Pruebas de Arquitectura

Arquitectura del Sistema

El sistema cuenta con módulos para autenticación, gestión de usuarios y roles, registro y consulta de datos, entrenamiento de modelos de IA, predicciones y visualización de resultados.

La arquitectura está enfocada en ser una arquitectura monolítica con un stack tecnológico moderno, que permita escalar y mantener fácilmente las funcionalidades del sistema.

Frontend del Sistema (Interfaz de Usuario)

- Framework: React + Vite (Typescript)
- UI Library: Material UI (MUI)
- Estado global: Zustand
- Ruteo: React Router
- Autenticación: Manejo de sesión vía JWT + Cookies seguras
- Visualizaciones: Chart.js

Backend del Sistema (lógica de la aplicación)

- Framework: Flask (Python 3.11+)
- Servidor de producción: Gunicorn
- Estrucutra modular: Blueprints (auth, cattle, datasets, training, prediction, reports, dashboard)
- Manejo de errores: Flask ErrorHandler + Custom JSON responses
- Autenticación: JWT + bcrypt
- Roles y permisos: Decoradores personalizados requires_role('admin')
- Encolado de tareas: Celery (integrado con Redis)
- Validación de archivos: Pandas (formato CSV, duplicados, nulos)
- Generación de reportes: Matplotlib / ReportLab (PDF o CSV)

Procesamiento Asíncrono y Aprendizaje Automático

- Motor de tareas: Celery
- Broker y backend de resultados: Redis
- Framework ML: scikit-learn / PyTorch
- Procesos asíncronos:
    - Validación de datasets
    - Entrenamiento de modelos
    - Evaluación y métricas

    - Generación de reportes
    - Cálculo de riesgo y alertas
- Almacenamiento de modelos: S3

Base de Datos

- Sistema: PostgreSQL
- Entidades principales:
    - Usuarios: id, nombre, correo, contraseña, rol
    - Roles: id, nombre, permisos
    - Vacas: id, raza, edad, periodo de transición, estado
    - Datos de Salud: id_vaca, fecha, variable, valor
    - Datasets: id, nombre, ruta_archivo, fecha_subida, validez
    - Modelos: id, nombre, versión, métricas, fecha_entrenamiento
    - Alertas: id_vaca, riesgo, fecha, observaciones

Capa de Cache y Sesiones

- Usos:
    - Almacenamiento temporal de tokens de sesión
    - Cache de predicciones recientes y métricas intermedias
    - Backend de resultados para Celery
    - Mecanismo de notificaciones en tiempo real (pub/sub)

Servidor Web y Proxy

- Servidor: Nginx
- Funciones:
    - Proxy reverso hacia Flask (Gunicorn)
    - Servir archivos estáticos del frontend (build de React)
    - Terminación SSL (Let's Encrypt / Certbot)
    - Compresión (gzip) y cacheo básico
    - Balanceo de carga si existen múltiples instancias Flask
    - Redirección de tráfico HTTP → HTTPS
    - Rutas:
        - /api/* → backend Flask
        - / → frontend React

Almacenamiento de Archivos

- Opción Cloud: AWS S3
- Opción Local: MinIO
- Contenido:
    - Archivos CSV subidos
    - Modelos entrenados (.pkl, .pt)

    - Reportes generados (PDF / CSV)
- Integración directa con Flask y Celery para lectura/escritura

Seguridad

- Autenticación: JWT + Cookies seguras (HttpOnly)
- Contraseñas: bcrypt (hash + salt)
- Roles: admin, veterinario, investigador, usuario
- Middleware de validación por rol y permisos
- Protección CSRF: Flask-WTF (en formularios, si aplica)
- HTTPS forzado vía Nginx
- CORS restringido a dominios específicos del frontend

Despliegue y DevOps

- Contenedores: Docker + Docker Compose
- Orquestación (si se escala): Docker Swarm
- CI/CD: GitHub Actions
- Monitoreo: Flask-Logging + CloudWatch
- Variables de entorno: .env + python-decouple
- Infraestructura reproducible: Terraform (opcional)

