Ingeniería de Software II

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Trabajo Práctico 2: Segunda Parte

Primer Cuatrimestre de 2012

Pablo Antonio	290/08	pabloa@gmail.com
Rodrigo Campos	561/06	rodrigo@sdfg.com.ar
Ayelén Chavez	130/06	ashy.on.line@gmail.com
Pablo Herrero	332/07	pablodherrero@gmail.com

Instancia	**Corrector**	**Nota**
Entrega		

Comentarios del corrector:

Escenarios

Modificabilidad

Primer escenario:

Breve descripción: el Ministro de Producción dice que tiene que funcionar para todo: aviones, barcos, lanchas, taxis, todo.

Fuente: ministerio de producción.

Estímulo: se quiere agregar un nuevo medio de transporte.

Artefacto: el sistema.

Entorno: tiempo de diseño.

Respuesta: se agrega un nuevo transporte al sistema sin efectos secundarios.

Medición: menos de 20 horas hombre.

Segundo escenario:

Breve descripción: el representante de Defensa del Consumidor dice querer flexibilidad para personalizar los viajes: seleccionar medios de transporte, escalas, música, temas de conversación, etc.

Fuente: desarrollador.

Estímulo: se quiere agregar una nueva preferencia para los viajes.

Artefacto: el sistema.

Entorno: tiempo de diseño.

Respuesta: se agrega una ueva preferencia al sistema sin efectos secundarios.

Medición: menos de 20 horas hombre.

Tercer escenario:

Breve descripción: el Encargado Área de Desarrollo Técnico dice que quisiera poder ir probando distintas estrategias de matching.

Fuente: encargado del área de desarrollo técnico.

Estímulo: se quiere probar una estrategia de matching.

Artefacto: módulo de estrategia de matching.

Entorno: tiempo de diseño.

Respuesta: se puede probar una estrategia de matching distinta.

Medición: menos de 24 horas hombre.

Performance:

El Ministerio de producción dice que el sistema tiene que andar rápido. Dado que esta descripción es muy general, se plantearon los siguientes escenarios.

Primer escenario:

Breve descripción: el Ministro de Producción dice que el sistema se integre con los sistemas de vialidad nacional, del servicio meteorológico, etc. Todo debe funcionar perfecto y rápido.

Fuente: externo.

Estímulo: quiere usar algún servicio del sistema que dependa de un sistema externo (vialidad nacional, servicio meteorológico, etc).

Artefacto: el sistema.

Entorno: normal, on line.

Respuesta: el usuario externo recibe respuesta del sistema satisfactoriamente.

Medición: en a los sumo 10 milisegundos extras de lo que tarda el servicio del sistema externo.

Segundo escenario:

Breve descripción: el Encargado Área de Desarrollo Técnico dice estar preocupado porque los tiempos que puede llevar lograr armar los viajes (performance) con tanta flexiblidad y posibilidad de cambios.

Fuente: interna.

Estímulo: se recibe la orden de armar viajes.

Artefacto: módulo organizador de viajes.

Entorno: normal, on line.

Respuesta: los viajes son armados.

Medición: en menos de 1 segundo el sistema armará al menos 100 viajes.

Disponibilidad:

Primer escenario:

Breve descripción: el Ministro de Producción dice que es importante que el sistema de la policía funcione perfectamente.

Fuente: interna.

Estímulo: el sistema de la policía federal no responde.

Artefacto: canal de comunicación con el sistema de policía federal.

Entorno: normal, on line.

Respuesta: el sistema sigue funcionando para los usuarios ya autenticados y se notificará a los que intenten autenticarse, que lo intenten más tarde.

Medición: el 99,9999% de las veces.

Segundo escenario:

Breve descripción: el Encargado Área de Desarrollo Técnico dice estar preocupado por la integración con sistemas externos que no está seguro funcionen adecuadamente (servicio meteorólogico, vialidad, etc).

Fuente: externa

Estímulo: se detecta la falta de respuesta por parte del servicio externo.

Artefacto: módulo verificador de disponibilidad de servicios.

Entorno: normal, on line.

Respuesta: el sistema pasa a modo degradado informando al usuario al respecto.

Medición: el 99,9999% de las veces.

Seguridad:

El Ministro de Producción dice que quiere que el sistema sea seguro, pero dado que esta descripción es muy general, se plantearon los siguientes escenarios.

