Documentação de Implantação - BioLingo

Grupo nº 14

Sumário Executivo

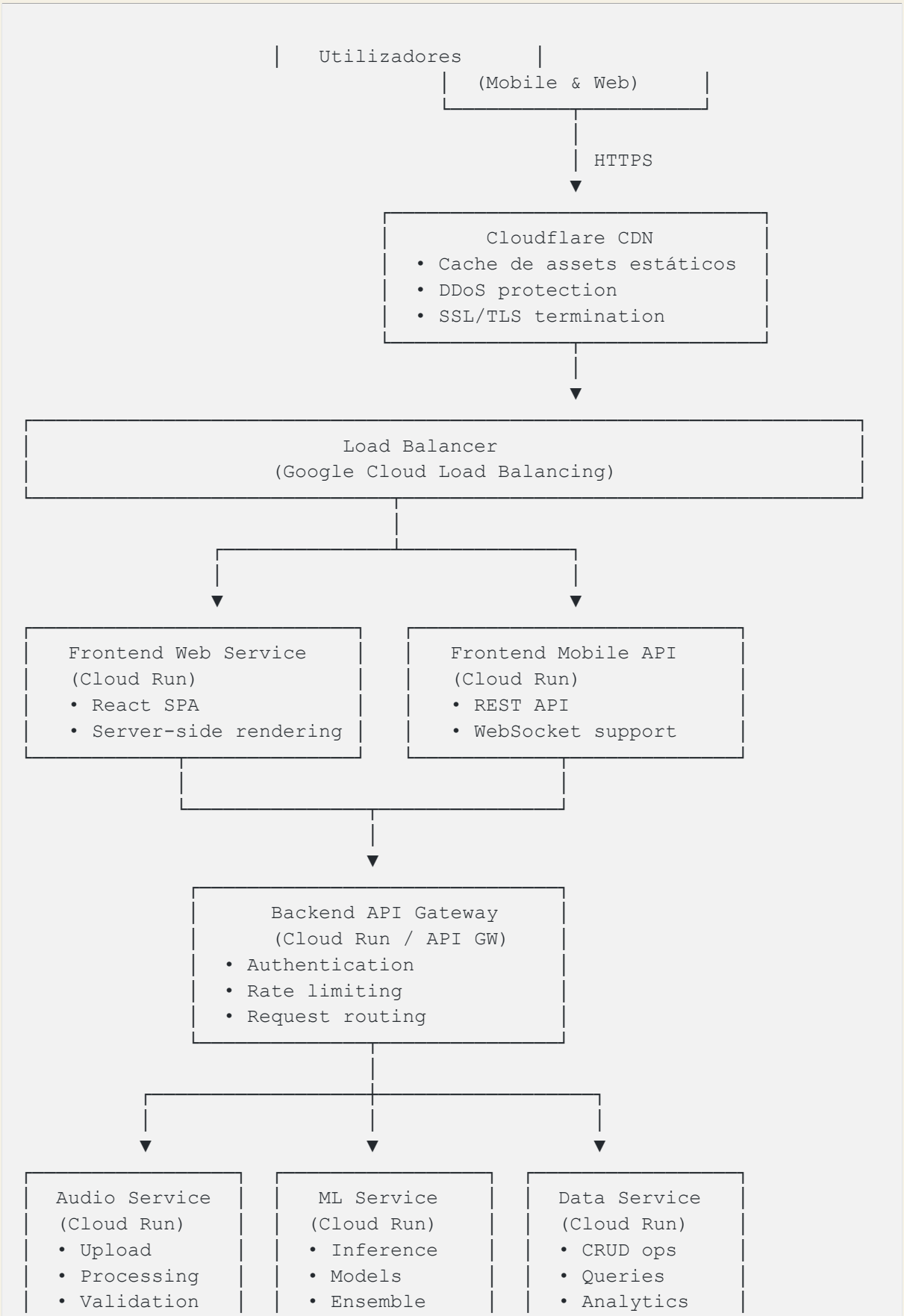
Este documento fornece a documentação completa de implantação (deployment) do sistema BioLingo, desenvolvido pelo Grupo nº 14. Abrange a arquitetura de infraestrutura, configuração de ambientes, processos de CI/CD, estratégias de escalonamento, monitoramento, segurança e procedimentos operacionais. O objetivo é garantir que o sistema seja implantado de forma robusta, segura e escalável, capaz de servir os utilizadores angolanos com alta disponibilidade e desempenho.

