Diagrama de Arquitectura - Hacienda API

🏗️ Vista General de la Arquitectura

```
graph TB
    subgraph "Cliente / Frontend"
        UI[Frontend Application]
        POSTMAN[Postman / Testing Tools]
    end

    subgraph "API Gateway"
        ROUTES[Express Routes]
        MW[Middlewares<br/>- CORS<br/>- Logger<br/>- Error Handler]
    end

    subgraph "Controladores"
        FC[FacturaController<br/>Spanish Endpoints]
        EIC[EnglishInvoiceController<br/>English Endpoints]
        HC[Health Controller<br/>System Status]
    end

    subgraph "Validadores"
        FV[FacturaValidator<br/>Joi Schemas ES]
        EIV[EnglishInvoiceValidator<br/>Joi Schemas EN]
        CONV[Language Converter<br/>EN ⇔ ES]
    end

    subgraph "Servicios de Negocio"
        ATV[ATV Adapter<br/>Tax System Integration]
        STORAGE[Invoice Storage<br/>File Management]
        LOGGER[Logger Service<br/>Structured Logging]
    end

    subgraph "Almacenamiento"
        JSON[(JSON Files<br/>Invoice Data)]
        XML[(XML Files<br/>Generated XMLs)]
        LOGS[(Log Files<br/>System Logs)]
    end

    subgraph "Sistema Externo"
        HACIENDA[Ministerio de Hacienda<br/>Costa Rica Tax System]
    end

    %% Flujo de datos
    UI --> ROUTES
    POSTMAN --> ROUTES

    ROUTES --> MW
    MW --> FC
    MW --> EIC
```

```
    MW --> HC

    FC --> FV
    EIC --> EIV
    EIC --> CONV
    CONV --> FV

    FV --> ATV
    EIV --> ATV

    FC --> STORAGE
    EIC --> STORAGE

    ATV --> HACIENDA

    STORAGE --> JSON
    STORAGE --> XML

    LOGGER --> LOGS

    %% Styling
    classDef controller fill:#e1f5fe
    classDef validator fill:#f3e5f5
    classDef service fill:#e8f5e8
    classDef storage fill:#fff3e0
    classDef external fill:#ffebee

    class FC,EIC,HC controller
    class FV,EIV,CONV validator
    class ATV,STORAGE,LOGGER service
    class JSON,XML,LOGS storage
    class HACIENDA external
```

🔄 Flujo de Procesamiento de Facturas

```
sequenceDiagram
    participant C as Cliente
    participant R as Routes
    participant FC as FacturaController
    participant FV as FacturaValidator
    participant ATV as ATV Adapter
    participant S as Storage
    participant H as Hacienda

    Note over C,H: Flujo completo de emisión de factura

    C->>R: POST /api/facturas/emitir
    R->>FC: emitirFactura(req, res)

    FC->>FV: validateFactura(payload)
```

```
    FV-->>FC: Validation Result

    alt Validation Failed
        FC-->>C: Error Response (400)
    else Validation Success
        FC->>ATV: init(config)
        ATV-->>FC: Initialized

        FC->>ATV: emitirComprobante(data)
        ATV->>H: Send to Tax System
        H-->>ATV: Response/Clave
        ATV-->>FC: Invoice Data + Clave

        FC->>S: saveInvoiceJSON(consecutivo, data)
        S-->>FC: JSON Saved

        FC->>S: saveInvoiceXML(consecutivo, xml)
        S-->>FC: XML Saved

        FC->>ATV: enviarComprobante(clave)
        ATV->>H: Send Invoice
        H-->>ATV: Sent Confirmation
        ATV-->>FC: Send Result

        FC->>S: markAsSent(consecutivo, meta)
        S-->>FC: Marked as Sent

        FC-->>C: Success Response (200)
    end
```

🌐 Arquitectura Bilingüe

```
graph LR
    subgraph "Spanish Interface"
        SPA[/api/facturas/*]
        SFC[FacturaController]
        SFV[FacturaValidator]
    end

    subgraph "English Interface"
        ENG[/api/en/invoices/*]
        EIC[EnglishInvoiceController]
        EIV[EnglishInvoiceValidator]
    end

    subgraph "Language Bridge"
        CONV[Language Converter<br/>convertToSpanish()<br/>convertFromSpanish()]
    end

    subgraph "Unified Backend"
```

```
        ATV[ATV Adapter]
        STORAGE[Storage Service]
        HACIENDA[(Tax System<br/>Spanish Only)]
    end

    SPA --> SFC
    SFC --> SFV

    ENG --> EIC
    EIC --> EIV
    EIC --> CONV
    CONV --> SFV

    SFV --> ATV
    SFC --> STORAGE
    EIC --> STORAGE

    ATV --> HACIENDA

    classDef spanish fill:#ffcdd2
    classDef english fill:#c8e6c9
    classDef bridge fill:#fff9c4
    classDef unified fill:#e1f5fe

    class SPA,SFC,SFV spanish
    class ENG,EIC,EIV english
    class CONV bridge
    class ATV,STORAGE,HACIENDA unified
```

