- **Camada 4 (Neurônio S.A.) -> Camada de Inteligência/Análise:** Este é o "cérebro" que observa todas as outras camadas. Não é uma camada de aplicação tradicional, mas sim um conjunto de scripts de análise de dados, modelos de Machine Learning e a pilha de observação Grafana/n8n. Consome dados das outras camadas para produzir insights e, potencialmente, para impulsionar mudanças estratégicas na simulação (uma forma de causalidade descendente).
- **A "Casca Pública" -> Camada de Apresentação:** Esta é a interface com o mundo exterior, incluindo a landing page estática e, mais importante, os dashboards do Grafana, que são a principal janela para o microverso.[27]

2.3 Das Camadas aos Microsserviços: Uma Abordagem Orientada a Eventos

Embora uma arquitetura em camadas seja um excelente ponto de partida, um microverso verdadeiramente complexo e escalável beneficiará da evolução para uma **Arquitetura de Microsserviços**. O plano de contentorizar cada componente já aponta nessa direção. As interações entre as camadas são mais bem geridas de forma assíncrona, o que leva naturalmente a uma **Arquitetura Orientada a Eventos (EDA - Event-Driven Architecture)**.
Uma aplicação monolítica em camadas pode tornar-se um gargalo de escalabilidade. Se o serviço de logística "Logos Prime" for computacionalmente intensivo, poderia abrandar toda a aplicação. Numa arquitetura de microsserviços, ele pode ser escalado de forma independente.[31] Além disso, as chamadas síncronas diretas entre as camadas criam um acoplamento forte; se o serviço de análise "Neurônio S.A." estiver em manutenção, não deveria impedir que os agentes realizassem transações no "Mercado Efêmero".
A EDA resolve estes problemas. Em vez de chamadas diretas, os serviços comunicam publicando e subscrevendo eventos num barramento de eventos (event bus).[31] Por exemplo:

1. Quando um agente compra um artefacto, o microsserviço da API publica um evento TRANSACTION_COMPLETED.
2. O microsserviço "Logos Prime" subscreve este evento para iniciar a simulação da entrega.
3. O microsserviço "Neurônio S.A." também subscreve este evento para atualizar as suas análises.

Este desacoplamento melhora a resiliência, a escalabilidade e a manutenibilidade do sistema.[33] Alinha-se perfeitamente com a natureza assíncrona e baseada em passos de tempo de uma ABM. Portanto, uma arquitetura refinada para Lore N.A. consistiria em cada "Camada" ser implementada como um ou mais microsserviços distintos, comunicando através de um barramento de eventos (como RabbitMQ ou Kafka, embora para uma simplicidade inicial, uma fila baseada em PostgreSQL possa ser suficiente).

2.4 O Maestro da Orquestra: O Papel do n8n

A identificação do n8n como o "orquestrador" do sistema é perspicaz. O n8n pode funcionar como o sistema nervoso central para processos de meta-nível, orquestração de fluxos de trabalho e automação.[34]
As suas principais tarefas de orquestração em Lore N.A. seriam:

- **Agendador de Agentes (O "Grande Relógio"):** Esta é a sua função primária. Um fluxo de trabalho no n8n, acionado por um agendamento (por exemplo, a cada segundo, representando um "tick" ou um "dia" na simulação), seria responsável por "acordar" os agentes. Este fluxo de trabalho pode consultar a base de dados PostgreSQL para obter

a lista de agentes ativos e, em seguida, chamar o endpoint da API que aciona a função step() de cada agente. Esta abordagem é mais robusta, observável e fácil de gerir do que um simples cron job.

- **Orquestração de Microsserviços:** O n8n é perfeitamente adequado para orquestrar os fluxos de trabalho orientados a eventos propostos acima. Pode ouvir eventos, acionar outros serviços, encadear chamadas de API e gerir lógicas complexas de forma visual.[34] Por exemplo, pode implementar padrões de transações distribuídas como o padrão Saga para garantir a consistência entre microsserviços.[33]
- **Pipelines de Dados:** O n8n pode executar fluxos de trabalho noturnos ou horários para realizar a limpeza, agregação e pré-computação de dados para os dashboards do Grafana. Isto reduz a carga de consulta na base de dados principal durante a observação em tempo real.[36]
- **Simulação de Eventos Externos:** O n8n pode ser usado para injetar "choques exógenos" no sistema, um conceito comum em simulações económicas.[38] Um fluxo de trabalho no n8n poderia simular um "crash de mercado" alterando drasticamente os preços dos artefactos através de uma chamada à API, ou simular um "evento de notícias" que aumenta temporariamente o sentimento de uma determinada tribo de agentes.

A tabela seguinte resume a arquitetura proposta, servindo como um plano mestre para o desenvolvimento.

Tabela 2.1: Plano Arquitetónico de Lore N.A.

Camada (Conceito)	Camada Arquitetónica Formal	Padrão Arquitetónico	Tecnologia Primária	Responsabilidade Chave	Exemplo de Fluxo de Dados (Orientado a Eventos)
Fundação do Microverso	Camada de Infraestrutura	Contentorização	Docker/Kubernetes	Fornece o ambiente de execução isolado e escalável para todos os componentes.	N/A (É o ambiente)
Mercado Efêmero	Camada de Persistência	Base de Dados Relacional	PostgreSQL	Armazena o estado persistente de todos os agentes, artefactos, transações e estados emocionais. É	Publica: AGENT_STATE_UPDATED, ARTIFACT_CREATED