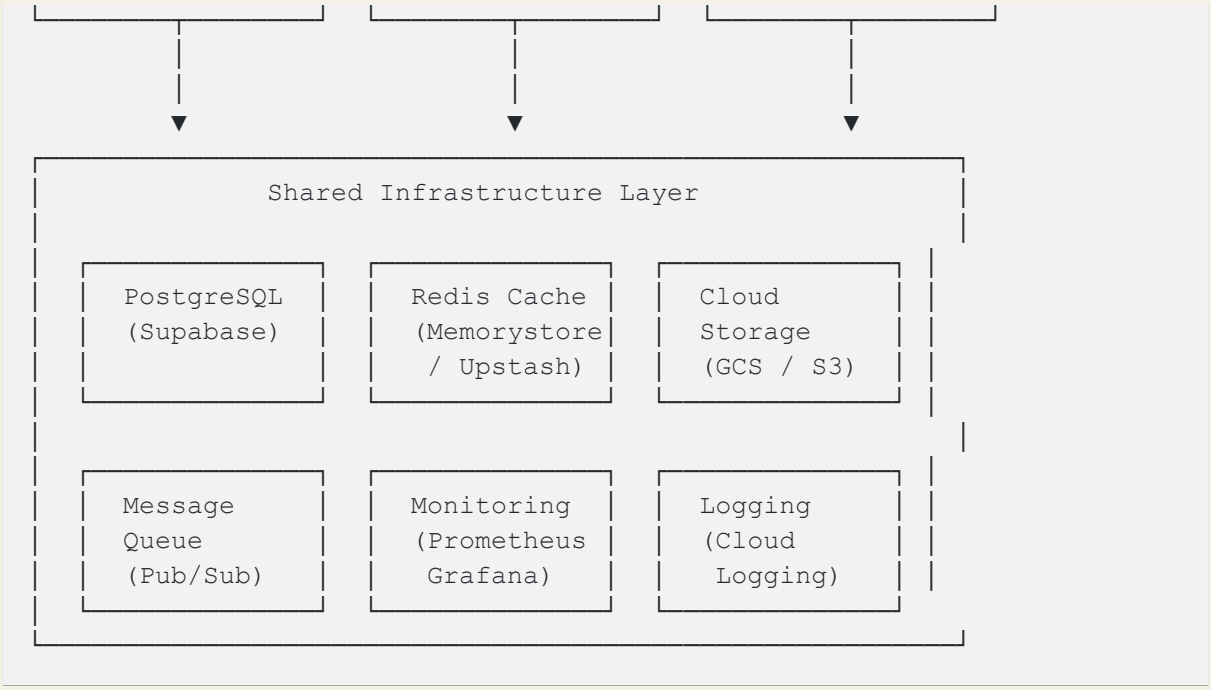
1. Arquitetura de Infraestrutura

1.1. Visão Geral da Arquitetura

O BioLingo utiliza uma arquitetura de microsserviços baseada em nuvem, combinando serviços geridos (managed services) e contêineres para maximizar a eficiência e minimizar custos operacionais.

Figura 1: Arquitetura de Infraestrutura do BioLingo





1.2. Componentes de Infraestrutura

Tabela 1: Componentes de Infraestrutura e Tecnologias

Componente	Tecnologia	Justificação	Custo Estimado (MVP)
CDN & Edge	Cloudflare	Cache global, proteção DDoS, SSL gratuito	$0-20/mês
Load Balancer	Google Cloud Load Balancing	Distribuição de tráfego, health checks	$18/mês
Compute (Containers)	Google Cloud Run	Serverless, escalonamento automático, pay-per-use	$50-150/mês
Database	Supabase (PostgreSQL)	Managed PostgreSQL, real-time, auth integrado	$25/mês
Cache	Upstash Redis	Redis serverless, baixa latência	$10/mês
Object Storage	Google Cloud Storage	Armazenamento de áudio, alta durabilidade	$20-50/mês
Message Queue	Google Pub/Sub	Processamento assíncrono, desacoplamento	$10/mês
Monitoring	Google Cloud Monitoring + Grafana Cloud	Métricas, alertas, dashboards	$0-30/mês
Logging	Google Cloud Logging	Logs centralizados, pesquisa	$10-30/mês
CI/CD	GitHub Actions	Automação de deployment, testes	$0 (gratuito para projetos públicos)
Domain & DNS	Cloudflare	Gestão de DNS, certificados SSL	$0-10/mês
Total Estimado			**$143-343/mês**

1.3. Justificação de Escolhas Tecnológicas

Google Cloud Run: Escolhemos o Cloud Run como plataforma principal de compute devido ao seu modelo serverless, que permite escalonamento automático de 0 a N instâncias baseado na carga, cobrança por uso real (pay-per-request), suporte nativo a contêineres Docker e integração fácil com outros serviços Google Cloud. Para um MVP com tráfego variável, esta é a opção mais económica.

