Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

a

**Arquitectura de Software**

**Obligatorio**

Te Pago Ya

Nicolás Hernández – 198765

Docentes:

Andrés Calviño

Mathias Fonseca

1- Introducción 3

1.1- Propósito 3

2- Antecedentes 3

2.1 - Propósito del sistema 3

2.2 - Requerimientos significativos de Arquitectura 5

2.2.1 - Resumen de requerimientos funcionales 5

2.2.2 - Resumen de requerimientos no funcionales 6

3- Documentación de la arquitectura 7

3.1 – Vista de componentes y conectores general. 8

3.1.1 – Representación primaria 8

3.1.2 – Catalogo de elementos 8

3.1.3 – Decisiones de diseño 9

3.2 – Vista de módulos general. 11

3.2.1 – Representación primaria 11

3.2.2 – Catalogo elementos 11

3.3 – Commerce vista de componentes y conectores. 12

3.3.1 – Representación primaria 12

3.3.2 – Catalogo de elementos 12

3.3.3 – Decisiones de diseño 12

3.4 – Commerce vista de modulos. 12

3.5 – Gateway vista de componentes y conectores. 12

3.6 – Gateway vista de modulos 12

3.7 – Network vista de componentes y conectores. 12

3.8 – Network vista de modulos 12

3.9 – Transmitter vista de componentes y conectores. 12

3.10 – Transmitter vista de modulos 12

3.11 – PayYouNow vista de componentes y conectores. 12

3.12 – PayYouNow vista de modulos 12

3.13 – ErrorHandler vista de componentes y conectores. 12

3.14 – ErrorHandler vista de modulos 12

3.15 – Authentication vista de componentes y conectores. 12

3.16 – Authentication vista de modulos 12

# Introducción

A continuación en el siguiente documento se describirán las decisiones técnicas, de diseño y decisiones de arquitectura tomadas para el desarrollo del sistema *Te Pago Ya,* mostrándolas, explicándolas y proveyendo también una justificación de porque fueron tomadas a cabo y como las mismas llevaron a la solución actual del problema. Para esto se hará un análisis exhaustivo del sistema anterior y los requerimientos solicitados e identificados por el equipo de desarrollo.

## Propósito

El propósito de este documento entonces será de proveer una especificación para el sistema de pagos *Te Pago Ya*

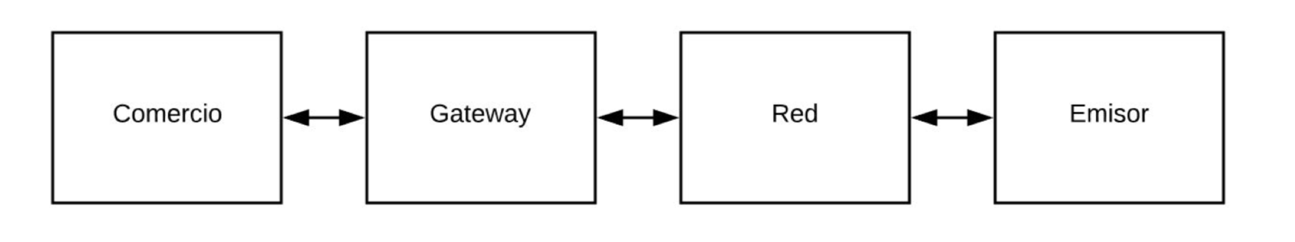
# Antecedentes

## 2.1 - Propósito del sistema

Te Pago Ya es un sistema de pagos el cual viene a tratar de resolver el complicado universo de los pagos online, surge como solución a la urgente necesidad de simplificar lo mas posible la integración de los diferentes actores que se encuentran en este universo ( comercios, gateways, emisores de tarjetas, etc.)

Te Pago Ya entonces será una herramienta que actuara como intermediario, resolviendo la mayor parte de los escenarios de integración como pueden ser los diferentes formatos de fechas y números, formatos de representación de datos, etc.

