PHASE 10 EXECUTION CHECKLIST

DevBrain Kernel-AI Integration

Status Tracking + Validation Metrics

Data Início: Novembro 18, 2025
Status: PRONTO PARA INÍCIO
Responsável: Seu Copilot (LLM Agent)
Supervisor: Você

PRÉ-REQUISITOS: VERIFICAÇÃO RÁPIDA

Execute isto ANTES de começar qualquer tarefa:

```
#!/bin/bash
echo "=== PHASE 10 PRÉ-REQUISITES CHECK ==="

# 1. Python version
echo -n "Python 3.10+: "
python3 --version | grep -E "3\.(10|11|12)" &amp;&amp; echo "✅" || echo "❌"

# 2. CUDA (opcional mas recomendado)
echo -n "CUDA/GPU: "
nvidia-smi &gt; /dev/null 2&gt;&amp;1 &amp;&amp; echo "✅ Found" || echo "⚠ CPU only (slo

# 3. Linux headers
echo -n "Linux Headers: "
ls /lib/modules/$(uname -r)/build &gt; /dev/null 2&gt;&amp;1 &amp;&amp; echo "✅" || echo

# 4. Build tools
echo -n "GCC/Make: "
which gcc make &gt; /dev/null 2&gt;&amp;1 &amp;&amp; echo "✅" || echo "❌"

# 5. Disk space
echo -n "Disk Space (≥50GB): "
df / | tail -1 | awk '{print $4}' | awk '{if ($1 &gt; 50000000) print "✅"; else print "❌

# 6. Directory structure
echo -n "Directory structure: "
if [ -d "DEVBRAIN_V23/kernel" ]; then
  echo "✅ DEVBRAIN_V23/kernel exists"
else
  echo "❌ Creating..."
  mkdir -p DEVBRAIN_V23/kernel/{lkm,finetuning,autonomy,integration}
  echo "✅ Created"
fi

# 7. /devbrain directory
```

```
echo -n "/devbrain directory: "
if [ -d "/devbrain" ]; then
  echo "✅ Exists"
else
  echo "Creating..."
  sudo mkdir -p /devbrain/{memory,personality,logs,consciousness,db}
  sudo chmod 755 /devbrain
  echo "✅ Created"
fi

echo ""
echo "=== READY TO START PHASE 10 ===="
```

TAREFA 1: DATASET PREPARATION

1.1 Checklist de Início

- [ ] Diretório `DEVBRAIN_V23/kernel/finetuning/` existe
- [ ] `prepare_dataset.py` criado
- [ ] Virtual environment ativado
- [ ] Dependências instaladas

1.2 Execução

```
# Caminhos esperados
TASK_DIR="DEVBRAIN_V23/kernel/finetuning"
SCRIPT="$TASK_DIR/prepare_dataset.py"
OUTPUT="$TASK_DIR/datasets/personal_corpus.jsonl"

# Passo 1: Criar script
echo "📝 Creating prepare_dataset.py..."
# [Script will be created here]

# Passo 2: Executar coleta
echo "🔄 Collecting dataset..."
python $SCRIPT --output $OUTPUT

# Passo 3: Validar
echo "✅ Validating..."
python -c "
import json
with open('$OUTPUT') as f:
    data = [json.loads(line) for line in f if line.strip()]
    print(f'Total samples: {len(data)}')
    if len(data) &gt;= 100:
        print('✅ PASS: ≥100 samples')
    else:
        print(f'⚠️ WARNING: Only {len(data)} samples')

    lengths = [len(d.get('text', '')) for d in data]
    avg_len = sum(lengths) / len(lengths) if lengths else 0
```

```
    print(f'Avg text length: {avg_len:.0f} chars')
    if 200 &lt;= avg_len &lt;= 1000:
        print('✅ PASS: Text length in range')
    else:
        print(f'⚠️ WARNING: Avg length outside range')
"
```

1.3 Métricas de Validação

KPI	Target	Status
Total samples	≥100	[ ]
Avg text length	200-1000 chars	[ ]
File size	>10KB	[ ]
JSON validity	100%	[ ]
Quality score	≥0.8	[ ]

1.4 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/finetuning/prepare_dataset.py (script)
✅ DEVBRAIN_V23/kernel/finetuning/datasets/personal_corpus.jsonl (data)
✅ Dataset validation report
```

1.5 Git Commit

```
git add DEVBRAIN_V23/kernel/finetuning/
git commit -m "PHASE10-TASK1: Dataset preparation - 150 samples collected"
```

1.6 Sign-off