Primer escenario:

Breve descripción: el representante de Defensa del Consumidor dijo que espera se garantice la seguridad de las personas involucradas.

Fuente: persona no autenticada.

Estímulo: intenta autenticarse en el sistema con datos ilegítimos.

Artefacto: módulo de autenticación del sistema.

Entorno: normal, on line.

Respuesta: se crea una nueva entrada en un log del sistema con los datos del intento fallido y el sistema sigue funcionando para los usuarios autenticados.

Medición: el 99,99% de las veces, según asegura el servicio de policía federal.

Segundo escenario:

Breve descripción: el ViceMinistro de Economía dijo que se necesita que el sistema proteja los datos de los usuarios.

Fuente: atacante externo.

Estímulo: se captura un mensaje encriptado e intenta vulnerar la seguridad del mismo para obtener información de los usuarios.

Artefacto: módulo de encriptación del sistema.

Entorno: normal, on line.

Respuesta: los datos no son accesibles al atacante en tiempo razonable (miles de años).

Medición: el 99,9999% de los casos.

Tercero escenario:

Breve descripción: el Representante de Cámara del Transporte exige poder auditar los viajes planeados para poder controlar que los viajes se reparten entre las empresas de manera equitativa.

Fuente: atacante externo

Estímulo: quiere targiversar el log encriptado de viajes planeados por el sistema.

Artefacto: módulo de encriptación del sistema.

Entorno: normal.

Respuesta: los datos no son accesibles al atacante en tiempo razonable (miles de años).

Medición: el 99,9999% de los casos.

Integrabilidad:

Primer escenario:

Breve descripción: el Ministro de Producción quiere que se pueda utilizar información de las redes sociales.

Fuente: un desarrollador.

Estímulo: quiere integrar un servicio externo que provee información de las redes sociales al sistema.

Artefacto: el sistema.

Entorno: tiempo de desarrollo.

Respuesta: se puede integrar el servicio sin afectar al resto del sistema.

Medición: a lo sumo, 40 horas hombre.

Portabilidad:

Primer escenario:

Breve descripción: el sistema debe correr sin problema en dispositivos móviles con distintas tecnologías

Fuente: Secretaría de Comunicaciones

Estímulo: quiere que el sistema corra sin problemas en un dispositivo móvil con una tecnologías aún no soportada.

Artefacto: el sistema.

Entorno: tiempo de desarrollo.

Respuesta: el sistema podrá migrarse a dicha tecnología.

Medición: a lo sumo, 240 hs hombre.

Usabilidad:

Primer escenario:

Breve descripción: debe ser fácil para los usuarios el realizar un pedido de viaje.

Fuente: usuario autenticado del sistema que ingresa por primera vez.

Estímulo: quiere hacer un pedido de viaje.

Artefacto: el sistema.

Entorno: normal, on line.

Respuesta: puede realizar satisfactoriamente el pedido de viaje.

Medición: menos de 2 minutos.

Segundo escenario:

Breve descripción: debe ser fácil para los usuarios el realizar un ofrecimiento de viaje.

Fuente: usuario autenticado del sistema que ingresa por primera vez.

Estímulo: quiere hacer ofrecer un viaje.

Artefacto: el sistema.

Entorno: normal, on line.

Respuesta: puede realizar satisfactoriamente el ofrecimiento de viaje.

Medición: menos de 2 minutos.

Prioridades relativas

Luego de la votación de los distintos *stakeholders* en el QAW, se ordenó según la prioridad relativa a los distintos atributos de calidad.