Actualmente las comunicaciones entre los diferentes actores funciona de forma directa, el cliente proveyó el siguiente diagrama a modo de ejemplo:



El emisor es el encargado de emitir las tarjetas de crédito

Cuando un comercio quiere realizar ventas con las tarjetas este se comunica con el Gateway, estos a su vez necesitan conectarse con la Red la cual hace control de fraudes, etc y este se comunica con el Emisor para realizar los últimos controles.

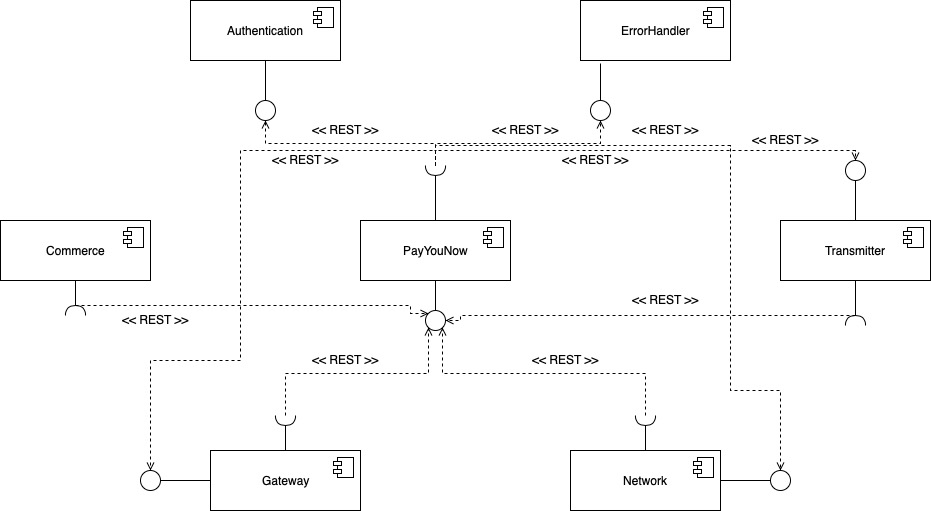
Al finalizar el Emisor los controles, este responde a la Red, la cual responde al Gateway y el Gateway al comercio.

Como se puede observar el escenario es bastante complejo y si además se añade que existen muy pocos estándares y reglas de cómo deben comunicarse estos actores se hace muy complicado manejar la integración y mantenibilidad de los sistemas.

Luego de analizar todo este escenario surgió la posibilidad de *Te Pago Ya* que viene a aparecer como una solución en el complicado universo de los pagos online.

Este se encargara de las comunicaciones entre los sistemas. También se podrán registrar nuevos sistemas de forma dinámica solamente proveyendo una interface para poder acceder a sus funcionalidades.

Se plantea como posible solución el siguiente modelo de conexiones entre los distintos módulos del sistema:



Como se puede observar, *Te Pago Ya* actuara como intermediario entre las comunicaciones. Para obtener este modelo fue puesto en consideración el patrón SOA. *Te Pago Ya* permitirá registrar nuevos servicios por medio de un endpoint REST, lo único necesario para hacerlo es enviarle toda la información del servicio nuevo a registrarse, es necesario especificar nombre, tipo de conexión, dirección de localización y las operaciones que expondrá.

Te Pago Ya dará la posibilidad también consumir los servicios especificados siempre y cuando se cumplan determinadas condiciones como son que:

* Se especifique el nombre del proveedor
* Se especifique el nombre de la operación y los parámetros y argumentos necesarios.
* El consumidor se encuentre autenticado debidamente y cuente con los permisos necesarios para realizar la operación.

A partir de esto respetando la correcta aplicación del patrón SOA Te Pago Ya validara, enriquecerá y transmitirá la información que pase atreves de él, usando además mecanismos de soporte como son la autenticación y el manejo de errores.

## 2.2 - Requerimientos significativos de Arquitectura

Como ya mencionamos anteriormente, los actores de este sistema son Comerce, PayYouNow, Network, Transmitter, Gateway, ErrorHandler y Authentication.

