EventVault

High-Level Architectural Design

System Overview

EventVault is a comprehensive event financial management platform designed to serve multiple stakeholders in the events industry with varying access levels and permissions. The system requires secure data handling, complex financial calculations, document storage, integration capabilities, and responsive user interfaces.

Architecture Diagram

```
┌──────────────────────────────────────────────────────────────────────────┐
│                                                                          │
│                           CLIENT LAYER                                   │
│                                                                          │
│  ┌───────────────┐   ┌───────────────┐   ┌─────────────────────────┐    │
│  │ Web           │   │ Mobile        │   │  Progressive            │    │
│  │ Application   │   │ Application   │   │  Web App (PWA)          │    │
│  └───────────────┘   └───────────────┘   └─────────────────────────┘    │
│            │                  │                      │                   │
└──────────────────────────────────────────────────────────────────────────┘
             │                  │                      │
┌──────────────────────────────────────────────────────────────────────────┐
│            │                  │                      │                   │
│            │      API GATEWAY / LOAD BALANCER        │                   │
│            │                                         │                   │
└──────────────────────────────────────────────────────────────────────────┘
             │                                         │
┌──────────────────────────────────────────────────────────────────────────┐
│            │                                         │                   │
│     MICROSERVICES ARCHITECTURE / APPLICATION LAYER   │                   │
│                                                                          │
│  ┌───────────────┐   ┌───────────────┐   ┌─────────────────────────┐    │
│  │               │   │               │   │                         │    │
│  │ Auth Service  │   │ Event Service │   │ Budget Service          │    │
│  │               │   │               │   │                         │    │
│  └───────────────┘   └───────────────┘   └─────────────────────────┘    │
│                                                                          │
│  ┌───────────────┐   ┌───────────────┐   ┌─────────────────────────┐    │
│  │               │   │               │   │                         │    │
│  │ User Service  │   │ Finance       │   │ Document Service        │    │
│  │               │   │ Service       │   │                         │    │
│  └───────────────┘   └───────────────┘   └─────────────────────────┘    │
│                                                                          │
│  ┌───────────────┐   ┌───────────────┐   ┌─────────────────────────┐    │
│  │               │   │               │   │                         │    │
│  │ Supplier      │   │ Analytics     │   │ Integration Service     │    │
│  │ Service       │   │ Service       │   │                         │    │
│  └───────────────┘   └───────────────┘   └─────────────────────────┘    │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
             │                                         │
┌──────────────────────────────────────────────────────────────────────────┐
│            │                                         │                   │
│                           DATA LAYER                                     │
│                                                                          │
│  ┌───────────────┐   ┌───────────────┐   ┌─────────────────────────┐    │
│  │ Relational    │   │ Document      │   │ Object Storage          │    │
│  │ Database      │   │ Database      │   │ (Files/Documents)       │    │
│  └───────────────┘   └───────────────┘   └─────────────────────────┘    │
│                                                                          │
│  ┌─────────────────────────────────────────────────────────────────┐    │
│  │                         Cache Layer                             │  │ │
│  └─────────────────────────────────────────────────────────────────┘    │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

Technology Stack

Frontend (Client Layer)

1. **Web Application**
    - **Framework**: React.js
    - **UI Components**: Material UI or Tailwind CSS
    - **State Management**: Redux or Context API
    - **Data Visualization**: Recharts or D3.js for financial reports and visualizations
2. **Mobile Application** (Future phase)
    - **Framework**: React Native for cross-platform capability
3. **Progressive Web App**
    - Service workers for offline capabilities
    - Responsive design for tablet use on event sites

API Gateway & Communication

- **API Gateway**: AWS API Gateway or Kong
- **Authentication**: OAuth 2.0 + JWT
- **Communication Protocol**: RESTful APIs with GraphQL for complex data queries
- **WebSockets**: For real-time updates on budget changes and approvals

Backend (Application Layer)

1. **Core Backend Framework**
    - **Primary Language**: Node.js with Express or NestJS
    - **Alternative Option**: Python with Django or FastAPI for financial computing
2. **Microservices**
    - Each service containerized with Docker
    - Orchestration with Kubernetes for scalability
    - Service mesh (Istio or Linkerd) for service-to-service communication