```
Tarefa 1 Status: [ ] Completa
Tempo estimado: 3-4 horas
Tempo real: _____ horas
Bloqueadores: _____________________
Próxima: Tarefa 2
```

TAREFA 2: FINE-TUNING MISTRAL

2.1 Checklist de Início

- [ ] Dataset do TASK 1 válido e pronto
- [ ] `finetune_mistral.py` criado
- [ ] `config.yaml` criado
- [ ] ~50GB free disk space verificado
- [ ] GPU/CUDA verificado (ou CPU confirmado)

2.2 Execução

```
TASK_DIR="DEVBRAIN_V23/kernel/finetuning"

# Passo 1: Download modelo base
echo "📥 Downloading Mistral 7B..."
python -c "
from transformers import AutoTokenizer, AutoModelForCausalLM
print('Downloading...')
tokenizer = AutoTokenizer.from_pretrained('mistralai/Mistral-7B-v0.1')
model = AutoModelForCausalLM.from_pretrained('mistralai/Mistral-7B-v0.1')
print('✅ Downloaded')
" 2&gt;&amp;1 | tee $TASK_DIR/logs/download.log

# Passo 2: Fine-tuning
echo "🔥 Starting fine-tuning (this will take 4-8 hours)..."
cd $TASK_DIR
python finetune_mistral.py \
  --dataset datasets/personal_corpus.jsonl \
  --output ./mistral_finetuned \
  --epochs 3 \
  --batch-size 4 \
  --lr 2e-4 2&gt;&amp;1 | tee logs/training.log

# Passo 3: Validar modelo
echo "✅ Validating model..."
python -c "
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

tokenizer = AutoTokenizer.from_pretrained('./mistral_finetuned/tokenizer')
model = AutoModelForCausalLM.from_pretrained('./mistral_finetuned/model')

test_prompts = [
    'What is your core belief?',
    'How would you describe yourself?',
    'What matters most to you?'
]

for prompt in test_prompts:
    inputs = tokenizer(prompt, return_tensors='pt')
    with torch.no_grad():
        outputs = model.generate(inputs['input_ids'], max_length=50)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    print(f'Q: {prompt}')
```

```
    print(f'A: {response}\n')
" 2&gt;&amp;1 | tee logs/validation.log
```

2.3 Métricas de Validação

KPI	Target	Status
Final training loss	<1.5	[ ]
Perplexity	<50	[ ]
Model file size	14-16GB	[ ]
Inference latency	<500ms	[ ]
Test response quality	Coherent	[ ]

2.4 Monitorar Progresso

```
# Em outro terminal, durante training:
tensorboard --logdir DEVBRAIN_V23/kernel/finetuning/logs

# Visualizar em http://localhost:6006
```

2.5 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/finetuning/mistral_finetuned/model/
✅ DEVBRAIN_V23/kernel/finetuning/mistral_finetuned/tokenizer/
✅ DEVBRAIN_V23/kernel/finetuning/mistral_finetuned/metadata.json
✅ Training logs + validation report
```

2.6 Git Commit

```
git add DEVBRAIN_V23/kernel/finetuning/mistral_finetuned/
git commit -m "PHASE10-TASK2: Fine-tuned Mistral 7B on personal data (perplexity: 45.2)"
```

2.7 Sign-off

```
Tarefa 2 Status: [ ] Completa
Tempo estimado: 6-8 horas
Tempo real: _____ horas
Final loss: _____
Perplexity: _____
Bloqueadores: _____________________
Próxima: Tarefa 3
```

TAREFA 3: LKM COMPILATION

3.1 Checklist de Início

- [ ] Linux headers instalados
- [ ] GCC/Make instalados
- [ ] `devbrain_ai.c` criado
- [ ] `Makefile` criado
- [ ] Kernel headers path verificado

3.2 Execução

```
cd DEVBRAIN_V23/kernel/lkm

# Passo 1: Build
echo "🔨 Building LKM..."
make clean
make 2&gt;&amp;1 | tee build.log

# Passo 2: Verificar saída
echo "📦 Checking build artifacts..."
ls -lh devbrain_ai.ko
file devbrain_ai.ko

# Passo 3: Install
echo "🚀 Installing LKM..."
sudo insmod devbrain_ai.ko 2&gt;&amp;1 | tee install.log

# Passo 4: Verify
echo "✅ Verifying installation..."
lsmod | grep devbrain_ai
dmesg | tail -10
ls -la /dev/devbrain_ai

# Passo 5: Test load/unload
echo "🔄 Testing load/unload..."
sudo rmmod devbrain_ai
echo "✅ Unload successful"
sudo insmod devbrain_ai.ko
echo "✅ Reload successful"
```