### 2.2.1 - Resumen de requerimientos funcionales

| Id Requerimiento | Descripción | Actor |
| --- | --- | --- |
| REQ-1 | Compra en comercio, un consumidor podrá realizar una transacción de compra en un comercio, este en función de la categoría del producto a comprar deberá seleccionar el Gateway que procesara la compra y enviarle la solicitud | Commerce, PayYouNow, Authentication, ErrorHandler |
| REQ-2 | Compra en Gateway, el Gateway procesa el pago y decide a que red enviarlo dependiendo de la tarjeta. Se debe realizar en tiempos menores a 10ms bajo cargas de 1000 solicitudes por minuto. | Gateway, PayYouNow, Authentication, ErrorHandler |
| REQ-3 | Compra en red, realiza controles de prevención de fraudes contra un limite determinado y envía la transaccion hacia el emisor para el paso final. Se debe poder alterar el limite para el control de fraude en tiempo de ejecución. Se debe realizar en tiempos menores a 10ms bajo cargas de 1000 solicitudes por minuto. | Network, PayYouNow, Authentication, ErrorHandler |
| REQ-4 | Compra en emisor, debe chequear que la tarjeta sea valida (que no este vencida, ni bloqueada, ni denunciada), ejecutar el algoritmo de luhn y que tenga saldo suficiente para realizar la transacción. | Transmitter, PayYouNow, Authentication, ErrorHandler |
| REQ-5 | Devolución, deberá ser posible la devolución de cualquier transacción que no haya superado el limite de días definido en el emisor. | Commerce, Network, Gateway, Transmitter, PayYouNow, Authentication, ErrorHandler |
| REQ-6 | El usuario puede realizar un desconocimiento de compra en su emisor. El emisor debe informar al comercio para que realice los tramites administrativos. | Commerce,  Transmitter, PayYouNow, Authentication, ErrorHandler |
| REQ-7 | El comercio le solicita diariamente al Gateway la transacción de cierre de lotes en donde se informa al máximo detalle los movimientos de dinero. Debe responder en un tiempo promedio menor a 50ms bajo cargas de 5000 solicitudes por minuto | Commerce, Gateway, PayYouNow, , Authentication, ErrorHandler |

### 2.2.2 - Resumen de requerimientos no funcionales

| Id Requerimiento | Nombre | Atributo de Calidad | Descripcion |
| --- | --- | --- | --- |
| RNF01 | Modificación de las distintas aplicaciones | Modificabilidad | Una aplicación debe poder ser modificada en un futuro sin afectar al resto |
| RNF02 | Integración de aplicaciones | Interoperabilidad | Desarrollar TePagoYa/PayYouNow para resolver la integración e interoperabilidad. Para cada prestador de servicios se debe poder:   * Registrar una interfaz de servicio para acceder a la funcionalidad. * Especificar propiedades sobra la interfaz. * Localizar la interfaz de un servicio registrado. * Facilitar la invocación de las funcionalidades |
| RNF03 | Gestión de fallas y errores | Testeabilidad | El sistema debe proveer suficiente información que permita conocer el detalle de las tareas que se realizan. |
| RNF04 | Recuperación de fallas | Disponibilidad | En caso de ocurrir una falla o errores es imprescindible que el sistema provea toda la información. |
| RNF05 | Autenticación | Seguridad | Autenticación, para toda aplicación prestado de servicios que se registre en TePagoYa/PayYouNow se requiere que todas sus operaciones publicadas sean invocadas únicamente a través de TePagoYa/PayYouNow |
| RNF06 | Estándares de seguridad | Seguridad | Toda aplicación debe cumplir con los estándares de seguridad de PCI al máximo nivel posible, pudiendo demostrar y justificar a cada momento el nivel de cumplimiento. |
| RNF07 | Tiempo de respuesta | Performance | Se debe minimizar el tiempo de respuesta de las peticiones entre aplicaciones |