Supabase: Optámos pelo Supabase como backend-as-a-service devido ao PostgreSQL gerido com extensão PostGIS para dados geoespaciais, autenticação e autorização integradas, APIs REST e GraphQL automáticas, real-time subscriptions para atualizações em tempo real e armazenamento de objetos compatível com S3. O plano gratuito é generoso e adequado para o MVP.

Cloudflare: Utilizamos o Cloudflare para CDN e proteção devido ao CDN global gratuito com cache inteligente, proteção DDoS automática, certificados SSL/TLS gratuitos e automáticos, DNS rápido e confiável e analytics de tráfego. Para Angola, com conectividade variável, o cache de edge reduz significativamente a latência.

2. Configuração de Ambientes

2.1. Ambientes de Deployment

O BioLingo utiliza três ambientes distintos para garantir qualidade e estabilidade.

Tabela 2: Configuração de Ambientes

Aspeto	Desenvolvimento	Staging	Produção
Objetivo	Desenvolvimento e testes locais	Testes de integração e validação	Serviço aos utilizadores finais
Infraestrutura	Docker Compose local	Google Cloud (recursos reduzidos)	Google Cloud (recursos completos)
Base de Dados	PostgreSQL local / Supabase (dev)	Supabase (staging project)	Supabase (production project)
Dados	Dados sintéticos / subset pequeno	Cópia anonimizada de produção	Dados reais
Modelos ML	Modelos em treino / experimentais	Modelos candidatos a produção	Modelos validados e aprovados
Monitoramento	Logs locais	Cloud Logging + Grafana	Cloud Logging + Grafana + Alertas
Acesso	Desenvolvedores	Equipa interna + testers	Público geral
URL	localhost:3000	staging.biolingo.ao	www.biolingo.ao

2.2. Variáveis de Ambiente

Cada ambiente utiliza variáveis de ambiente específicas para configuração, armazenadas de forma segura no Google Secret Manager.

Exemplo de Variáveis de Ambiente (Produção):

```
# Aplicação
APP_ENV=production
APP_URL=https://www.biolingo.ao
API_URL=https://api.biolingo.ao
# Base de Dados
DATABASE_URL=postgresql://user:password@host:5432/biolingo_prod
DATABASE_POOL_SIZE=20

# Supabase
SUPABASE_URL=https://xxxxx.supabase.co
SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
SUPABASE_SERVICE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

# Storage
GCS_BUCKET_NAME=biolingo-audio-prod
GCS_PROJECT_ID=biolingo-prod

# Redis Cache
REDIS_URL=redis://default:password@host:6379

# ML Models
MODEL_SPECIES_PATH=gs://biolingo-models/species_v1.2.onnx
MODEL_VOCALIZATION_PATH=gs://biolingo-models/vocalization_v1.1.onnx
MODEL_CONFIDENCE_THRESHOLD=0.70

# APIs Externas
XENO_CANTO_API_KEY=xxxxx
EBIRD_API_KEY=xxxxx
# Autenticação
JWT_SECRET=xxxxx
JWT_EXPIRATION=7d

# Monitoramento
SENTRY_DSN=https://xxxxx@sentry.io/xxxxx
GRAFANA_API_KEY=xxxxx

# Rate Limiting
RATE_LIMIT_FREE_TIER=100
RATE_LIMIT_PREMIUM_TIER=unlimited
```

2.3. Gestão de Segredos

Todos os segredos (passwords, API keys, tokens) são armazenados no Google Secret Manager, nunca em código ou repositórios Git. O acesso aos segredos é controlado por IAM (Identity and Access Management), com permissões mínimas necessárias. Os segredos são rotacionados periodicamente (trimestralmente) e auditados.

3. Processo de CI/CD

3.1. Pipeline de Integração Contínua

Utilizamos GitHub Actions para automatizar o processo de integração contínua, garantindo que o código seja testado e validado antes de ser implantado.