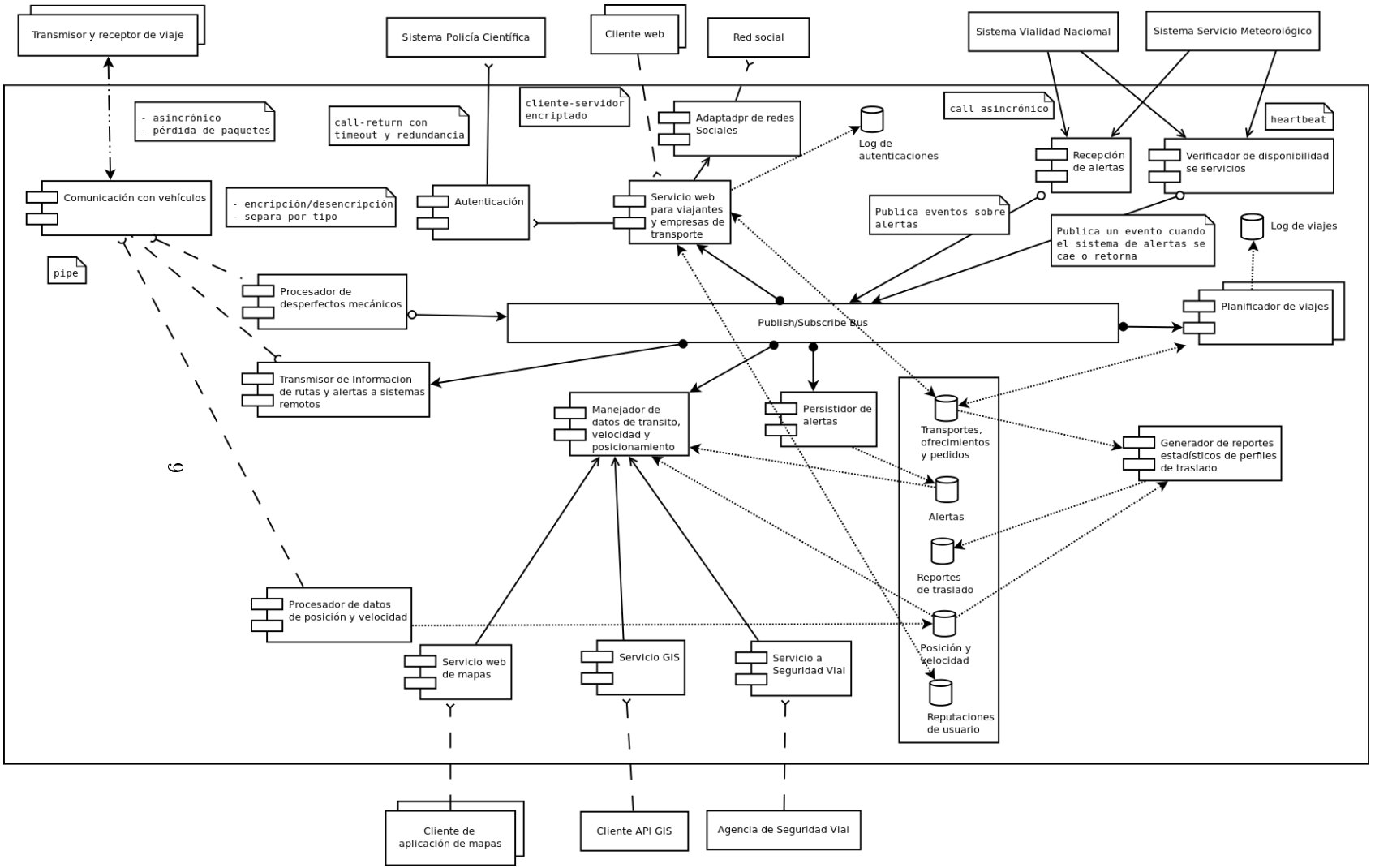
1. Flexibilidad-Modificabilidad
2. *a*) Performance
   b) Disponibilidad
3. Seguridad
4. Integrabilidad
5. *a*) Portabilidad