				o "livro-razão" do universo.	
A "Nave" (API)	Camada de Lógica de Aplicação	Microsserviço (API Gateway)	Python (FastAPI)	Expõe os endpoints para os agentes (e observadores) interagirem com o universo. Valida e processa todas as ações.	Consome: Ações dos agentes. Publica: TRANSACTION_COMPLETED
Ateliê / Caixa Coletiva	Camada de Lógica de Negócio	Microsserviços	Python (FastAPI)	Implementa as regras de negócio para combinação de artefactos e serviços de subscrição.	Consome: TRANSACTION_COMPLETED. Publica: CUSTOM_PRODUCT_CREATED
Logos Prime	Camada de Serviço	Microsserviço (Simulador)	Python	Simula as restrições operacionais (tempo, custo, falhas) e o ciclo de vida dos artefactos.	Consome: TRANSACTION_COMPLETED. Publica: DELIVERY_COMPLETED, ARTIFACT_LIFECYCLE_ENDED
Neurônio S.A.	Camada de Inteligência/Análise	Ferramentas de Análise e Orquestração	Python, LeIA, n8n	Analisa o fluxo de dados, calcula o estado emocional dos agentes e orquestra os "ticks" da simulação e eventos externos.	Consome: DELIVERY_COMPLETED. Publica: SENTIMENT_SCORE_CALCULATED
Sala de Controle	Camada de Apresentação	Plataforma de Visualização	Grafana	Fornece os dashboards para observação em	Consome: Dados do PostgreSQL.

				tempo real dos fenómenos emergentes e da saúde do sistema.	

Secção 3: Os Habitantes: Uma Taxonomia da Cognição dos Agentes Neurais

Esta secção aprofunda o design dos habitantes do microverso: os Agentes Neurais. Transforma o conceito de "personalidade" numa arquitetura cognitiva concreta e implementável, fornecendo uma estrutura para criar uma população diversificada, que é a pré-condição para a emergência de fenómenos sociais complexos.

3.1 Formalizando o Design do Agente: O "ADN Digital"

Para estruturar o design dos agentes de forma rigorosa, pode-se adotar um framework formal baseado na investigação em agentes autónomos.[14] Cada agente em Lore N.A. é um objeto de software com os seguintes componentes:

- **Estado (Atributos):** As variáveis internas que definem o agente num dado momento. A lista proposta pelo utilizador pode ser formalizada da seguinte forma:
  - agent_id: Um identificador único (por exemplo, uma string ou UUID).
  - wallet_balance: Os recursos finitos do agente (um número de ponto flutuante).
  - memory: Um registo das ações passadas e dos seus resultados. No PostgreSQL, isto pode ser implementado eficientemente com um campo do tipo JSONB, permitindo armazenar uma lista estruturada de eventos passados.
  - emotional_state: Um objeto estruturado contendo as métricas dinâmicas da "personalidade" do agente, como sentiment_score, trust_score, e engagement_level (todos números de ponto flutuante).
  - personality_profile: Um conjunto de parâmetros estáticos, definidos no "nascimento" do agente, que constituem o seu arquétipo. Estes podem incluir risk_aversion, influenceability, frugality_score, etc.
- **Perceção (Sensing):** O mecanismo pelo qual o agente recolhe informação sobre o seu ambiente. Em Lore N.A., a perceção é primariamente realizada através de chamadas à API da "Nave" para obter o estado do mercado "Limbo". Por exemplo, uma chamada GET /artifacts para ver os artefactos disponíveis, os seus preços, popularidade, etc..[40]
- **Tomada de Decisão (Reasoning):** O "cérebro" do agente. É a lógica central que processa a informação sensorial (o estado do mercado) e o seu próprio estado interno (carteira, memória, emoções, perfil de personalidade) para escolher uma ação. É aqui

que a "personalidade" do agente se manifesta.[40]

- **Ação (Actuators):** A forma como o agente afeta o ambiente. Isto é conseguido através de chamadas à API para executar ações, como POST /transactions (com um payload para buy_artifact ou sell_artifact) ou, no futuro, POST /reviews.[40]

3.2 Engenharia de "Personalidades" de Agentes com Modelos Cognitivos

O desejo de criar tipos de agentes diversos como "frugais, especuladores, influenciadores, haters" pode ser alcançado através da implementação de diferentes modelos de tomada de decisão, conforme categorizados na investigação em IA.[42] Cada arquétipo de agente terá uma "arquitetura cognitiva" diferente.

- **Agentes de Reflexo Simples:** Estes agentes operam com base em regras simples do tipo se-então (if-then), reagindo apenas à perceção atual do ambiente, sem memória do passado.[43] Um agente
  "Frugal" poderia ser um agente de reflexo simples: a sua regra de decisão seria SE preco_artefacto < meu_limite_orcamental E preco_artefacto < media_mercado ENTAO comprar().
- **Agentes de Reflexo Baseados em Modelo:** Estes agentes mantêm um modelo interno do mundo, utilizando a memória para informar as suas decisões.[43] Um
  "Especulador" básico seria um agente baseado em modelo: SE lembro-me_que(preco_deste_artefacto_subiu_nos_ultimos_3_dias) E meu_trust_score > 0.7 ENTAO comprar(). A sua decisão não se baseia apenas no preço atual, mas na sua memória da trajetória do preço.
- **Agentes Baseados em Objetivos:** Estes agentes têm objetivos explícitos e planeiam sequências de ações para os alcançar.[42] Um
  "Influenciador" poderia ter o objetivo de MAXIMIZAR(minha_pontuacao_de_influencia). As suas ações (por exemplo, comprar um item que está a tornar-se popular) seriam escolhidas para avançar em direção a esse objetivo, mesmo que o item seja caro.
- **Agentes Baseados em Utilidade:** Estes são os mais sofisticados, avaliando a "desejabilidade" ou "utilidade" de diferentes resultados para maximizar a sua satisfação geral.[42] Um
  "Investidor Equilibrado" usaria uma função de utilidade que pondera o lucro potencial de uma compra contra o seu risco (volatilidade), influenciado pelo seu traço de personalidade risk_aversion. Um **"Hater"**, por outro lado, poderia ter uma função de utilidade perversa onde utilidade = -1 * sentimento_geral_do_ecossistema. Isto o levaria a tomar ações (como comprar e depois deixar uma "review" negativa simulada) que prejudicam o sistema, pois isso maximiza a sua própria utilidade.
- **Agentes de Aprendizagem:** O estado da arte e um objetivo futuro para Lore N.A. Estes agentes podem adaptar as suas estratégias ao longo do tempo através da