Frontend — React + Vite + Atomic Design + Clean Architecture

None

```
📦 frontend/
├── 📁 src/
│   ├── 📁 app/                              # Capa de
configuración y orquestación
│   │   ├── App.tsx
│   │   ├── main.tsx
│   │   ├── routes/
│   │   │   ├── index.tsx
│   │   │   ├── ProtectedRoute.tsx
│   │   │   └── AuthRoute.tsx
│   │   ├── providers/
│   │   │   ├── AuthProvider.tsx
│   │   │   ├── ThemeProvider.tsx
│   │   │   └── QueryProvider.tsx
│   │   └── hooks/
│   │       ├── useAuth.ts
│   │       ├── useTheme.ts
```

```
│   │       └── useFetch.ts
│   │
│   ├── 📁 domain/                          # Capa de dominio
│   │   ├── entities/
│   │   │   ├── Cow.ts
│   │   │   ├── Dataset.ts
│   │   │   ├── User.ts
│   │   │   └── Model.ts
│   │   ├── factories/
│   │   │   └── AuthValidatorFactory.ts
│   │   ├── repositories/
│   │   │   ├── ICowRepository.ts
│   │   │   ├── IUserRepository.ts
│   │   │   └── IDatasetRepository.ts
│   │   ├── validations/
│   │   │   ├── auth/
│   │   │   │   ├── PasswordValidation.ts
│   │   │   │   └── EmailValidation.ts
│   │   │   └── IValidation.ts
│   │   ├── usecases/
│   │   │   ├── RegisterUser.ts
│   │   │   ├── TrainModel.ts
│   │   │   ├── PredictHealth.ts
│   │   │   └── UploadDataset.ts
│   │   └── events/                         # 🆕 Domain Events
│   │       ├── CowHealthChanged.ts
│   │       ├── DatasetUploaded.ts
│   │       ├── ModelTrained.ts
│   │       └── UserRegistered.ts
│   │
│   ├── 📁 infrastructure/
│   │   ├── api/
│   │   │   ├── axiosConfig.ts
│   │   │   ├── AuthAPI.ts
│   │   │   ├── CowAPI.ts
│   │   │   ├── DatasetAPI.ts
│   │   │   └── ModelAPI.ts
│   │   ├── storage/
│   │   │   └── LocalStorageService.ts
```

```
|   |   └── adapters/
|   |       └── repositoryAdapters.ts
|
|   ├── 📁 presentation/
|   |   ├── 📁 components/
|   |   |   ├── 📁 atoms/
|   |   |   |   ├── Button.tsx
|   |   |   |   ├── Input.tsx
|   |   |   |   ├── Label.tsx
|   |   |   |   └── Icon.tsx
|   |   |   ├── 📁 molecules/
|   |   |   |   ├── FormField.tsx
|   |   |   |   ├── CardInfo.tsx
|   |   |   |   └── AlertBox.tsx
|   |   |   ├── 📁 organisms/
|   |   |   |   ├── LoginForm.tsx
|   |   |   |   ├── RegisterForm.tsx
|   |   |   |   ├── DatasetUploader.tsx
|   |   |   |   └── CowHealthForm.tsx
|   |   |   ├── 📁 templates/
|   |   |   |   ├── DashboardTemplate.tsx
|   |   |   |   ├── AuthTemplate.tsx
|   |   |   |   └── ModelTrainingTemplate.tsx
|   |   |   └── 📁 pages/
|   |   |       ├── LoginPage.tsx
|   |   |       ├── DashboardPage.tsx
|   |   |       ├── CowDetailsPage.tsx
|   |   |       ├── TrainingPage.tsx
|   |   |       └── ReportsPage.tsx
|   |   ├── 📁 hooks/                    # 🆕 Custom hooks
usando patrones
|   |   |   ├── useObserver.ts           # Hook para Observer
pattern
|   |   |   ├── useCommand.ts            # Hook para Command
pattern
|   |   |   ├── useStrategy.ts           # Hook para Strategy
pattern
|   |   |   ├── useFactory.ts            # Hook para Factory
pattern
```

```
|   |   |   └── useState.ts                 # Hook para State
pattern
|   |   └── styles/
|   |       ├── theme.ts
|   |       └── globals.css
|
|   ├── 📁 utils/
|   |   ├── constants.ts
|   |   ├── formatters.ts
|   |   └── validators.ts
|
|   └── 📁 tests/
|       ├── unit/
|       |   ├── components/
|       |   └── usecases/
|       ├── integration/
|       └── e2e/
|
├── tsconfig.json
├── vite.config.ts
└── package.json
```