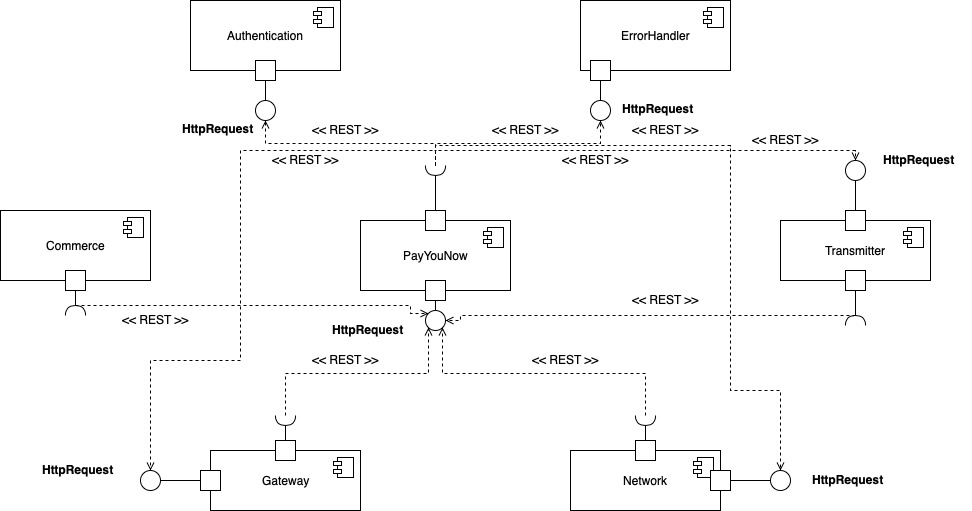
# Documentación de la arquitectura

En esta sección se presentara la arquitectura del sistema desarrollado como solución a la problemática descripta en las secciones anteriores. Esta representación se llevara a cabo mediante la utilización de diferentes vistas del sistemas, las cuales se irán detallando y explicando de manera de poder demostrar los requerimientos que fueron implementados en el sistemas y los atributos de calidad que los mismos contemplan.

Comenzaremos presentando el sistema en general mediante una vista de componentes y conectores, esta vista nos permitirá demostrar el funcionamiento del sistema desarrollador y dar una base de cómo esta compuesto el mismo. Luego de haber entendido el funcionamiento general del sistema se pasara el foco a los módulos del mismo, para lograr darle el detalle y la dimensión adecuada se utilizara la vista de módulos la cual nos ayudara a mostrar las decisiones de diseño que se fueron tomando en cada uno. Finalizaremos este apartado presentando una vista de asignación para demostrar como se distribuye el sistema en general.

## 3.1 – Vista de componentes y conectores general.

### 3.1.1 – Representación primaria



### 3.1.2 – Catalogo de elementos

| Elemento | Tipo | Responsabilidades |
| --- | --- | --- |
| Authentication | Express Server | Es el encargado de manejar la autenticación de los modulos, registra los modulos, provee tokens y los valida. |
| PayYouNow | Express Server | Es el encargado de la interoperabilidad entre todos los modulos y de las comunicaciones del mismo expone servicios REST en las cuales se pueden registrar servicios o consumir servicios registrados. |
| Commerce | Express Server | Es el encargado de iniciar el proceso de transacción, determina el Gateway con el cual se procesara la compra. |
| Gateway | Express Server | Es el encargado de determinar la Red a partir de la tarjeta con la que se realiza la compra, expone servicios REST para el mismo. |
| Transmitter | Express Server | Es el encargado de validar los últimos pasos de la transacción, que la tarjeta sea valida, que no este bloqueada, que el saldo sea suficiente, además también se encarga de emitir tarjetas. Expone servicios REST para las operaciones nombradas. |
| Network | Express Server | Es el encargado de llevar a cabo un control de fraude en la tarjeta de la transacción y también de derivar la misma al emisor correspondiente, también lo hace mediante un servicio REST. |
| ErrorHandler | Express Server | Encargado de llevar una trazabilidad de las operaciones realizadas, también provee información en caso de defectos y fallas. Expone un servicio REST que usa un servicio en la WEB para control de fallas. |

### 3.1.3 – Decisiones de diseño

Se opto por usar como mecanismo de comunicación REST y a partir de este usar los beneficios del mismo como lo es la comunicación por JSON. Gracias a esta decisión ya logramos desacoplar la implementación de los diferentes módulos con sus operaciones provistas, ya que nada mas para acceder a la misma solo necesitamos la uri indicada y los parámetros requeridos, además JSON también nos aporta otra capa mas de desacople donde no juega realmente la tecnología que este detrás de la interfaz REST.