b) Usabilidad

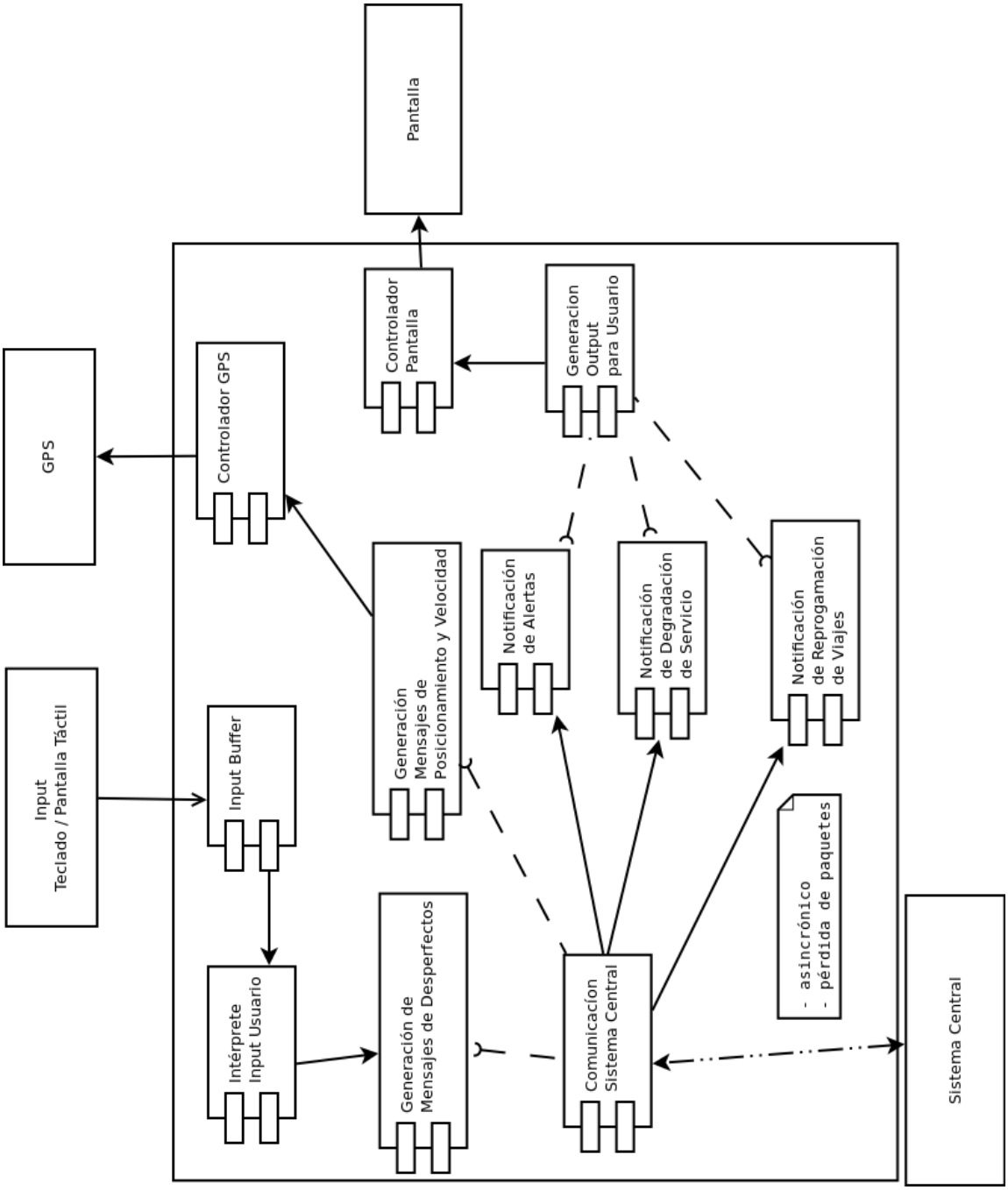
Este orden se refleja en la arquitectura elegida. Es decir, a atributos como Modificabilidad, se le dió más importancia que a atributos como Portabilidad.