2. Backend — Flask + Clean Architecture + Redis + Celery

None

```
📦 backend/
├── 📁 src/
|   ├── __init__.py
|   ├── main.py                           # Punto de entrada
(Gunicorn)
|   ├── config.py                         # Configuración
global
|   ├── 📁 domain/                        # Entidades del
dominio
|   |   ├── 📁 entities/
|   |   |   ├── cow.py
|   |   |   ├── user.py
|   |   |   ├── dataset.py
```

```
|   |   |   ├── model.py
|   |   |   └── alert.py
|   |   |
|   |   ├── 📁 value_objects/            # 🆕 Value Objects
(DDD)
|   |   |   ├── email.py
|   |   |   ├── health_score.py
|   |   |   ├── date_range.py
|   |   |   └── coordinates.py
|   |   |
|   |   ├── 📁 events/                   # 🆕 Domain Events
|   |   |   ├── base_event.py
|   |   |   ├── cow_registered_event.py
|   |   |   ├── model_trained_event.py
|   |   |   ├── prediction_made_event.py
|   |   |   ├── alert_triggered_event.py
|   |   |   └── dataset_uploaded_event.py
|   |   |
|   |   ├── 📁 repositories/             # Interfaces de
repositorios
|   |   |   ├── base_repository.py
|   |   |   ├── cow_repository.py
|   |   |   ├── user_repository.py
|   |   |   ├── dataset_repository.py
|   |   |   └── model_repository.py
|   |   |
|   |   ├── 📁 services/                 # Servicios de
dominio
|   |   |   ├── risk_service.py
|   |   |   ├── training_service.py
|   |   |   ├── prediction_service.py
|   |   |   └── notification_service.py
|   |   |
|   |   └── 📁 specifications/           # 🆕 Specification
Pattern
|   |       ├── specification_interface.py
|   |       ├── cow_specifications.py
|   |       ├── dataset_specifications.py
|   |       └── user_specifications.py
```

```
│
│   ├── 📁 application/                    # 🆕 Capa de
aplicación (casos de uso)
│   │   ├── 📁 usecases/
│   │   │   ├── 📁 auth/
│   │   │   │   ├── register_user.py
│   │   │   │   ├── login_user.py
│   │   │   │   └── refresh_token.py
│   │   │   ├── 📁 cattle/
│   │   │   │   ├── register_cow.py
│   │   │   │   ├── update_cow_health.py
│   │   │   │   └── get_cow_history.py
│   │   │   ├── 📁 dataset/
│   │   │   │   ├── upload_dataset.py
│   │   │   │   ├── validate_dataset.py
│   │   │   │   └── process_dataset.py
│   │   │   ├── 📁 model/
│   │   │   │   ├── train_model.py
│   │   │   │   ├── evaluate_model.py
│   │   │   │   └── deploy_model.py
│   │   │   ├── 📁 prediction/
│   │   │   │   ├── make_prediction.py
│   │   │   │   └── batch_prediction.py
│   │   │   └── 📁 reporting/
│   │   │       ├── generate_report.py
│   │   │       └── schedule_report.py
│   │   │
│   │   ├── 📁 dto/                        # 🆕 Data Transfer
Objects
│   │   │   ├── user_dto.py
│   │   │   ├── cow_dto.py
│   │   │   ├── prediction_dto.py
│   │   │   └── report_dto.py
│   │   │
│   │   └── 📁 mappers/                    # 🆕 Mappers (Entity
↔ DTO)
│   │       ├── user_mapper.py
│   │       ├── cow_mapper.py
│   │       └── dataset_mapper.py
```

```
│
│   ├── 📁 infrastructure/                # Implementaciones
técnicas
│   │   ├── 📁 orm/
│   │   │   ├── models.py
│   │   │   ├── base.py
│   │   │   └── migrations/
│   │   │
│   │   ├── 📁 repositories/              # Implementaciones
concretas