El modulo PayYouNow esta capacitado para registrar aplicaciones que pueden ser prestadores de servicios así como también consumidoras de los mismos los cuales se van a comunicar con el en run-time. Para llevar a las operaciones descriptas como pueden ser el registro así como también el consumo de algunas operaciones particulares expone un servicio REST, dicho servicio mapea las peticiones hacia los controladores los cuales se encargan de determinar cual es la operación solicitada y llevarla a cabo. Gracias a REST logramos también conseguir una cierta capacidad de carga importante, pudiendo soportar una gran cantidad de peticiones por minuto, siendo algo crucial en el universo de las compras online.

El modulo ErrorHandler se llevo a cabo debido a la necesidad de manejar un cierto mecanismo de trazabilidad de las operaciones llevadas a cabo y también de tener información que sea de utilidad en caso de ocurrir defectos o fallas y lograr aumentar la disponibilidad así como también la testeabilidad.

La decisión de hacer un modulo totalmente separado para esta función surge de lograr un desacoplamiento total de los demás y que los módulos no tuvieran que encargarse cada uno de esta función sino que pudieran delegarla en otro diferente, este modulo solo expone un servicio REST el cual tiene dos POST, uno para loggers y el otro para errores, de esta manera cualquier modulo puede consumir su servicio y hacer uso de el, logrando así también una extensibilidad en la solución ya que si en algún caso se llega a agregar algún actor a la solución, este puede usar fácilmente el servicio REST de manejo de errores.

El modulo Authentication se llevo a cabo para manejar la necesidad de tener ciertas credenciales entre los diferentes módulos, dar un toque de seguridad y proteger a PayYouNow de consumo de servicios indeseados así como también de posibles ataques. Para esto no quedaba otra solución mejor que sacar esta funcionalidad a una entidad superior y apartada de los diferentes módulos, igual que en el caso anterior esto nos da una gran extensibilidad ya que se pueden registrar y autenticar una cantidad virtualmente infinita de módulos.

Este modulo lleva a cabo la tarea de autenticación mediante el uso de JsonWebToken los cuales son un método de autenticación sencillo, seguro y además totalmente desacoplado pudiendo ser utilizado por módulos de todo tipo. Expone un servicio REST con las operaciones de Registrar, Autenticar y Validar.

Como extra podemos añadir que el mismo maneja encriptación en las contraseñas guardadas en la base de datos.

Tambien se quiere añadir que debido a la complejidad y la falta de tiempo no se llevo a cabo la implementación de la política de token de refresco ya que la misma no es tan trivial, sin embargo puede ser fácilmente implementable debido a que JsonWebToken ya provee esta funcionalidad siendo fácilmente extensible la solución en futuras implementaciones.

Los módulos Commerce, Gateway, Transmitter y Network son módulos los cuales funcionan al estilo de “Mocks” ya que en la realidad el equipo no va a ser el encargado de desarrollar estos diferentes módulos. Sin embargo para esta solución se opto por la decisión de que por mas que fueran mocks lograr que esto fuera lo mas parecido a la realidad. La comunicación entre estos 4 actores y PayYouNow se decisión que fuera REST debido a la naturaleza de la misma, que ya fue descripta en apartados superiores, usando esto se logro desacoplar totalmente los 5 actores fundamentales de la solución y lograr una comunicación sencilla, eficaz y fácilmente modificable.

Al principio en el primer reléase como se puede ver en el repositorio de la solución la comunicación entre los módulos era diferente pensando mas que nada en temas de performance pero sacrificando mucha modificabilidad, la misma la hacia en su totalidad PayYouNow utilizando las respuestas, lo cual eliminaba muchas peticiones en la red y como esta es un recurso caro nos daba un salto interesante en performance. Sin embargo después de tener una charla con el cliente el mismo prefirió sacrificar un poco de performance y ganar en modificabilidad dejando la solución actual de hoy en dia.