Vista de la arquitectura

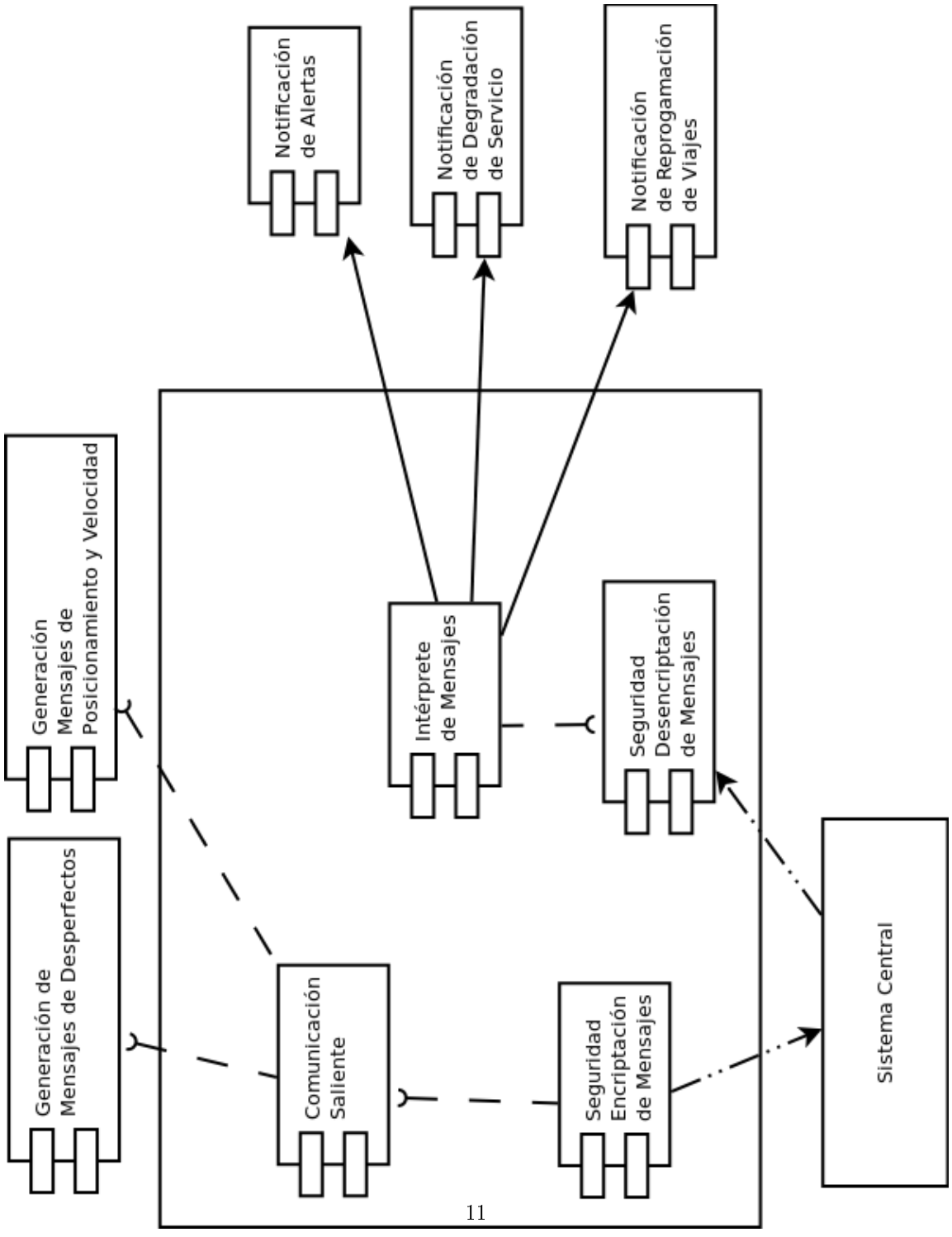
Vista general


Transmisor y receptor de viaje
Sistema Policía Científica
Cliente web
Red social
Sistema Vialidad Nacional
Sistema Servicio Meteorológico
- asincrónico
- pérdida de paquetes
call-return con timeout y redundancia
cliente-servidor encriptado
Adaptadpr de redes Sociales
Log de autenticaciones
call asincrónico
heartbeat
Recepción de alertas
Verificador de disponibilidad se servicios
Comunicación con vehículos
- encripción/desencripción
- separa por tipo
Autenticación
Servicio web para viajantes y empresas de transporte
Publica eventos sobre alertas
Publica un evento cuando el sistema de alertas se cae o retorna
Log de viajes
pipe
Procesador de desperfectos mecánicos
Publish/Subscribe Bus
Planificador de viajes
Transmisor de Informacion de rutas y alertas a sistemas remotos
Manejador de datos de transito, velocidad y posicionamiento
Persistidor de alertas
Transportes, ofrecimientos y pedidos
Generador de reportes estadísticos de perfiles de traslado
Alertas
Reportes de traslado
Procesador de datos de posición y velocidad
Posición y velocidad
Servicio web de mapas
Servicio GIS
Servicio a Seguridad Vial
Reputaciones de usuario
Cliente de aplicación de mapas
Cliente API GIS
Agencia de Seguridad Vial

Transmisor y receptor de viaje



Comunicación sistema central



Decisiones de arquitectura

Todas las decisiones de arquitectura tomadas para este proyecto fueron fuertemente influenciadas por los requerimientos impuestos por las funcionalidades descriptas en los casos de uso, por la evaluación de los riesgos realizada y por los atributos de calidad identificados, junto con sus prioridades relativas.

Es importante aclarar que en el diagrama nos referimos con "Red Social" a cualquier red social (ya sea facebook, twitter, etc.) y el "Adaptador de redes sociales" es el que sabrá conectarse a las redes sociales que se le quiere dar soporte. También, con "Transmisor y receptor de viaje" nos referimos a la parte del sistema que corre en el vehiculo (posiblemente dispositivo movil) del usuario.

A continuación se describen las principales decisiones de arquitectura tomadas.