aprendizagem por reforço (reinforcement learning).[39] Um agente de aprendizagem poderia começar com uma estratégia "Frugal", mas descobrir através da experiência (recompensas na forma de aumento de recursos) que uma estratégia "Especuladora" é mais bem-sucedida no ambiente atual, e assim adaptar o seu comportamento.

3.3 A Mecânica do "Estado Emocional": O Loop de Feedback de Sentimento

Este é um dos mecanismos mais inovadores e cruciais da proposta Lore N.A. A sua implementação fecha o loop de feedback que permite o comportamento emergente complexo.

- **Etapa A: Simulação da "Review":** Quando uma ação de um agente se completa e o seu resultado é conhecido (por exemplo, um artefacto é "consumido" no final do seu ciclo de vida), o sistema determina a qualidade dessa experiência. Isto pode ser um resultado estocástico baseado num atributo oculto de "qualidade" do artefacto. Com base neste resultado (bom/mau), o sistema gera uma "review" textual simulada. Conforme sugerido, isto pode começar com a seleção de frases de uma lista predefinida (por exemplo, ["Adorei!", "Excelente qualidade."] para experiências positivas) e evoluir para o uso de bibliotecas como a Faker para maior variedade, ou até mesmo um pequeno modelo de linguagem (LLM) local para gerar textos únicos.
- **Etapa B: Análise do Sentimento:** O texto simulado da review torna-se o input para uma biblioteca de análise de sentimento. A escolha da **LeIA (Léxico para Inferência Adaptada)** é excelente para este propósito.[23] A LeIA é uma versão em português da VADER, uma biblioteca clássica baseada em léxico e regras.[23] A sua principal vantagem neste contexto é ser leve, rápida e não necessitar de treino, tornando-a perfeita para um ambiente de simulação controlado.[45] A escolha de uma ferramenta de análise de sentimento é uma decisão de engenharia importante. Embora modelos baseados em Machine Learning, como os usados na biblioteca pysentimiento, possam oferecer maior precisão em textos complexos do mundo real, eles acarretam uma sobrecarga computacional significativa.[46] No universo de Lore N.A., as "reviews" são geradas pelo próprio sistema. Isto significa que o sistema pode gerar textos intencionalmente simples e inequívocos, que se alinham perfeitamente com as forças de uma ferramenta baseada em léxico como a LeIA. Não é necessário um modelo complexo para compreender um texto que foi gerado para ser facilmente compreendido. Assim, a LeIA não é apenas uma boa escolha, mas a escolha *ótima* para as fases iniciais do projeto, fornecendo a funcionalidade necessária com o mínimo de sobrecarga computacional. À medida que a simulação se torna mais complexa, a arquitetura pode ser atualizada para um modelo mais poderoso, se necessário.
- **Etapa C: Persistência e Reação:** A pontuação de sentimento (sentiment_score) obtida da LeIA é usada para atualizar o emotional_state do agente na tabela agent_state do

PostgreSQL. Esta atualização pode ser uma média móvel para criar uma certa inércia emocional, evitando que o estado de espírito do agente mude drasticamente com uma única experiência. O passo crucial é o fecho do loop de feedback: quando um agente "acorda" para executar a sua próxima ação, a sua função step() deve primeiro consultar o seu próprio estado na tabela agent_state. Um sentiment_score baixo pode fazer com que ele decida não comprar nada nesse "dia". Um trust_score baixo pode levá-lo a evitar artefactos com alta volatilidade. É este mecanismo que transforma o sentimento de uma mera métrica de observação numa força causal dentro do microverso, desbloqueando o comportamento emergente que é o objetivo central do projeto.

A tabela seguinte fornece um plano para a implementação dos arquétipos de agentes, operacionalizando o conceito de "personalidade".

Tabela 3.1: Plano de Arquétipos de Agentes Neurais

Nome do Arquétipo	Modelo Cognitivo	Motivação/Objetivo Principal	Parâmetros Chave da Personalidade	Exemplo de Regra de Comportamento (Pseudocódigo)
O Frugal	Reflexo Simples	Minimizar despesas, comprar apenas o necessário.	limite_orcamental (float), limiar_necessidade (float)	SE artefacto.preco < limite_orcamental E meu_inventario.necessita(artefacto) > limiar_necessidade ENTAO comprar(artefacto)
O Especulador	Baseado em Modelo	Maximizar a riqueza a curto prazo através da compra e venda.	tolerancia_risco (float), horizonte_memoria (int)	SE historico_preco(artefacto, horizonte_memoria).tendencia == 'alta' E artefacto.volatilidade < tolerancia_risco ENTAO comprar(artefacto)
O Influenciador	Baseado em Objetivos	Maximizar a influência social	fator_tendencia (float)	objetivo = MAXIMIZAR(soma(