🔧 Patrón de Adaptadores

```
graph TB
    subgraph "Application Layer"
        CTRL[Controllers]
        VALID[Validators]
    end

    subgraph "Business Logic Layer"
        subgraph "ATV Adapter Pattern"
            IFACE[IAtvInterface<br/>Abstract Interface]
            REAL[RealAtvAdapter<br/>Production Mode]
            MOCK[MockAtvAdapter<br/>Development Mode]
        end

        STORAGE[Storage Service]
        LOGGER[Logger Service]
    end

    subgraph "Infrastructure Layer"
        ATV_LIB[ATV Library<br/>Costa Rica]
```

```
        FS[File System]
        WINSTON[Winston Logger]
    end

    CTRL --> IFACE
    VALID --> IFACE

    IFACE -.-> REAL
    IFACE -.-> MOCK

    REAL --> ATV_LIB
    MOCK --> MOCK

    STORAGE --> FS
    LOGGER --> WINSTON

    classDef app fill:#e3f2fd
    classDef business fill:#f1f8e9
    classDef infra fill:#fafafa

    class CTRL,VALID app
    class IFACE,REAL,MOCK,STORAGE,LOGGER business
    class ATV_LIB,FS,WINSTON infra
```

📊 Gestión de Estados

```
stateDiagram-v2
    [*] --> Draft: Nueva factura
    Draft --> Validating: Validar datos

    Validating --> ValidationFailed: Error validación
    Validating --> Validated: Validación exitosa

    ValidationFailed --> Draft: Corregir errores

    Validated --> Emitting: Emitir factura

    Emitting --> EmissionFailed: Error emisión
    Emitting --> Emitted: Factura emitida

    EmissionFailed --> Validated: Reintentar

    Emitted --> Sending: Enviar a Hacienda

    Sending --> SendFailed: Error envío
    Sending --> Sent: Enviada exitosamente

    SendFailed --> Emitted: Reintentar envío

    Sent --> Querying: Consultar estado
```

```
    Querying --> Accepted: Aceptada
    Querying --> Rejected: Rechazada
    Querying --> Sent: Procesando

    Accepted --> [*]: Proceso completo
    Rejected --> [*]: Proceso terminado
```

🔐 Capas de Validación

```
graph TD
    INPUT[Input Data]

    subgraph "Validation Layers"
        L1[Layer 1: Schema Validation<br/>Joi Schemas<br/>Structure & Types]
        L2[Layer 2: Business Logic<br/>Custom Rules<br/>Cross-field Validation]
        L3[Layer 3: ATV Validation<br/>Tax System Rules<br/>External Validation]
    end

    subgraph "Error Handling"
        E1[Schema Errors<br/>400 Bad Request]
        E2[Business Errors<br/>422 Unprocessable]
        E3[ATV Errors<br/>502 Bad Gateway]
    end

    OUTPUT[Valid Data]

    INPUT --> L1
    L1 --> L2
    L2 --> L3
    L3 --> OUTPUT

    L1 -.->|Fail| E1
    L2 -.->|Fail| E2
    L3 -.->|Fail| E3

    classDef validation fill:#e8f5e8
    classDef error fill:#ffcdd2
    classDef success fill:#c8e6c9

    class L1,L2,L3 validation
    class E1,E2,E3 error
    class OUTPUT success
```

📁 Estructura de Almacenamiento

```
graph TB
    subgraph "Root Directory"
        ROOT[/facturas_storage/]
```

```
    end

    subgraph "Organized by Date"
        YEAR[/2024/]
        MONTH[/01/, /02/, .../]
    end

    subgraph "File Types per Invoice"
        JSON[invoice_001.json<br/>Original Data]
        XML[invoice_001.xml<br/>Generated XML]
        META[invoice_001_meta.json<br/>Metadata & Status]
    end

    subgraph "Indexes"
        INDEX[invoices_index.json<br/>Quick Lookup]
        STATS[statistics.json<br/>Counters & Metrics]
    end

    ROOT --> YEAR
    YEAR --> MONTH
    MONTH --> JSON
    MONTH --> XML
    MONTH --> META

    ROOT --> INDEX
    ROOT --> STATS

    classDef folder fill:#fff3e0
    classDef file fill:#e1f5fe
    classDef index fill:#f3e5f5

    class ROOT,YEAR,MONTH folder
    class JSON,XML,META file
    class INDEX,STATS index
```