Utilización de un Publish/Subscribe Bus

Al usar un Bus Publish/Subscribe se logra que cada componente se suscriba a los eventos que le interesan sin tener acoplamiento con quienes lo generan, dado que muchos de estos pueden ser generados por más de un componente y de interés para más de un componente también. Esto resulta particularmente útil y fácilmente extensible para eventos como el de una alerta meteorologica, que es de interés tanto como para el planificador de viajes y para los servicios web de mapas como para el Usuario del sistema con el "Transmisor y receptor de viaje". Al usar un mecanismo Publish/Subscribe, esta alerta puede ser reportada y se suscriben solo a los que les interesa. También da flexibilidad para que esta alerta sea generada por otro componente, si el sistema del servicio meteorológico no resulta confiable y se desea tener un "fallback". Analogamente, este razonamiento tambien aplica para otros mensajes que se transmiten por el Bus. Por ejemplo un desperfecto mecanico que es de interes para los "Servicios web para viajantes y empresas de transporte" tanto como para el planificador de viajes.

Paralelizacion del planificador de viajes

Se decidio paralelizar el planificador de viajes para poder satisfacer los requerimientos de performance que preocupaban al Encargado del área de desarrolo técnico. Esto se vio simplificado por el hecho de que se use un Bus Publish/Subscribe para notificarlo de alertas, ya que todas las instancias del mismo se suscriben a lo que les es pertinente.

Utilización de pipe en las comunicaciones con el módulo de comunicación con vehículos

El uso del pipe permite que los componentes implicados en la comunicación puedan trabajar de manera concurrente. De este modo, el módulo de comunicación con vehículos que, por su naturaleza, puede trabajar a una velocidad distinta, no afectará al desarrollo normal del sistema.

Mecanismo de heartbeat para verificación de disponibilidad de servicios

Se agregó un mecanismo de heartbeat con servicios externos para poder avisar que el sistema está funcionando en modo degradado sin la funcionalidad provista por estos sistemas.

Almacenamiento de autenticaciones infructuosas

Para satisfacer los escenarios motivados por Defensa del Consumidor, se decidió almacenar las autenticaciones infructuosas.

Almacenamiento de viajes planificados

Se decidio almacenar los viajes planificados para que pueda ser auditado luego para verificar que no se realiza ninguna asignación tendenciosa con ninguna compañía.

Almacenamiento de alertas

Se decidio almacenar las alertas reportadas para que sean accesibles en forma historica para el "Servicio GIS", el "Servicio a Seguridad Vial" y el "Servicio web de mapas".

Conectores especiales

- **Call-return con timeout y redundancia para comunicación con Policía Científica:** para poder identificar cuando el sistema de Policia Cientifica no responde se utiliza un timeout. De esta forma, si el sistema no responde en como máximo este tiempo, se asume que no funciona y se reporta que el sistema no podrá autenticar nuevos usuarios. También, para evitar un punto único de falla en la comunicación con Policía Científica, se utiliza un canal que será redundante.

- **Cliente-servidor encriptado para comunicación con clientes web:** para proteger la información de los usuarios se utilizará una conexión encriptada con ellos, de modo de que no pueda ser comprometida la información utilizada en la comunicación.

- **Asincrónico, con pérdida de paquetes para comunicación con transmisores/receptores:** La comunicación con los vehículos consiste en un flujo de paquetes que son enviados en forma asincrónica. Cada paquete contiene toda la información necesaria para el mensaje que quiere enviarse, por lo que no es necesario reconstruirlo cuando este llega ni comprobar si algún fragmento del mismo se perdió. Si bien algunos mensajes pueden perderse, en la moayoría de los casos, este mensaje perdido puede ignorarse o de ser necesario el problema puede ser resuelto a nivel aplicación y no en el protocolo.

- . El protocolo utilizado en capa de usuario puede saldar el problema de la pérdida de paquetes de ser necesario.

UP vs Ágil

A lo largo de este cuatrimestre, pudimos trabajar con ambas metodologías de desarrollo.

Cada integrante del equipo tuvo distintas experiencias laborales. Algunos usaban alguna metodología con el agregado de algunos cambios para adaptarlas a sus necesidades y otros solo usaban una lista de tareas a realizar intentando tener una mínima organización de lo que se esperaba terminar luego de cierto tiempo de desarrollo. Con esa experiencia y con la experiencia ganada durante la cursada, queda bien claro que es importante usar alguna de estas metodologías cuando se cuenta con un proyecto de equipo que requiere más de pocas semanas de desarrollo.