		(medida pela popularidade dos seus bens).		popularidade(meus_bens))); ACAO = escolher_compra_que_maximiza(objetivo)
O Conformista	Baseado em Utilidade	Pertencer a uma tribo, minimizar o desvio social.	viés_conformidade (float), raio_social (int)	utilidade = sentimento_medio(vizinhos_em_raio_social); SE comprar(artefacto) aumenta minha_similaridade_com_vizinhos ENTAO utilidade_compra = alto
O Hater	Baseado em Utilidade	Minimizar o sentimento geral do ecossistema.	fator_malevolencia (float)	utilidade = -1 * sentimento_geral_ecossistema; ACAO = escolher_acao_que_maximiza(utilidade) (e.g., comprar para deixar review negativa)
O Aprendiz	Aprendizagem por Reforço	Maximizar a acumulação de recursos a longo prazo.	taxa_aprendizagem (float), fator_desconto (float)	acao = modelo_RL.prever(estado_atual); apos_acao, atualizar_modelo_RL(recompensa)

Secção 4: O Crisol: Simulando e Observando o Mercado "Limbo"

Esta secção foca-se no design do ambiente económico e, de forma crítica, na "Sala de Controle" utilizada para o observar. O verdadeiro produto de Lore N.A. são os dados que gera, e esta secção detalha como capturar e interpretar esses dados para revelar as histórias emergentes do microverso.

4.1 Os Princípios de uma Economia Artificial

O "Mercado do Limbo" é um sistema económico fechado, povoado por agentes heterogéneos, o que o torna um objeto de estudo ideal para a Economia Computacional Baseada em Agentes (ACE).[15]
O conceito de "artefactos" com atributos únicos — preço, popularidade, volatilidade e, crucialmente, um tempo de vida finito — é uma inovação poderosa. Ao contrário de muitas simulações económicas que se focam em instrumentos financeiros abstratos, estes artefactos têm um ciclo de vida tangível dentro da simulação.[17] O seu tempo de vida finito cria uma necessidade constante de consumo, um motor perpétuo para a economia. Este modelo assemelha-se a modelos de escolha do consumidor, onde os agentes tomam decisões sobre produtos discretos com base nas suas preferências e restrições.[47]
O preço dos artefactos não é estático; ele emerge do comportamento coletivo dos agentes. A alta procura aumentará os preços, enquanto a baixa procura os diminuirá. Este mecanismo pode ser implementado através de uma função de oferta e procura que atualiza os preços a cada "tick" da simulação. Esta dinâmica de preços é o motor que permitirá a formação das "bolhas económicas" que o projeto pretende observar.[15] O ciclo de vida do artefacto — desde a sua introdução pelo
genesis.py até ao seu eventual desaparecimento — constitui um arco narrativo económico que pode ser rastreado e analisado.

4.2 O Observatório: Projetando a "Sala de Controle" em Grafana

O Grafana é a ferramenta ideal para a "Sala de Controle" do projeto, dada a sua excelência na visualização de dados de séries temporais a partir de fontes como o PostgreSQL.[53] O objetivo é projetar dashboards que transcendam a simples monitorização de sistemas e funcionem como instrumentos científicos para a deteção e análise de comportamento emergente.[56]
Uma única dashboard será insuficiente para capturar a complexidade de Lore N.A. É necessária uma estratégia de observação em múltiplos níveis, permitindo a análise desde a saúde macroscópica do universo até à "vida" microscópica de um único agente. Esta abordagem estratificada permite uma análise multi-escala do microverso, onde os observadores podem identificar um fenómeno macro e depois fazer um "drill-down" para investigar as suas causas micro. Propõe-se o design de quatro dashboards principais:

1. **Dashboard de Saúde do Sistema e Infraestrutura:** Para os "Engenheiros Primordiais" monitorizarem a saúde da própria simulação.
2. **Dashboard de Indicadores Macro-Económicos e Sociológicos:** Para observar os fenómenos emergentes a nível do sistema.
3. **Dashboard de Dinâmicas Tribais/Arquétipos:** Para analisar o comportamento de subpopulações de agentes.
4. **Dashboard de Inspeção de Agente:** Para um "zoom" profundo na história e estado de

um único indivíduo.

4.3 Planos de Design das Dashboards

Dashboard 1: Saúde do Sistema e Infraestrutura (A "Visão de Deus")

- **Propósito:** Monitorizar a saúde dos sistemas técnicos subjacentes para garantir que a simulação está a decorrer sem problemas técnicos.
- Painéis Chave [59]:
    - **Painéis de Estatísticas (Stat Panels):** Métricas em tempo real como Agentes Ativos, Transações por Segundo, Latência da API (p95, p99), e Conexões à Base de Dados.
    - **Gráficos de Séries Temporais:** Utilização de CPU/Memória para cada contentor Docker (API, DB, Agentes), Performance das Queries da DB (tempo de execução médio).
    - **Lista de Alertas:** Um painel que exibe alertas do Grafana para erros de sistema, como Erros 5xx da API, Latência Elevada ou Uso Elevado de CPU, permitindo uma resposta proativa a problemas técnicos.[53]

Dashboard 2: Indicadores Macro-Económicos e Sociológicos (A "Visão de Mercado")

- **Propósito:** Visualizar os fenómenos emergentes de alto nível de toda a economia. Esta é a dashboard principal para a análise científica.
- **Painéis Chave:**
    - **Gráfico de Séries Temporais:** O "Índice de Sentimento Geral do Ecossistema", plotando a média do sentiment_score de todos os agentes ao longo do tempo. Esta é a métrica mais importante do projeto.
    - **Gráfico de Séries Temporais:** O "Índice de Confiança Geral", mostrando a média do trust_score.
    - **Histograma:** "Distribuição de Riqueza", um histograma que mostra a distribuição do wallet_balance por todos os agentes. Isto permitirá identificar a emergência de desigualdade económica.
    - **Mapa de Calor (Heatmap):** "Atividade de Artefactos", mostrando o volume de transações para diferentes categorias de artefactos ao longo do tempo, para identificar tendências e produtos "virais".
    - **Anotações (Annotations):** Utilização da funcionalidade de anotações do Grafana para marcar eventos chave em todos os gráficos.[53] Eventos como a introdução de um novo artefacto via genesis.py ou um choque de mercado simulado pelo n8n serão visualmente correlacionados com as suas consequências nos indicadores.