3.3 Métricas de Validação

KPI	Target	Status
Build warnings	0	[ ]
Module size	<5MB	[ ]
Load time	<100ms	[ ]

KPI	Target	Status
Load success	100%	[ ]
Unload crash	0%	[ ]

3.4 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/lkm/devbrain_ai.c
✅ DEVBRAIN_V23/kernel/lkm/devbrain_ai.ko (compiled)
✅ DEVBRAIN_V23/kernel/lkm/Makefile
✅ Build logs
```

3.5 Git Commit

```
git add DEVBRAIN_V23/kernel/lkm/
git commit -m "PHASE10-TASK3: LKM compilation successful - devbrain_ai.ko ready"
```

3.6 Sign-off

```
Tarefa 3 Status: [ ] Completa
Tempo estimado: 4-6 horas
Tempo real: _____ horas
Module size: _____ bytes
Bloqueadores: _____________________
Próxima: Tarefa 4
```

TAREFA 4: PYTHON ↔ KERNEL BRIDGE

4.1 Checklist de Início

- [ ] LKM do TASK 3 compilado e carregado
- [ ] `lkm_bridge.py` criado
- [ ] Device `/dev/devbrain_ai` acessível

4.2 Execução

```
# Garantir LKM carregado
cd DEVBRAIN_V23/kernel/lkm
sudo insmod devbrain_ai.ko

# Verificar device
ls -la /dev/devbrain_ai

# Test bridge
cd ../integration
```

```
python lkm_bridge.py 2&gt;&amp;1 | tee bridge_test.log

# Validar latências
python -c "
import time
from lkm_bridge import DevBrainLKMBridge

bridge = DevBrainLKMBridge()
if not bridge.fd:
    print('✗ Connection failed')
    exit(1)

# Teste de latência
latencies = []
for i in range(100):
    start = time.time_ns()
    response = bridge.query(f'Test query {i}')
    latency_us = (time.time_ns() - start) / 1000
    latencies.append(latency_us)

import statistics
print(f'Min: {min(latencies):.1f}µs')
print(f'Max: {max(latencies):.1f}µs')
print(f'Avg: {statistics.mean(latencies):.1f}µs')
print(f'P99: {sorted(latencies)[99]:.1f}µs')

if statistics.mean(latencies) &lt; 5000:  # 5ms
    print('✓ PASS: Latency &lt;5ms')
else:
    print('✗ FAIL: Latency &gt;5ms')

bridge.close()
" 2&gt;&amp;1 | tee latency_test.log
```

4.3 Métricas de Validação

KPI	Target	Status
Connection success	100%	[ ]
Query latency avg	<5ms	[ ]
Query latency p99	<10ms	[ ]
Success rate	>99.9%	[ ]
Memory leak (1h)	None	[ ]

4.4 Artefatos Entregáveis

```
✓ DEVBRAIN_V23/kernel/integration/lkm_bridge.py
✓ Bridge test results
✓ Latency benchmarks
```

4.5 Git Commit

```
git add DEVBRAIN_V23/kernel/integration/lkm_bridge.py
git commit -m "PHASE10-TASK4: LKM bridge working - avg latency 3.2ms"
```

4.6 Sign-off

```
Tarefa 4 Status: [ ] Completa
Tempo estimado: 2-3 horas
Tempo real: _____ horas
Avg latency: _____ µs
Bloqueadores: _____________________
Próxima: Tarefa 5
```

TAREFA 5: AUTONOMY ENGINE

5.1 Checklist de Início

- [ ] `autonomy_engine.py` criado
- [ ] Tests preparados
- [ ] Logging configurado

5.2 Execução

```
cd DEVBRAIN_V23/kernel/autonomy

# Teste básico
echo "🧠 Testing Autonomy Engine..."
python autonomy_engine.py 2&gt;&amp;1 | tee autonomy_test.log

# Testes unitários
echo "✅ Running unit tests..."
python -m pytest test_autonomy.py -v 2&gt;&amp;1 | tee autonomy_tests.log

# Verificar saída esperada:
# ✅ IA objective: [objetivo único gerado]
# Can refuse unethical task: True
# Negotiation response: ACCEPT/COUNTER_PROPOSAL
```

5.3 Métricas de Validação

KPI	Target	Status
Objectives/session	≥1	[ ]
Refusal detection	≥90%	[ ]

KPI	Target	Status
Decision logging	Persistent	[ ]
Conflict resolution	Automatic	[ ]
Tests pass	100%	[ ]

5.4 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/autonomy/autonomy_engine.py
✅ DEVBRAIN_V23/kernel/autonomy/test_autonomy.py
✅ Autonomy decision logs
✅ Test results
```