│   │   │   ├── base_sql_repository.py
│   │   │   ├── sql_user_repository.py
│   │   │   ├── sql_cow_repository.py
│   │   │   ├── sql_dataset_repository.py
│   │   │   └── sql_model_repository.py
│   │   │
│   │   ├── 📁 cache/                     # 🆕 Cache
implementations
│   │   │   ├── redis_cache.py
│   │   │   ├── in_memory_cache.py
│   │   │   └── cache_manager.py
│   │   │
│   │   ├── 📁 files/
│   │   │   ├── storage_interface.py
│   │   │   ├── s3_service.py
│   │   │   ├── azure_blob_service.py
│   │   │   └── local_storage.py
│   │   │
│   │   ├── 📁 ml/
│   │   │   ├── trainer.py
│   │   │   ├── predictor.py
│   │   │   ├── metrics.py
│   │   │   ├── feature_engineering.py
│   │   │   └── model_registry.py
│   │   │
│   │   ├── 📁 messaging/                 # 🆕 Message Queue /
Event Bus
│   │   │   ├── event_bus.py
│   │   │   ├── rabbitmq_client.py
```

```
|   |   |   └── kafka_producer.py
|   |   |
|   |   └── 📁 external/                  # 🆕 Servicios
externos
|   |       ├── email_service.py
|   |       ├── sms_service.py
|   |       └── webhook_service.py
|
|   ├── 📁 presentation/                  # API Layer
|   |   ├── 📁 api/
|   |   |   ├── 📁 v1/
|   |   |   |   ├── __init__.py
|   |   |   |   ├── auth.py
|   |   |   |   ├── cattle.py
|   |   |   |   ├── datasets.py
|   |   |   |   ├── models.py
|   |   |   |   ├── predictions.py
|   |   |   |   └── reports.py
|   |   |   └── dependencies.py           # Dependency
injection
|   |   |
|   |   ├── 📁 schemas/                   # Pydantic schemas
|   |   |   ├── base_schema.py
|   |   |   ├── user_schema.py
|   |   |   ├── cow_schema.py
|   |   |   ├── dataset_schema.py
|   |   |   ├── prediction_schema.py
|   |   |   └── response_schema.py
|   |   |
|   |   ├── 📁 middleware/
|   |   |   ├── auth_middleware.py
|   |   |   ├── cors_middleware.py
|   |   |   ├── rate_limit_middleware.py
|   |   |   ├── logging_middleware.py
|   |   |   └── error_handler_middleware.py
|   |   |
|   |   └── 📁 validators/                # 🆕 Request
validators
|   |       ├── request_validator.py
```

```
|   |       └── custom_validators.py
|
|   ├── 📁 tasks/                          # Tareas Celery
asíncronas
|   |   ├── base_task.py                   # 🆕 Base task con
patterns
|   |   ├── dataset_tasks.py
|   |   ├── training_tasks.py
|   |   ├── prediction_tasks.py
|   |   ├── report_tasks.py
|   |   └── notification_tasks.py
|
|   └── 📁 tests/
|       ├── 📁 unit/
|       |   ├── 📁 domain/
|       |   ├── 📁 usecases/
|       |   └── 📁 services/
|       ├── 📁 integration/
|       |   ├── test_api.py
|       |   ├── test_repository.py
|       |   └── test_ml_pipeline.py
|       ├── 📁 e2e/
|       |   └── test_full_workflow.
|       └── 📁 fixtures/
|           ├── database_fixtures.py
|           └── mock_data.py
|
├── 📁 scripts/                            # 🆕 Scripts de
utilidad
|   ├── seed_database.py
|   ├── migrate_data.py
|   └── generate_test_data.py

|
├── requirements.txt
├── requirements-dev.txt                   # 🆕 Dependencias de
desarrollo
├── Dockerfile
├── docker-compose.yml
```

```
├── .env.example
├── alembic.ini                          # 🆕 Migraciones
└── README.md
```