A continuación presentaremos un diagrama de secuencia para describir la comunicación hoy en dia

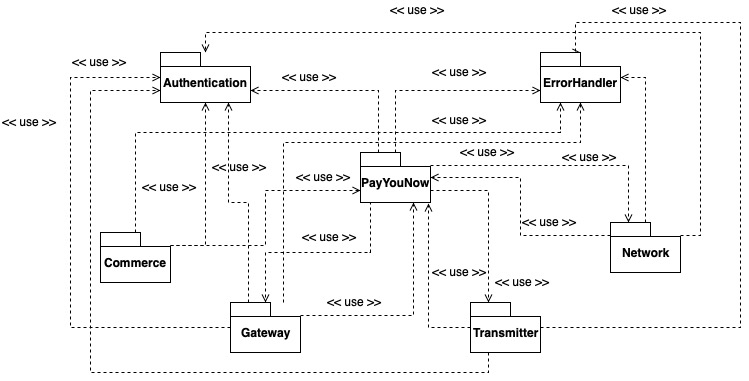
Como extra general podemos decir que todos los módulos cuentan archivos de configuración los cuales pueden ser fácilmente cambiados dándoles a los módulos un extra en modificabilidad.

Tambien en futuras instancias de desarrollo de PayYouNow se le agregaría una queue para manejar las solicitudes REST que recibe con el fin de no bloquear a sus consumidores esperando por el servicio. Lamentablemente, por asuntos de tiempo, esto no llegó a ser implementado para la presente iteración, pero sería una mejora a implementar a futuro.

Debido a la falta de tiempo de los integrantes del equipo, no se llegaron a implementar muchas de las ideas para el sistema. Es por esto que se hará especial hincapié en describirlas en este documento, para plasmar las implementaciones a futuro que no fueron posibles ejecutar.

## 3.2 – Vista de módulos general.

### 3.2.1 – Representación primaria



### 3.2.2 – Catalogo elementos

| Elemento | Tipo | Responsabilidades |
| --- | --- | --- |
| Authentication | Express Server | Es el encargado de manejar la autenticación de los módulos, registra los módulos, provee tokens y los valida. |
| PayYouNow | Express Server | Es el encargado de la interoperabilidad entre todos los módulos y de las comunicaciones del mismo expone servicios REST en las cuales se pueden registrar servicios o consumir servicios registrados. |
| Commerce | Express Server | Es el encargado de iniciar el proceso de transacción, determina el Gateway con el cual se procesara la compra. |
| Gateway | Express Server | Es el encargado de determinar la Red a partir de la tarjeta con la que se realiza la compra, expone servicios REST para el mismo. |
| Transmitter | Express Server | Es el encargado de validar los últimos pasos de la transacción, que la tarjeta sea valida, que no este bloqueada, que el saldo sea suficiente, además también se encarga de emitir tarjetas. Expone servicios REST para las operaciones nombradas. |
| Network | Express Server | Es el encargado de llevar a cabo un control de fraude en la tarjeta de la transacción y también de derivar la misma al emisor correspondiente, también lo hace mediante un servicio REST. |
| ErrorHandler | Express Server | Encargado de llevar una trazabilidad de las operaciones realizadas, también provee información en caso de defectos y fallas. Expone un servicio REST que usa un servicio en la WEB para control de fallas. |

## 3.3 – Commerce vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.4 – Commerce vista de modulos.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.5 – Gateway vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.6 – Gateway vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.7 – Network vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.8 – Network vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.9 – Transmitter vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.10 – Transmitter vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.11 – PayYouNow vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.12 – PayYouNow vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.13 – ErrorHandler vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.14 – ErrorHandler vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.15 – Authentication vista de componentes y conectores.

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño

## 3.16 – Authentication vista de modulos

### 3.3.1 – Representación primaria

### 3.3.2 – Catalogo de elementos

### 3.3.3 – Decisiones de diseño