Data Layer

1. **Relational Database**
    - **Technology**: PostgreSQL
    - **Purpose**: Financial data, relationships, transactions
    - **Features needed**: Strong transaction support and financial integrity
2. **Document Database**
    - **Technology**: MongoDB
    - **Purpose**: Flexible documents, event templates, configurations
    - **Features needed**: Flexible schema for different event types
3. **Object Storage**
    - **Technology**: AWS S3 or Google Cloud Storage
    - **Purpose**: Document storage (invoices, contracts, quotes)
    - **Features needed**: Versioning, access control, thumbnail generation

4. **Cache Layer**
    - **Technology**: Redis
    - **Purpose**: Session management, frequently accessed data
    - **Features needed**: High performance for dashboard calculations

DevOps & Infrastructure

1. **Hosting**: AWS, Google Cloud, or Azure
2. **CI/CD**: GitHub Actions or Jenkins
3. **Infrastructure as Code**: Terraform or CloudFormation
4. **Monitoring**: Prometheus with Grafana dashboards
5. **Logging**: ELK Stack (Elasticsearch, Logstash, Kibana)

Core Services Description

1. Auth Service

- User authentication and authorization
- Role-based access control management
- Single sign-on capabilities
- Partner/third-party access management

2. Event Service

- Event creation and management
- Event template management
- Event timeline and scheduling
- Assignment of stakeholders to events

3. Budget Service

- Budget creation and management
- Budget vs. actual comparisons
- Year-over-year analysis
- Financial calculations and projections

4. User Service

- User profile management
- Organization and team structure
- Permission management
- Notification preferences

5. Finance Service

- Financial transaction processing
- VAT/tax calculations
- Payment tracking
- Financial reporting engine

6. Document Service

- Document storage and versioning
- Document approval workflows
- OCR for invoice processing
- Template management

7. Supplier Service

- Supplier profile management
- Quote and invoice submission
- Approval workflows
- Supplier ratings and history

8. Analytics Service

- Business intelligence dashboards
- Custom report generation
- Data export functionality
- Historical trend analysis

9. Integration Service

- Third-party integrations (Howler)
- API management for external systems
- ETL processes
- Webhook management

Data Flow Architecture

Key Workflows

1. **Event Creation Flow** `User Service → Event Service → Document Service → Notification`

2. **Budget Planning Flow** `Event Service → Budget Service → Finance Service → Document Service`

3. **Supplier Invoice Processing**
   `Supplier Service → Document Service → Finance Service → Budget Service (for actual vs budget)`

4. **Financial Reporting Flow**
   `Budget Service + Finance Service → Analytics Service → Document Service (for report generation)`

Security Considerations

1. **Data Security**

   - Encryption at rest and in transit
   - PII (Personally Identifiable Information) protection
   - Financial data isolation

2. **Access Control**

   - Granular, role-based permissions
   - Multi-factor authentication for financial operations
   - API key security for integrations

3. **Compliance**
    - Financial audit trails
    - GDPR compliance for European markets
    - Financial reporting regulations

Scalability Design

1. **Horizontal Scaling**
    - Stateless microservices designed for horizontal scaling
    - Database sharding strategy for large clients
    - CDN for global content delivery
2. **Performance Optimization**
    - Cache strategy for financial calculations
    - Asynchronous processing for non-critical operations
    - Database query optimization for financial reporting

Implementation Phases

Phase 1: Core Platform (MVP)

- Authentication and basic user management
- Event and budget creation
- Basic financial reporting
- Document storage

Phase 2: Advanced Features

- Supplier portal and workflow management
- Integration with Howler
- Advanced financial reporting
- Mobile application

Phase 3: Enterprise Features

- White-labeling capabilities
- Advanced analytics
- Additional third-party integrations
- AI-powered budget recommendations

Integration Points

1. **Howler Integration**
    - Ticket sales data import
    - Revenue synchronization
    - Customer information (limited to requirements)
2. **Accounting System Integration**

    - Export to common accounting formats
    - Tax calculation synchronization
    - Financial reconciliation

3. **Payment Gateway Integration**

    - Invoice payment processing
    - Deposit tracking
    - Payment reconciliation

Recommended Development Approach

1. **Agile Methodology**

    - Two-week sprints
    - Regular stakeholder demos
    - Continuous integration/deployment

2. **Development Team Structure**

    - Frontend team (3-4 developers)
    - Backend team (4-5 developers)
    - DevOps engineer
    - UX/UI designer
    - Product manager
    - QA specialists

3. **Testing Strategy**

    - Automated unit and integration tests
    - Financial calculation verification
    - Security penetration testing
    - User acceptance testing with event industry professionals

---

This high-level architectural design provides a framework for building the EventVault platform. The recommended technologies balance modern development practices with the stability requirements of a financial system, while the microservices architecture allows for scalability and future enhancements as the product grows.