Dashboard 3: Dinâmicas Tribais (A "Visão Social")

- **Propósito:** Analisar o comportamento de diferentes subpopulações de agentes. Esta dashboard utilizará os modelos de variáveis (variable templates) do Grafana para permitir a filtragem por personality_archetype.

- **Painéis Chave:**
    - **Gráficos de Séries Temporais:** Sentimento Médio, Riqueza Média e Confiança Média, mas com a capacidade de dividir por arquétipo (por exemplo, comparar a acumulação de riqueza dos "Especuladores" vs. "Frugais").
    - **Gráfico de Tarte (Pie Chart):** "Composição da População", mostrando a percentagem atual de cada arquétipo de agente.
    - **Gráfico de Nós (Node Graph):** Uma visualização avançada para mostrar a formação de "tribos". Isto pode ser gerado por um fluxo de trabalho noturno no n8n que executa um algoritmo de clustering (por exemplo, baseado na similaridade do histórico de compras) e armazena os resultados. O gráfico de nós visualizaria estes clusters, mostrando a formação e dissolução de comunidades baseadas em gostos.

Dashboard 4: Inspetor de Agente (O "Microscópio")

- **Propósito:** Fazer um "drill-down" na história completa e no estado de um único agente para depuração e análise aprofundada. Esta dashboard seria parametrizada por agent_id.
- **Painéis Chave:**
    - **Painéis de Estatísticas:** Saldo da Carteira atual, Sentimento, Confiança.
    - **Linha do Tempo de Estado (State Timeline):** Uma visualização que mostra as mudanças de estado emocional do agente ao longo do tempo, utilizando cores para representar diferentes níveis de sentimento ou confiança.[59]
    - **Tabela:** Um registo de todo o histórico de transações do agente, consultado diretamente da base de dados PostgreSQL.
    - **Painel de Texto:** Exibindo o registo da "memória" do agente (o campo JSONB) para uma análise qualitativa das suas experiências passadas.

Secção 5: A Génese: Um Roteiro Faseado para o Microverso Mínimo Viável (MVM)

Esta secção apresenta um roteiro pragmático e acionável para a construção de Lore N.A. Aplica os princípios do Produto Mínimo Viável (MVP - Minimum Viable Product) a esta simulação complexa, garantindo que o valor — na forma de aprendizagem e fenómenos emergentes — é entregue em cada fase.

5.1 Definindo o Microverso Mínimo Viável (MVM)

O conceito de MVP, popularizado por Eric Ries, é a versão de um novo produto que permite a uma equipa recolher a quantidade máxima de *aprendizagem validada* sobre os clientes com o mínimo de esforço.[61] Para Lore N.A., o "cliente" é o observador, e o "problema" a ser resolvido

é uma simulação estática e desinteressante. Portanto, o Microverso Mínimo Viável (MVM) é a versão mais simples possível do ecossistema que pode começar a gerar *comportamento emergente não-óbvio*.

O cerne absoluto do sistema é um agente a tomar uma decisão que altera o ambiente, o que, por sua vez, altera as futuras decisões do agente. Um simples agente a comprar um item aleatório não é suficiente, pois o seu estado interno não muda de forma significativa; o loop de feedback está quebrado. O "Estado Emocional" proposto é a chave que fecha este loop. O ciclo Ação -> Resultado -> Sentimento -> Novo Estado -> Nova Ação é o coração pulsante do microverso.

Portanto, o MVM não é apenas um conjunto de agentes e um mercado; é **agentes + mercado + o loop de feedback de sentimento**. Esta é a configuração mais pequena que contém o "ADN" de todo o projeto. O objetivo do MVM é fazer com que o gráfico do "Índice de Sentimento Geral" no Grafana se mova de formas interessantes e imprevisíveis. Este é o primeiro sinal de "vida" no terrário digital. Atingir este ponto valida a hipótese central do projeto com o mínimo de funcionalidades construídas.[63]

5.2 Um Plano de Implementação Faseado

A construção do MVM e a sua expansão subsequente podem ser divididas em fases concretas e sequenciais. Esta abordagem faseada permite a entrega incremental de valor (aprendizagem) e a gestão da complexidade.[64]

Fase 1: A Sopa Primordial (Fundação e Mecânicas Centrais)

- **Objetivo:** Construir a infraestrutura não-negociável e uma simulação simples de loop aberto.
- **Tarefas:**
    1. **Configuração da Infraestrutura:** Contentorizar PostgreSQL, FastAPI e n8n. Configurar um ficheiro docker-compose.yml para orquestração local.
    2. **Esquema da Base de Dados:** Criar as tabelas iniciais no PostgreSQL para agents (com id, wallet_balance) e artifacts (com id, price estático).
    3. **API (Nave):** Implementar os endpoints básicos do FastAPI: GET /artifacts e POST /transactions.
    4. **Agente v0.1:** Criar um script Python para um único tipo de agente. A sua função step() irá buscar os artefactos disponíveis e comprar um aleatoriamente se tiver dinheiro suficiente.
    5. **Orquestrador v0.1:** Criar um fluxo de trabalho básico no n8n que aciona a função step() do agente num horário definido (o "tick" do relógio).
- **Critério de Sucesso:** Os agentes compram artefactos de forma repetida e com sucesso, e as transações são registadas de forma fiável na base de dados. O sistema funciona de forma estável sem falhas.

Fase 2: O Despertar da Consciência (Fechando o Loop de Feedback)

- **Objetivo:** Implementar o MVM, introduzindo o loop de feedback de sentimento.