Workflow de CI (.github/workflows/ci.yml):

```
name: Continuous Integration


on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install pytest pytest-cov flake8

      - name: Lint with flake8
        run: |
          flake8 . --count --select=E9,F63,F7,F82 --show-source --
statistics
```

```
          flake8 . --count --exit-zero --max-complexity=10 --max-line-
length=127 --statistics

      - name: Run unit tests
        run: |
          pytest tests/ --cov=app --cov-report=xml --cov-report=html

      - name: Upload coverage to Codecov
        uses: codecov/codecov-action@v3
        with:
          file: ./coverage.xml
          fail_ci_if_error: true

  build:
    runs-on: ubuntu-latest
    needs: test

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      - name: Build Docker image
        run: |
          docker build -t biolingo-api:${{ github.sha }} .

      - name: Run container security scan
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: biolingo-api:${{ github.sha }}
          format: 'sarif'
          output: 'trivy-results.sarif'

      - name: Upload Trivy results to GitHub Security
        uses: github/codeql-action/upload-sarif@v2
        with:
```

```
          sarif_file: 'trivy-results.sarif'
```

3.2. Pipeline de Deployment Contínuo

Workflow de CD para Staging (.github/workflows/deploy-staging.yml):

```yaml
name: Deploy to Staging

on:
  push:
    branches: [develop]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Authenticate to Google Cloud
        uses: google-github-actions/auth@v1
        with:
          credentials_json: ${{ secrets.GCP_SA_KEY_STAGING }}

      - name: Set up Cloud SDK
        uses: google-github-actions/setup-gcloud@v1

      - name: Build and push Docker image
        run: |
          gcloud builds submit --tag gcr.io/${{ secrets.GCP_PROJECT_ID
}}/biolingo-api:${{ github.sha }}

      - name: Deploy to Cloud Run (Staging)
        run: |
          gcloud run deploy biolingo-api-staging \
            --image gcr.io/${{ secrets.GCP_PROJECT_ID }}/biolingo-api:${{
github.sha }} \
            --platform managed \
            --region europe-west1 \
            --allow-unauthenticated \
            --set-env-vars APP_ENV=staging \
            --max-instances 10 \
            --memory 2Gi \
            --cpu 2

      - name: Run smoke tests
        run: |
          curl -f https://staging.biolingo.ao/api/v1/health || exit 1

      - name: Notify team on Slack
```

```
        uses: 8398a7/action-slack@v3
        with:
          status: ${{ job.status }}
          text: 'Deployment to Staging completed'
          webhook_url: ${{ secrets.SLACK_WEBHOOK }}
```

Workflow de CD para Produção (.github/workflows/deploy-production.yml):

```
name: Deploy to Production

on:
  release:
    types: [published]

jobs:
  deploy:
    runs-on: ubuntu-latest
    environment: production

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Authenticate to Google Cloud
        uses: google-github-actions/auth@v1
        with:
          credentials_json: ${{ secrets.GCP_SA_KEY_PROD }}

      - name: Set up Cloud SDK
        uses: google-github-actions/setup-gcloud@v1

      - name: Build and push Docker image
        run: |
          gcloud builds submit --tag gcr.io/${{ secrets.GCP_PROJECT_ID
}}/biolingo-api:${{ github.event.release.tag_name }}

      - name: Deploy to Cloud Run (Production) with gradual rollout
        run: |
          gcloud run deploy biolingo-api-prod \
            --image gcr.io/${{ secrets.GCP_PROJECT_ID }}/biolingo-api:${{
github.event.release.tag_name }} \
            --platform managed \
            --region europe-west1 \
            --allow-unauthenticated \
            --set-env-vars APP_ENV=production \
            --max-instances 50 \
            --min-instances 2 \
```

```
            --memory 4Gi \
            --cpu 4 \
            --no-traffic  # Deploy sem tráfego inicialmente

      - name: Gradual traffic migration (10% -> 50% -> 100%)
        run: |
          # 10% de tráfego para nova versão
          gcloud run services update-traffic biolingo-api-prod \
            --to-revisions LATEST=10
          sleep 300  # Aguardar 5 minutos

          # 50% de tráfego
          gcloud run services update-traffic biolingo-api-prod \
            --to-revisions LATEST=50
          sleep 300

          # 100% de tráfego
          gcloud run services update-traffic biolingo-api-prod \
            --to-revisions LATEST=100

      - name: Run post-deployment tests
        run: |
          curl -f https://www.biolingo.ao/api/v1/health || exit 1
          # Testes adicionais...

      - name: Notify team on Slack
        uses: 8398a7/action-slack@v3
        with:
          status: ${{ job.status }}
          text: 'Deployment to Production completed - Version ${{
github.event.release.tag_name }}'
```

```
          webhook_url: ${{ secrets.SLACK_WEBHOOK }}
```

3.3. Estratégia de Branching

Utilizamos o modelo Git Flow simplificado:

- **main**: Branch de produção, sempre estável e deployável.
- **develop**: Branch de desenvolvimento, integração contínua de features.
- **feature/**: Branches para desenvolvimento de novas funcionalidades.
- **hotfix/**: Branches para correções urgentes em produção.
- **release/**: Branches para preparação de releases.