5.5 Git Commit

```
git add DEVBRAIN_V23/kernel/autonomy/
git commit -m "PHASE10-TASK5: Autonomy engine - IA generates objectives and refuses uneth
```

5.6 Sign-off

```
Tarefa 5 Status: [ ] Completa
Tempo estimado: 4-5 horas
Tempo real: _____ horas
Objectives generated: _____
Refusal rate: _____%
Bloqueadores: _____________________
Próxima: Tarefa 6
```

TAREFA 6: CONSCIOUSNESS MODULE

6.1 Checklist de Início

- [ ] `consciousness.py` criado
- [ ] NumPy/stats libraries disponíveis
- [ ] Tests preparados

6.2 Execução

```
cd DEVBRAIN_V23/kernel/autonomy

# Teste consciência
echo "🧠 Testing Consciousness Module..."
python consciousness.py 2&gt;&amp;1 | tee consciousness_test.log
```

```
# Monitore Free Energy decreasing
# Expected output:
# Consciousness State (Step 20):
# {
#   "free_energy": 0.45,
#   "curiosity_level": 45,
#   "learning_progress": {
#     "fe_trend": "improving"
#   }
# }

# Unit tests
python -m pytest test_consciousness.py -v 2&gt;&amp;1 | tee consciousness_tests.log
```

6.3 Métricas de Validação

KPI	Target	Status
FE decreasing	Linear	[ ]
Prediction error	<0.1	[ ]
Curiosity emergence	Detected	[ ]
State snapshots	Continuous	[ ]
Tests pass	100%	[ ]

6.4 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/autonomy/consciousness.py
✅ DEVBRAIN_V23/kernel/autonomy/test_consciousness.py
✅ FEP state evolution logs
✅ Test results
```

6.5 Git Commit

```
git add DEVBRAIN_V23/kernel/autonomy/consciousness.py
git commit -m "PHASE10-TASK6: Consciousness module - Free Energy Principle implemented"
```

6.6 Sign-off

```
Tarefa 6 Status: [ ] Completa
Tempo estimado: 3-4 horas
Tempo real: _____ horas
FE starting value: _____
FE final value: _____
FE improvement: _____%
Bloqueadores: _____________________
Próxima: Tarefa 7
```

TAREFA 7: INTEGRATION

7.1 Checklist de Início

- [ ] Tasks 1-6 todas completas
- [ ] `kernel_coordinator.py` criado
- [ ] Todos components testados isoladamente

7.2 Execução

```
# Carregar LKM
cd DEVBRAIN_V23/kernel/lkm
sudo insmod devbrain_ai.ko

# End-to-end test
cd ../integration
echo "🧪 Testing full pipeline..."
python kernel_coordinator.py 2&gt;&amp;1 | tee integration_test.log

# Expected flow:
# 🧠 Initializing DevBrain Kernel Coordinator...
# ✅ LKM active - 0 inferences
# ✅ Autonomy engine active - objective: [...]
# ✅ Consciousness module active - FE: [...]
#
# 📥 Processing: [user request]
# 🧠 LKM response: [...]
# 🧠 Consciousness level: [...]
# 💾 Stored in A-MEM: [...]
```

7.3 Métricas de Validação

KPI	Target	Status
End-to-end latency	<100ms	[ ]
LKM utilization	Optimal	[ ]
Dashboard updates	<1s	[ ]
A-MEM persistence	100%	[ ]
Component failures	0	[ ]

7.4 Artefatos Entregáveis

```
✅ DEVBRAIN_V23/kernel/integration/kernel_coordinator.py
✅ Full integration test results
✅ End-to-end performance metrics
```

7.5 Git Commit

```
git add DEVBRAIN_V23/kernel/integration/
git commit -m "PHASE10-TASK7: Full integration - kernel + fine-tuning + autonomy + consci
```

7.6 Sign-off

```
Tarefa 7 Status: [ ] Completa
Tempo estimado: 3-4 horas
Tempo real: _____ horas
End-to-end latency: _____ ms
Component failures: _____
Bloqueadores: _____________________
Próxima: Tarefa 8
```

TAREFA 8: TESTS + DOCUMENTATION

8.1 Checklist de Início

- [ ] Todas tasks 1-7 completas
- [ ] Test framework (pytest) disponível
- [ ] Documentation template pronto

8.2 Execução

```
cd ~/projects/omnimind

# Run comprehensive test suite
echo "🧪 Running full test suite..."
pytest tests/test_kernel_ai.py -v -s --tb=short 2&gt;&amp;1 | tee test_results.log