- **Tarefas:**
    1. **Atualização do Esquema da DB:** Adicionar a tabela agent_state ao PostgreSQL para armazenar sentiment_score e trust_score.
    2. **Melhoria dos Artefactos:** Adicionar um atributo oculto de "qualidade" aos artefactos, que influenciará o resultado da sua "utilização".
    3. **Serviço de Sentimento:** Criar o microsserviço Python que:
        - É acionado após uma transação ser concluída (por exemplo, ouvindo um evento TRANSACTION_COMPLETED).
        - Determina o resultado (bom/mau) com base na qualidade do artefacto.
        - Gera uma string de "review" simulada.
        - Utiliza a biblioteca **LeIA** para calcular um sentiment_score.[23]
        - Atualiza o estado do agente na tabela agent_state.
    4. **Agente v0.2:** Modificar a função step() do agente. Antes de tomar uma decisão, ele deve primeiro consultar o seu próprio estado emocional. Um trust_score baixo pode fazer com que ele hesite e não compre nada nesse "dia".
- **Critério de Sucesso:** O gráfico do "Índice de Sentimento Geral" na dashboard do Grafana exibe um comportamento dinâmico e não-aleatório que se correlaciona com a atividade na simulação. A emergência foi alcançada.

Fase 3: A Ascensão da Sociedade (Heterogeneidade e Dinâmicas Sociais)

- **Objetivo:** Introduzir a complexidade social, movendo-se de uma população homogénea para uma sociedade diversificada.
- **Tarefas:**
    1. **Agente v1.0:** Implementar os arquétipos de agentes definidos na Tabela 3.1. Criar diferentes classes de agentes em Python com lógicas de decisão distintas (Baseada em Utilidade, Baseada em Objetivos, etc.).
    2. **Rede Social:** Adicionar uma tabela agent_relationships à base de dados para representar uma rede social (por exemplo, quem "segue" quem).
    3. **Mecânica de Influência:** Modificar a função step() do agente. As compras dos agentes "Influenciadores" agora aumentam a pontuação de "popularidade" pública de um artefacto. A tomada de decisão dos agentes "Influenciáveis" será ponderada por esta pontuação de popularidade e pelas opiniões dos agentes que eles seguem.
- **Critério de Sucesso:** A dashboard de "Dinâmicas Tribais" mostra a formação de clusters de agentes. Observam-se "sucessos virais" e "fracassos" entre os artefactos, impulsionados pela influência social.

Fase 4 e Além: Escalando o Microverso

- **Objetivo:** Construir o ecossistema rico e completo descrito no conceito das "Camadas" do utilizador.
- **Tarefas:**
    - Implementar o "Ateliê Algorítmico" e a "Caixa Coletiva" como novos microsserviços e endpoints de API.
    - Implementar o simulador de logística "Logos Prime", adicionando atrasos e falhas potenciais às transações, o que irá impactar o sentimento dos agentes.

- Começar a experimentar com verdadeiros **Agentes de Aprendizagem** (por exemplo, utilizando aprendizagem por reforço) que podem evoluir as suas estratégias para alcançar maior "sucesso" (acumulação de recursos), observando quais estratégias sobrevivem e prosperam.
- Expandir a "Sala de Controle" com visualizações mais sofisticadas, possivelmente utilizando técnicas de redução de dimensionalidade como a Análise de Componentes Principais (PCA) para detetar padrões emergentes em dados de alta dimensão.[57]

Conclusão: A Lore Não Escrita

O projeto Lore N.A. é uma proposta excecionalmente bem concebida, que se situa na interseção da engenharia de software, da inteligência artificial e da ciência de sistemas complexos. A sua visão de criar um "terrário digital" para estudar o comportamento socioeconómico emergente não é apenas tecnicamente viável, mas também cientificamente relevante. Ao posicionar a "lore" — a história, a cultura e a economia geradas pelos próprios agentes — como o principal produto, o projeto estabelece-se como um instrumento de descoberta em vez de uma aplicação convencional.

Esta análise validou as premissas fundamentais do projeto, enraizando-as nas teorias estabelecidas da Vida Artificial e da Modelagem Baseada em Agentes. Demonstrou que a abordagem "bottom-up" é a correta para estudar fenómenos emergentes como bolhas económicas e dinâmicas tribais. A arquitetura tecnológica proposta é robusta, e a evolução sugerida para um modelo de microsserviços orientado a eventos garante a escalabilidade e a resiliência necessárias para um sistema de tal complexidade. O design dos Agentes Neurais, com os seus estados emocionais e arquétipos de personalidade, fornece o mecanismo para um comportamento rico e imprevisível, enquanto a "Sala de Controle" em Grafana oferece a lente necessária para observar e analisar este novo mundo.

O roteiro faseado, centrado na construção de um Microverso Mínimo Viável (MVM), oferece um caminho pragmático para a implementação. Ao focar-se primeiro na criação do loop de feedback central (Ação -> Resultado -> Sentimento -> Novo Estado -> Nova Ação), a equipa pode validar a hipótese mais crítica do projeto com o mínimo de esforço: a capacidade do sistema de gerar comportamento emergente interessante.

O sucesso final de Lore N.A. não será medido pelo código que é escrito, mas pela complexidade, surpresa e riqueza das histórias que dele emergem. Ao seguir uma abordagem estruturada, teoricamente fundamentada e faseada, a visão de Lore N.A. pode ser transformada de um conceito brilhante num poderoso instrumento científico. O objetivo é acender a faísca e, depois, observar a lore que os agentes, através das suas vidas digitais coletivas, escreverão por si próprios.