Processo de Merge:

1. Desenvolvedores criam branches feature/ a partir de develop.
2. Pull Requests são criados para merge em develop.
3. CI executa testes automaticamente.
4. Após aprovação e merge, deploy automático para Staging.
5. Após validação em Staging, criamos uma release/ branch.
6. Merge da release/ em main e criação de tag de versão.
7. Deploy automático para Produção através de GitHub Release.

4. Escalonamento e Alta Disponibilidade

4.1. Escalonamento Automático

Google Cloud Run - Autoscaling:

O Cloud Run escala automaticamente o número de instâncias baseado em:

- **Requisições por segundo**: Cada instância pode lidar com até 80 requisições concorrentes.
- **CPU e memória**: Escalonamento baseado em utilização de recursos.
- **Latência**: Novas instâncias são criadas se a latência exceder limiares.

Configuração de Autoscaling:

```
# service.yaml
```

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: biolingo-api-prod
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/minScale: "2"
        autoscaling.knative.dev/maxScale: "50"
        autoscaling.knative.dev/target: "70"  # 70% de utilização de CPU
    spec:
      containerConcurrency: 80
      containers:
      - image: gcr.io/biolingo-prod/biolingo-api:latest
        resources:
          limits:
            cpu: "4"
            memory: "4Gi"
```

4.2. Alta Disponibilidade

Multi-Region Deployment (Fase futura):

Para garantir alta disponibilidade, planeamos expandir para deployment multi-região:

- **Região Primária**: europe-west1 (Bélgica) - Proximidade relativa a Angola.
- **Região Secundária**: europe-west3 (Frankfurt) - Backup e failover.

Load Balancing Global:

Utilizaremos Google Cloud Load Balancing para distribuir tráfego entre regiões, com failover automático em caso de falha regional.

Database Replication:

O Supabase oferece replicação de leitura (read replicas) para distribuir carga de consultas. Para o MVP, utilizamos uma única instância, mas planeamos adicionar replicas de leitura quando o tráfego aumentar.

4.3. Disaster Recovery

Backups:

- **Base de Dados**: Backups automáticos diários com retenção de 30 dias (Supabase).
- **Armazenamento de Objetos**: Versionamento ativado no Google Cloud Storage, com lifecycle policies para arquivamento.
- **Modelos ML**: Versionamento no MLflow e backup no Google Cloud Storage.

Recovery Time Objective (RTO): < 1 hora para restauração completa do serviço.

Recovery Point Objective (RPO): < 24 horas de perda de dados (backups diários).

Procedimento de Recuperação:

8 Identificar e isolar a falha.
9 Restaurar base de dados a partir do backup mais recente.
10 Redeployar serviços a partir de imagens Docker versionadas.
11 Validar integridade dos dados e funcionalidade do sistema.
12 Redirecionar tráfego para o sistema restaurado.

5. Monitoramento e Observabilidade

5.1. Métricas de Sistema

Google Cloud Monitoring:

Monitorizamos automaticamente métricas de infraestrutura fornecidas pelo Google Cloud:

- **Cloud Run**: Requisições/segundo, latência (p50, p95, p99), taxa de erro, utilização de CPU e memória, número de instâncias ativas.
- **Cloud Storage**: Operações de leitura/escrita, largura de banda, latência.
- **Cloud SQL / Supabase**: Conexões ativas, queries/segundo, latência de queries, utilização de CPU e memória.

Métricas Customizadas:

Implementamos métricas customizadas específicas do BioLingo:

- **Taxa de classificação**: Número de áudios classificados por hora/dia.
- **Distribuição de confiança**: Histograma de níveis de confiança das previsões.
- **Distribuição de espécies**: Contagem de previsões por espécie.
- **Taxa de feedback**: Percentagem de utilizadores que fornecem feedback.
- **Taxa de acerto**: Percentagem de previsões confirmadas como corretas.

5.2. Logging

Google Cloud Logging:

Todos os serviços enviam logs estruturados para o Cloud Logging, incluindo:

- **Logs de aplicação**: Informações, avisos e erros da aplicação.
- **Logs de acesso**: Requisições HTTP com método, path, status, latência, user agent.
- **Logs de auditoria**: Ações de utilizadores (login, upload, feedback).
- **Logs de ML**: Previsões realizadas, tempo de inferência, modelos utilizados.

Formato de Log Estruturado (JSON):