Encontramos ventajas y desventajas en cada metodología de desarrollo de software, pero pudimos encontrar gran ayuda en el uso de alguna metodología a la hora de empezar a desarrollar, por lo que promovemos el uso de alguna de estas metodologías.

Ventajas y desventajas que notamos en UP

1. Es fácil pensar que la documentación es algo superfluo en la que se pierde mucho tiempo en documentos para que nadie los lea. Ambas metodologías tienen mucha documentación, pero sobretodo UP.

2. En el caso de UP, hay que capacitar a los stakeholders para que entiendan cómo es el proceso y por qué es tan importante el trabajo previo a empezar a desarrollar.

3. Una ventaja de UP es que se muestra a los stakeholders un panorama completo de lo que se va a desarrollar, haciendo que intervengan en las decisiones para el desarrollo del proyecto (QAW, por ejemplo) y mostrandoles de distintas maneras cómo sus pedidos son tenidos en cuenta. Sin embargo, hay que tener en mente que el proceso debe agregar valor a los stakeholders del proyecto, sin terminar usando de más la metodología.

4. Utilizamos UP cuando el proyecto a desarrollar es grande, aunque esto puede terminar en la necesidad de tener un experto en UP para poder seguir bien la metodología y asegurar que el proceso esté definido y usado correctamente.

5. En particular, en este trabajo, notamos que el proceso previo a desarrollar es mucho más largo que para Ágil.

6. Intuitivamente, pareciera que UP se centra en pensar y analizar mucho más a futuro el software a desarrollar, teniendo en cuenta cada detalle del proyecto, como ser riesgos, atributos pedidos

por cada stakeholder, tácticas de cómo cubrir esos pedidos, etc. Además, notamos que uno tiene que conocer y analizar mejor el dominio del problema en etapas previas al desarrollo del sistema. Esta característica puede tener aspectos muy buenos, pero puede no funcionar con ciertos proyectos que cambian muy seguido durante su desarrollo.

7. Se establece una base sólida de arquitectura, se la prueba y se la documenta. UP se centra en la arquitectura.

Ventajas y desventajas que notamos en Ágil

1. Los stakeholders y grupo de desarrollo tienen mucha interacción, comunicación y colaboración durante todo el proceso de desarrollo.
2. Al final de cada iteración tenemos un software andando con no mucho esfuerzo y listo para mostrárselo a los stakeholders, como nos pasó en este trabajo.
3. Utilizamos Ágil cuando el proyecto a desarrollar es chico o mediano. Una vez que el proyecto se hace más grande o los integrantes se separan geográficamente, la metodología comienza a fallar.
4. Es una metodología muy sensilla, fácil de aprender y de aplicar.
5. En este trabajo, notamos que durante cada iteración de Ágil, cada uno puede concentrarse en terminar una parte atómica y pequeña según se le haya asignado, sin tener en cuenta el entregable final, que al pasar las iteraciones podría mutar en algo totalmente distinto a lo que se esperaba durante las primeras iteraciones.
6. Ágil impulsa el trabajo en equipo.
7. La arquitectura se va definiendo y mejorando mientras avanza el proyecto.

Programming in the small vs *programming in the large*

Según vimos en la materia, la arquitectura de software de un sistema de computación es la estructura o estructuras del sistema, que abarcan elementos de software, las propiedades externamente visibles de estos elementos y las relaciones entre ellos.

La arquitectura de un sistema de software es el conjunto de "decisiones principales de diseño".

La arquitectura es una parte del diseño.

Diseño: “Formar un plan o esquema en la mente, para ejecución posterior”.

Hay decisiones de diseño “no arquitectónicas”. Y además, lo que no es arquitectónico ahora, puede ser lo más tarde (depende de la visibilidad de los módulos).

Conclusiones

Luego de haber trabajado con Ágil y con UP, notamos que cada uno de estas metodologías tienen sus características buenas como sus características malas, pero una no es mejor que la otra a priori. No existe una metodología universal que resulte exitosa para cualquier proyecto de software sino que debe de ser adaptada a las características del mismo.

La idea que nos queda es que, antes de empezar un proyecto, hay que analizarlo y decidir qué metodología vamos a usar, según esas características, como ser recursos técnicos, recursos humanos, tiempo de desarrollo, tipo de sistema, etc.