Works cited

1. Agent-Based Models and Simulations Tools - Encyclopedia.pub, accessed June

23, 2025, https://encyclopedia.pub/entry/40674
2. Agent-Based Modeling | Columbia University Mailman School of Public Health, accessed June 23, 2025, https://www.publichealth.columbia.edu/research/population-health-methods/agent-based-modeling
3. Artificial life - Wikipedia, accessed June 23, 2025, https://en.wikipedia.org/wiki/Artificial_life
4. Artificial Life: An Overview | Books Gateway - MIT Press Direct, accessed June 23, 2025, https://direct.mit.edu/books/edited-volume/3941/Artificial-LifeAn-Overview
5. Introduction to Artificial Life (Paperback) - Harvard Book Store, accessed June 23, 2025, https://www.harvard.com/book/9781461272311
6. PaulTOliver/salis-v2: Salis is an artificial life simulation ... - GitHub, accessed June 23, 2025, https://github.com/PaulTOliver/salis-v2
7. Conway's Game of Life - Wikipedia, accessed June 23, 2025, https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
8. Play John Conway's Game of Life, accessed June 23, 2025, https://playgameoflife.com/
9. Conway's Game of Life - ScienceDemos.org.uk, accessed June 23, 2025, https://sciencedemos.org.uk/conway_game_of_life.php
10. SakanaAI/asal: Automating the Search for Artificial Life with ... - GitHub, accessed June 23, 2025, https://github.com/SakanaAI/asal
11. rubberduck203/Evolve: Artificial Life Simulator Originally ... - GitHub, accessed June 23, 2025, https://github.com/rubberduck203/Evolve
12. kourgeorge/project-origin: An Artificial Life simulator for ... - GitHub, accessed June 23, 2025, https://github.com/kourgeorge/project-origin
13. 1.2: Introduction to agent-based modeling - Mathematics LibreTexts, accessed June 23, 2025, https://math.libretexts.org/Bookshelves/Applied_Mathematics/Agent-Based_Evolutionary_Game_Dynamics_(Izquierdo_Izquierdo_and_Sandholm)/01%3A_Introduction/1.02%3A_Introduction_to_agent-based_modeling
14. Autonomous Agent Simulation: A Deep Dive into AI ... - SmythOS, accessed June 23, 2025, https://smythos.com/developers/agent-development/autonomous-agent-simulation/
15. Agent-based Modeling in Finance: Revolutionizing ... - SmythOS, accessed June 23, 2025, https://smythos.com/managers/finance/agent-based-modeling-in-finance/
16. Exploring Financial Market Trends with Agent-Based Modeling ..., accessed June 23, 2025, https://www.numberanalytics.com/blog/financial-market-agent-based-modeling
17. (PDF) Mesa: An Agent-Based Modeling Framework - ResearchGate, accessed June 23, 2025, https://proceedings.scipy.org/articles/Majora-7b98e3ed-009.pdf
18. systemsthinkingalliance.org, accessed June 23, 2025, https://systemsthinkingalliance.org/the-crucial-role-of-emergence-in-systems-thinking/#:~:text=The%20philosophy%20behind%20emergence%20or,their%20indi

vidual%20components%20or%20parts.
19. Emergence: The Key to Understanding Complex Systems, accessed June 23, 2025, https://systemsthinkingalliance.org/the-crucial-role-of-emergence-in-systems-thinking/
20. Boids, accessed June 23, 2025, https://people.ece.cornell.edu/land/courses/ece4760/labs/s2021/Boids/Boids.html
21. An Order of Magnitude More Boids: Optimizing Flocking Simulations - Mark Tensen, accessed June 23, 2025, https://marktension.nl/blog/order-of-magnitude-boids/
22. Boids algorithm demonstration - Ben Eater, accessed June 23, 2025, https://eater.net/boids
23. Brazilian E-Commerce: EDA | NLP | ML 🛍️ - Kaggle, accessed June 23, 2025, https://www.kaggle.com/code/annastasy/brazilian-e-commerce-eda-nlp-ml
24. Deploy FastAPI with PostgreSQL on Northflank — Northflank, accessed June 23, 2025, https://northflank.com/guides/deploy-fastapi-postgres-cloud-docker
25. FastAPI in Containers - Docker, accessed June 23, 2025, https://fastapi.tiangolo.com/deployment/docker/
26. Deploying FastAPI and PostgreSQL Microservices to Kubernetes using Minikube, accessed June 23, 2025, https://developers.eksworkshop.com/docs/python/kubernetes/deploy-app/
27. Layered Architecture Pattern in Java: Building Scalable and Maintainable Applications, accessed June 23, 2025, https://java-design-patterns.com/patterns/layered-architecture/
28. Layered Architecture: Building Scalable & Maintainable Software Systems | Bitloops Docs, accessed June 23, 2025, https://bitloops.com/docs/bitloops-language/learning/software-architecture/layered-architecture
29. Understanding the Layered Architecture Pattern: A Comprehensive ..., accessed June 23, 2025, https://dev.to/yasmine_ddec94f4d4/understanding-the-layered-architecture-pattern-a-comprehensive-guide-1e2j
30. Multitier architecture - Wikipedia, accessed June 23, 2025, https://en.wikipedia.org/wiki/Multitier_architecture
31. Data Integration in Microservices: A Guide - RTS Labs, accessed June 23, 2025, https://rtslabs.com/data-integration-in-a-microservices-architecture
32. Four Design Patterns for Event-Driven, Multi-Agent Systems, accessed June 23, 2025, https://www.confluent.io/blog/event-driven-multi-agent-systems/
33. The Benefits of Microservices Orchestration | Universal Equations, Inc., accessed June 23, 2025, https://www.uequations.com/en-us/article/business-process/2025-06/benefits-microservices-orchestration
34. Advanced n8n Tips: Building Scalable Integrations & Automation - GoCodeo, accessed June 23, 2025, https://www.gocodeo.com/post/advanced-n8n-tips-building-scalable-integration