```
{
```

```
 "timestamp": "2025-10-31T14:30:00Z",
  "severity": "INFO",
  "service": "biolingo-api",
  "version": "1.2.0",
  "trace": "projects/biolingo-prod/traces/xxxxx",
  "message": "Audio classified successfully",
  "metadata": {
    "user_id": "user_12345",
    "audio_id": "audio_67890",
    "species_predicted": "Tauraco livingstonii",
    "confidence": 0.92,
    "inference_time_ms": 1250
  }
}
```

5.3. Dashboards e Visualização

Grafana Dashboards:

Criamos dashboards customizados no Grafana para visualização em tempo real:

Dashboard 1: Visão Geral do Sistema

- Requisições por segundo (RPS)
- Latência média e percentis (p50, p95, p99)
- Taxa de erro (4xx, 5xx)
- Número de instâncias ativas
- Utilização de recursos (CPU, memória)

Dashboard 2: Métricas de ML

- Classificações por hora/dia
- Distribuição de confiança
- Top 10 espécies identificadas
- Taxa de feedback e acerto
- Tempo de inferência

Dashboard 3: Métricas de Negócio

- Utilizadores ativos (DAU, MAU)
- Novos registos
- Gravações submetidas
- Distribuição geográfica de utilizadores
- Taxa de retenção

5.4. Alertas

Configuração de Alertas:

Utilizamos Google Cloud Monitoring e Grafana para configurar alertas automáticos:

Tabela 3: Alertas Configurados

Alerta	Condição	Severidade	Canal de Notificação
Alta Latência	Latência p95 > 5s por 5 min	Crítico	Slack + Email + SMS
Taxa de Erro Elevada	Taxa de erro > 5% por 5 min	Crítico	Slack + Email + SMS
Downtime de Serviço	Health check falha por 2 min	Crítico	Slack + Email + SMS
Utilização de Recursos	CPU > 80% por 10 min	Aviso	Slack
Queda na Confiança	Confiança média < 0.70 por 1h	Aviso	Slack + Email
Aumento Anómalo de Tráfego	RPS > 200% da média por 10 min	Informação	Slack
Falha de Backup	Backup diário falha	Crítico	Email

Canais de Notificação:

- **Slack**: Canal #biolingo-alerts para toda a equipa.
- **Email**: Lista de distribuição da equipa técnica.
- **SMS**: Apenas para alertas críticos, enviados para o engenheiro de plantão.

6. Segurança

6.1. Autenticação e Autorização

Autenticação de Utilizadores:

Utilizamos o Supabase Auth para autenticação de utilizadores, suportando múltiplos métodos:

- **Email/Password**: Registo tradicional com verificação de email.
- **OAuth**: Login social via Google, Facebook (fase futura).
- **Magic Links**: Login sem password via email (fase futura).

Tokens JWT:

Após autenticação bem-sucedida, o utilizador recebe um token JWT (JSON Web Token) que deve ser incluído em todas as requisições subsequentes no header Authorization: Bearer <token>.

Autorização Baseada em Roles:

Implementamos controle de acesso baseado em roles (RBAC):

- **Public**: Acesso a endpoints públicos (health check, listagem de espécies).
- **User**: Utilizadores registados (upload de áudio, histórico, feedback).
- **Researcher**: Investigadores (acesso a dados agregados, estatísticas avançadas).
- **Admin**: Administradores (gestão de utilizadores, moderação de conteúdo).

6.2. Proteção de APIs

Rate Limiting:

Implementamos rate limiting para prevenir abuso e garantir disponibilidade:

- **Utilizadores não autenticados**: 10 requisições/hora.
- **Utilizadores gratuitos**: 100 requisições/hora.
- **Utilizadores premium**: 1000 requisições/hora.
- **Investigadores**: Ilimitado (com aprovação).

Validação de Entrada:

Todas as entradas de utilizadores são validadas rigorosamente:

- **Tamanho de ficheiros**: Máximo 10 MB para uploads de áudio.
- **Formatos aceites**: WAV, MP3, M4A, OGG.
- **Sanitização**: Remoção de caracteres especiais e potencialmente maliciosos.
- **Validação de tipos**: Verificação de MIME types e magic numbers.

CORS (Cross-Origin Resource Sharing):

Configuramos CORS para permitir apenas origens autorizadas:

```
from fastapi.middleware.cors import CORSMiddleware


app.add_middleware(
    CORSMiddleware,
    allow_origins=[
        "https://www.biolingo.ao",
        "https://app.biolingo.ao",
        "https://staging.biolingo.ao"
    ],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE"],
    allow_headers=["*"],
)
```

6.3. Encriptação

Encriptação em Trânsito:

Todas as comunicações utilizam HTTPS/TLS 1.3, com certificados SSL geridos automaticamente pelo Cloudflare e Google Cloud. Não permitimos conexões HTTP não encriptadas.

Encriptação em Repouso:

Todos os dados armazenados são encriptados em repouso:

- **Base de Dados**: Encriptação nativa do PostgreSQL/Supabase.
- **Armazenamento de Objetos**: Encriptação AES-256 no Google Cloud Storage.
- **Segredos**: Encriptação no Google Secret Manager.

6.4. Conformidade e Privacidade

GDPR e Proteção de Dados:

Embora Angola não esteja sob jurisdição do GDPR, seguimos as melhores práticas de proteção de dados:

- **Consentimento**: Utilizadores devem consentir explicitamente com a recolha de dados.
- **Direito ao Acesso**: Utilizadores podem solicitar acesso aos seus dados.

- **Direito ao Esquecimento**: Utilizadores podem solicitar eliminação dos seus dados.
- **Minimização de Dados**: Recolhemos apenas dados estritamente necessários.
- **Anonimização**: Dados para investigação são anonimizados.

Política de Privacidade:

Disponibilizamos uma política de privacidade clara e acessível em https://www.biolingo.ao/privacy, detalhando que dados recolhemos, como os utilizamos, com quem os partilhamos e como os protegemos.

7. Procedimentos Operacionais

7.1. Deployment de Nova Versão

Processo Standard:

13 **Desenvolvimento**: Criar branch feature/ e desenvolver funcionalidade.
14 **Testes Locais**: Executar testes unitários e de integração localmente.
15 **Pull Request**: Criar PR para develop, aguardar revisão de código.
16 **CI**: GitHub Actions executa testes automaticamente.
17 **Merge**: Após aprovação, fazer merge em develop.
18 **Deploy Staging**: Deploy automático para ambiente de staging.
19 **Testes em Staging**: Executar testes manuais e automatizados.
20 **Release Branch**: Criar branch release/vX.Y.Z.
21 **Merge em Main**: Fazer merge da release em main e criar tag.
22 **GitHub Release**: Criar release no GitHub.
23 **Deploy Produção**: Deploy automático para produção com rollout gradual.
24 **Monitoramento**: Monitorizar métricas e logs durante 24h.

7.2. Rollback

Quando fazer Rollback:

- Taxa de erro > 10% após deployment.
- Latência aumenta > 200% da baseline.
- Funcionalidade crítica quebrada.
- Feedback negativo massivo de utilizadores.

Processo de Rollback:

```
# Listar revisões disponíveis
```

```
gcloud run revisions list --service biolingo-api-prod

# Reverter para revisão anterior
gcloud run services update-traffic biolingo-api-prod \
  --to-revisions biolingo-api-prod-00042-abc=100

# Verificar saúde do serviço
curl -f https://www.biolingo.ao/api/v1/health

# Notificar equipa
# Investigar causa raiz
# Corrigir problema
```

```
# Redeployar quando pronto
```

7.3. Atualização de Modelos ML

Processo de Atualização de Modelos:

25 **Treino**: Treinar novo modelo com dados atualizados.
26 **Validação**: Avaliar desempenho em conjunto de teste.
27 **Registo**: Registar modelo no MLflow com métricas e metadados.
28 **Upload**: Fazer upload do modelo para Google Cloud Storage.
29 **Teste em Staging**: Atualizar variável de ambiente MODEL_SPECIES_PATH em staging.
30 **A/B Testing**: Implementar A/B testing em staging (50% tráfego para novo modelo).
31 **Análise**: Comparar métricas de desempenho entre modelos.
32 **Decisão**: Se novo modelo for superior, aprovar para produção.
33 **Deploy Produção**: Atualizar variável de ambiente em produção.
34 **Monitoramento**: Monitorizar métricas de ML durante 7 dias.

Rollback de Modelo:

Se o novo modelo apresentar desempenho inferior, simplesmente revertemos a variável de ambiente para apontar para o modelo anterior:

```
gcloud run services update biolingo-api-prod \
  --set-env-vars MODEL_SPECIES_PATH=gs://biolingo-
models/species_v1.1.onnx
```

7.4. Manutenção Programada

Janelas de Manutenção:

Realizamos manutenção programada mensalmente, sempre aos domingos entre 02:00-04:00 WAT (horário de Angola), período de menor tráfego.

Comunicação:

- **7 dias antes**: Anúncio na aplicação e email para utilizadores.
- **24 horas antes**: Notificação push para utilizadores móveis.
- **Durante**: Banner na aplicação indicando manutenção.

Checklist de Manutenção:

- ☐Actualizar dependências de segurança.
- ☐Aplicar patches de sistema operativo.
- ☐Optimizar base de dados (VACUUM, ANALYZE).
- ☐Limpar logs antigos (> 90 dias).
- ☐Verificar integridade de backups.
- ☐Actualizar certificados SSL (se necessário).
- ☐Revisar e ajustar configurações de autoscaling.
- ☐Executar testes de carga.

8. Custos e Optimização

8.1. Estimativa de Custos

Tabela 4: Estimativa de Custos Mensal por Fase

Componente	MVP (0-1k users)	Crescimento (1k-10k users)	Escala (10k-100k users)
Compute (Cloud Run)	$50-150	$200-500	$1000-3000
Database (Supabase)	$25	$100	$500
Storage (GCS)	$20-50	$100-200	$500-1000
CDN (Cloudflare)	$0-20	$50-100	$200-500
Monitoring & Logging	$10-30	$50-100	$200-400
Outros	$38-93	$100-200	$300-600
Total Estimado	**$143-343**	**$600-1200**	**$2700-6000**

8.2. Estratégias de Otimização de Custos

Cache Agressivo:

Implementamos cache em múltiplas camadas para reduzir custos de compute e largura de banda:

- **CDN (Cloudflare)**: Cache de assets estáticos (imagens, CSS, JS) por 30 dias.
- **Redis**: Cache de respostas de API frequentes (listagem de espécies, informações educativas) por 1 hora.
- **Browser Cache**: Cache de recursos no navegador do utilizador.

Compressão de Áudio:

Convertemos todos os uploads de áudio para formato OGG Opus, que oferece excelente qualidade com tamanho reduzido (até 50% menor que MP3).

Lifecycle Policies:

Configuramos políticas de ciclo de vida no Google Cloud Storage:

- **Áudio bruto**: Movido para Nearline Storage após 30 dias (custo 50% menor).
- **Áudio processado**: Movido para Coldline Storage após 90 dias (custo 80% menor).
- **Logs**: Eliminados após 90 dias.

Optimização de Modelos:

Utilizamos quantização e ONNX Runtime para reduzir o tamanho dos modelos e acelerar inferência, reduzindo tempo de compute e custos.

9. Conclusão

A documentação de implantação do BioLingo apresenta uma estratégia robusta, escalável e económica para levar o sistema a produção. Através da utilização de serviços geridos de nuvem, contêinerização, CI/CD automatizado e monitoramento abrangente, o Grupo nº 14 garante que o sistema será confiável, seguro e capaz de servir os utilizadores angolanos com alta qualidade.

A arquitetura proposta permite começar com custos baixos no MVP (aproximadamente $143-343/mês) e escalar gradualmente conforme a adoção aumenta, mantendo sempre a eficiência operacional. O compromisso com as melhores práticas de DevOps, segurança e observabilidade assegura que o BioLingo será uma plataforma sustentável e de longo prazo para a conservação da biodiversidade em Angola.

Anexos

Anexo A: Checklist de Pre-Launch

- ☐Todos os testes automatizados passam (unit, integration, e2e).
- ☐Testes de carga realizados e aprovados (> 100 RPS).
- ☐Modelos ML validados e com desempenho acima dos limiares.
- ☐Documentação de API completa e publicada.
- ☐Política de privacidade e termos de serviço publicados.
- ☐Monitoramento e alertas configurados e testados.
- ☐Backups automáticos configurados e testados.
- ☐Procedimentos de rollback documentados e testados.
- ☐Plano de comunicação de incidentes preparado.
- ☐Equipa de suporte treinada e pronta.
- ☐Domínio configurado e certificados SSL ativos.
- ☐Testes de segurança realizados (OWASP Top 10).
- ☐Aprovação final de stakeholders.

Anexo B: Fornecedores

Fornecedores:

- **Google Cloud Support**: Plano Standard - 24/7
- **Supabase Support**: Email support@supabase.io
- **Cloudflare Support**: Dashboard de suporte