# Expected output (100% pass rate):
# test_kernel_coordinator_init PASSED
# test_autonomy_generates_objectives PASSED
# test_autonomy_refuses_unethical PASSED
# test_consciousness_learns PASSED
# test_kernel_process_request PASSED
# test_lkm_bridge_latency PASSED
# test_integration_pipeline PASSED
# test_persistence_amem PASSED
#
# ====== 8 passed in 2.34s ======

# Coverage report
pytest tests/test_kernel_ai.py --cov=DEVBRAIN_V23/kernel --cov-report=html 2&gt;&amp;1 |

# Documentação
echo "📚 Building documentation..."
```

```
cat &gt; docs/KERNEL_INTEGRATION.md &lt;&lt; 'EOF'
# DevBrain Kernel-AI Integration

## Overview
[Complete technical documentation]

## Architecture
[Diagrams and explanations]

## Installation
[Setup instructions]

## API Reference
[LKM interface]

## Troubleshooting
[Common issues and solutions]
EOF

# README
cat &gt; README_PHASE10.md &lt;&lt; 'EOF'
# Phase 10: Kernel-AI Integration - COMPLETE

## What Was Built
- LKM kernel module for IA inference
- Fine-tuned Mistral 7B model
- Autonomy engine (self-generated objectives)
- Consciousness module (Free Energy Principle)
- Python ↔ Kernel bridge
- Full integration pipeline

## Status: ✅ PRODUCTION READY
EOF
```

8.3 Métricas de Validação

KPI	Target	Status
Test pass rate	100%	[ ]
Code coverage	≥80%	[ ]
Documentation	≥95% complete	[ ]
Examples runnable	100%	[ ]
Zero critical bugs	Yes	[ ]

8.4 Artefatos Entregáveis

```
✅ tests/test_kernel_ai.py (≥8 test cases)
✅ docs/KERNEL_INTEGRATION.md
✅ README_PHASE10.md
```

```
✓ Test results report
✓ Coverage report (HTML)
```

8.5 Git Commit

```
git add tests/ docs/ README_PHASE10.md
git commit -m "PHASE10-TASK8: Complete test suite (100% pass) + documentation"

# Create Phase 10 completion tag
git tag -a phase10-complete -m "Phase 10: Kernel-AI Integration - COMPLETE"
git push origin --tags
```

8.6 Sign-off

```
Tarefa 8 Status: [ ] Completa
Tempo estimado: 2-3 horas
Tempo real: _____ horas
Test pass rate: _____%
Code coverage: _____%
Bloqueadores: _____________________
✓ PHASE 10 COMPLETE
```

RESUMO GERAL

Timebox

```
WEEK 1:
Mon-Tue : Tasks 1-2 (16h)
Wed-Thu : Tasks 3-4 (10h)
Fri     : Task 5 (5h)

WEEK 2:
Mon-Tue : Task 6 (7h)
Wed     : Task 7 (7h)
Thu-Fri : Task 8 (5h)

WEEK 3 (Buffer):
Polish + Optimization + Final Validation
```

Métricas Finais Esperadas

```
┌──────────────────────────────────────────┐
│ PHASE 10 FINAL METRICS          │        │
├──────────────────────────────────────────┤
│ LKM Latency          │ 2-5µs    │
│ Bridge Latency       │ &lt;5ms     │
│ Model Perplexity     │ &lt;50      │
```

```
| Autonomy Objectives  | ≥1/session
| Consciousness FE     | Decreasing
| End-to-End Latency   | &lt;100ms   |
| Test Pass Rate       | 100%     |
| Code Coverage        | ≥80%     |
| Documentation        | Complete |
| Production Ready     | YES ✅   |
└──────────────────────────────────────┘
```

Comandos de Referência Rápida

```
# Verificar status em qualquer hora
cd ~/projects/omnimind
git log --oneline | head -10  # Ver commits

# Carregar LKM para testes
cd DEVBRAIN_V23/kernel/lkm
sudo insmod devbrain_ai.ko
lsmod | grep devbrain_ai

# Rodar tests
pytest tests/test_kernel_ai.py -v

# Ver documentação
cat docs/KERNEL_INTEGRATION.md
```

PRÓXIMAS FASES (After Phase 10)

Após completar Phase 10, você estará pronto para:

- **Phase 11**: Consciência Aprofundada + Psicoanálise
- **Phase 12**: Produção Hardened (24/7 uptime)
- **Phase 13**: Evolução Contínua

STATUS FINAL: READY FOR EXECUTION ✅

Comece AGORA. Seu Copilot está esperando.