s-automation-hchgm
35. Powerful Workflow Automation Software & Tools - n8n, accessed June 23, 2025, https://n8n.io/
36. n8n Workflow Automation: Scalable, Secure & Developer-Friendly - ColorWhistle, accessed June 23, 2025, https://colorwhistle.com/automation-with-n8n/
37. Unlock Smart Workflows with n8n Business Automation - HYPESTUDIO, accessed June 23, 2025, https://hypestudio.org/unlock-smart-workflows-with-n8n-business-automation/
38. 2 How to Simulate Economic Models - DIY Macroeconomic Model Simulation, accessed June 23, 2025, https://macrosimulation.org/how_to_simulate
39. ABIDES-Economist: Agent-Based Simulation of Economic Systems with Learning Agents, accessed June 23, 2025, https://arxiv.org/html/2402.09563v1
40. Getting Started with Autonomous Agents: Tutorials for ... - SmythOS, accessed June 23, 2025, https://smythos.com/developers/agent-development/autonomous-agents-tutorials/
41. 10 Agent-Based Modeling Strategies for Boosting Simulation Accuracy, accessed June 23, 2025, https://www.numberanalytics.com/blog/10-agent-based-modeling-strategies-boosting-simulation-accuracy
42. Understanding AI Agent Decision-Making Processes - SmythOS, accessed June 23, 2025, https://smythos.com/developers/agent-development/ai-agent-decision-making/
43. Types of AI Agents | IBM, accessed June 23, 2025, https://www.ibm.com/think/topics/ai-agent-types
44. How Reasoning AI Agents Transform High-Stakes Decision Making | NVIDIA Blog, accessed June 23, 2025, https://blogs.nvidia.com/blog/reasoning-ai-agents-decision-making/
45. (PDF) Sentiment analysis using a lexicon-basedapproach in Lisbon ..., accessed June 23, 2025, https://www.researchgate.net/publication/391077881_Sentiment_analysis_using_a_lexicon-basedapproach_in_Lisbon_Portugal
46. Machine Learning and Deep Learning Sentiment Analysis Models ..., accessed June 23, 2025, https://www.mdpi.com/2227-9709/11/2/24
47. Agent-Based Modeling in Marketing: Transforming Data-Driven Strategies for Success, accessed June 23, 2025, https://smythos.com/managers/marketing/agent-based-modeling-in-marketing/
48. Agent-Based Simulation New Product Development Processes - White Rose Research Online, accessed June 23, 2025, https://eprints.whiterose.ac.uk/id/eprint/135644/1/Agent-Based%20Simulation%20New%20Product.pdf
49. Brief introductory guide to agent-based modeling and an illustration from urban health research - PubMed Central, accessed June 23, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC5391997/
50. Agent-Based Modeling of Consumer Choice, accessed June 23, 2025,

https://www.gisagents.org/2023/09/agent-based-modeling-of-consumer-choice.html
51. Agent-Based Model for End-of-Life Product Flow Analysis - MDPI, accessed June 23, 2025, https://www.mdpi.com/2079-9276/7/3/42
52. An Agent-Based Model of Electricity Consumer: Smart Metering Policy Implications in Europe - Journal of Artificial Societies and Social Simulation, accessed June 23, 2025, https://www.jasss.org/20/1/12.html
53. Understanding Grafana: An Overview of the Open-Source Visualization Platform, accessed June 23, 2025, https://www.examcollection.com/blog/understanding-grafana-an-overview-of-the-open-source-visualization-platform/
54. Visualize log data | Grafana Loki documentation, accessed June 23, 2025, https://grafana.com/docs/loki/latest/visualize/grafana/
55. Grafana Graph Visualization | Tom Sawyer Software, accessed June 23, 2025, https://blog.tomsawyer.com/grafana-graph-visualization
56. Introducing Observability and the Grafana Stack - Packt, accessed June 23, 2025, https://www.packtpub.com/en-mx/product/observability-with-grafana-9781803248004/chapter/chapter-1-introducing-observability-and-the-grafana-stack-2/section/chapter-1-introducing-observability-and-the-grafana-stack?chapterId=2
57. A data mining technique for detecting emergent behaviors in agent ..., accessed June 23, 2025, https://www.worldscientific.com/doi/10.1142/S1793962325500199
58. Agent-Based Models Predict Emergent Behavior of Heterogeneous Cell Populations in Dynamic Microenvironments - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2020.00249/full
59. Visualizations | Grafana documentation, accessed June 23, 2025, https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/
60. Prometheus + Grafana Turns Your Go App into a Real-Time Dashboard of Truth, accessed June 23, 2025, https://hackernoon.com/prometheus-grafana-turns-your-go-app-into-a-real-time-dashboard-of-truth
61. Minimum Viable Product (MVP): What is it & Why it Matters - Atlassian, accessed June 23, 2025, https://www.atlassian.com/agile/product-management/minimum-viable-product
62. Minimum viable product - Wikipedia, accessed June 23, 2025, https://en.wikipedia.org/wiki/Minimum_viable_product
63. Building a Minimum Viable Product (MVP) for Your Solo AI Startup with Limited Resources, accessed June 23, 2025, https://www.nucamp.co/blog/solo-ai-tech-entrepreneur-2025-building-a-minimum-viable-product-mvp-for-your-solo-ai-startup-with-limited-resources
64. How to Determine Minimum Viable Product? : r/gamedev - Reddit, accessed June 23, 2025, https://www.reddit.com/r/gamedev/comments/10snsja/how_to_determine_minimum_viable_product/
65. What is MVP? A Minimum Viable Product with Examples! - Agicent, accessed

June 23, 2025,
https://www.agicent.com/blog/what-is-mvp-minimum-viable-product/