🔄 Patrón de Configuración

```
graph LR
    subgraph "Environment Sources"
        ENV[.env Files]
        ARGS[Process Args]
        DEFAULTS[Default Values]
    end

    subgraph "Config Manager"
        MERGER[Config Merger<br/>Priority Order]
        VALIDATOR[Config Validator<br/>Required Fields]
    end

    subgraph "Application Modules"
```

```
        ATV_CFG[ATV Config]
        STORAGE_CFG[Storage Config]
        LOGGER_CFG[Logger Config]
        SERVER_CFG[Server Config]
    end

    ENV --> MERGER
    ARGS --> MERGER
    DEFAULTS --> MERGER

    MERGER --> VALIDATOR
    VALIDATOR --> ATV_CFG
    VALIDATOR --> STORAGE_CFG
    VALIDATOR --> LOGGER_CFG
    VALIDATOR --> SERVER_CFG

    classDef source fill:#e8f5e8
    classDef process fill:#fff3e0
    classDef target fill:#e1f5fe

    class ENV,ARGS,DEFAULTS source
    class MERGER,VALIDATOR process
    class ATV_CFG,STORAGE_CFG,LOGGER_CFG,SERVER_CFG target
```

📈 Flujo de Monitoreo

```
graph TB
    subgraph "Application Events"
        REQ[HTTP Requests]
        ERRORS[Application Errors]
        ATV_CALLS[ATV Operations]
        STORAGE_OPS[Storage Operations]
    end

    subgraph "Logger Service"
        LOGGER[Winston Logger<br/>Structured Logging]
        FORMATTER[Log Formatter<br/>JSON + Timestamp]
    end

    subgraph "Log Outputs"
        CONSOLE[Console<br/>Development]
        FILES[Log Files<br/>Production]
        COMBINED[combined.log<br/>All Events]
        ERROR_LOG[error.log<br/>Errors Only]
    end

    subgraph "Monitoring"
        STATS[Statistics<br/>Counter & Metrics]
        HEALTH[Health Checks<br/>System Status]
    end
```

```
REQ --> LOGGER
ERRORS --> LOGGER
ATV_CALLS --> LOGGER
STORAGE_OPS --> LOGGER

LOGGER --> FORMATTER

FORMATTER --> CONSOLE
FORMATTER --> FILES
FILES --> COMBINED
FILES --> ERROR_LOG

LOGGER --> STATS
STATS --> HEALTH

classDef event fill:#e3f2fd
classDef process fill:#f1f8e9
classDef output fill:#fff3e0
classDef monitor fill:#f3e5f5

class REQ,ERRORS,ATV_CALLS,STORAGE_OPS event
class LOGGER,FORMATTER process
class CONSOLE,FILES,COMBINED,ERROR_LOG output
class STATS,HEALTH monitor
```

🎯 Puntos Clave de la Arquitectura

1. Separación por Idiomas

- **Rutas separadas**: `/api/facturas/*` (ES) vs `/api/en/invoices/*` (EN)
- **Controladores independientes** pero compartiendo servicios de backend
- **Conversión automática** para compatibilidad con sistema fiscal

2. Patrón Adapter

- **ATV Adapter** abstrae la complejidad del sistema fiscal costarricense
- **Modo REAL/SIMULATED** para desarrollo y producción
- **Interface consistente** independiente del backend

3. Validación en Capas

- **Schema validation** (Joi) para estructura de datos
- **Business logic validation** para reglas de negocio
- **ATV validation** para validación fiscal externa

4. Storage Organizado

- **Estructura temporal** para fácil navegación
- **Múltiples formatos** (JSON, XML, metadata)
- **Índices** para consultas rápidas

5. Logging Estructurado

- **Winston** para logging profesional
- **Diferentes niveles** según ambiente
- **Metadata contextual** para debugging

6. Configuración Flexible

- **Variables de entorno** para diferentes ambientes
- **Valores por defecto** para desarrollo rápido
- **Validación de configuración** requerida

Esta arquitectura soporta escalabilidad, mantenibilidad y testing efectivo del sistema de facturación electrónica.