CAMPUS
DE EXCELENCIA
INTERNACIONAL

Máster Universitario en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE MÁSTER

Estudio práctico de soluciones y tecnologías para aplicaciones de emergencias móviles

Autor: Jaime Pajuelo Chavez

Director: Francisco Javier Soriano Camino

MADRID, JUNIO 2019

Resumen

Resumen — Hoy en día, la Web tiende a ser un espacio interactivo donde la comunicación entre un cliente y un servidor web es bidireccional y la información se mueve en tiempo real. Este cambio puede reflejarse en los teléfonos móviles, con la explosión de las apps de mensajería instantánea que ofrecen una nueva forma de comunicación a través del envío de mensajes de texto gratuitos.

Muchas empresas han apostado por remodelar sus servicios para aprovecharse de los beneficios de las tecnologías en tiempo real, debido a que ahora existen herramientas que nos facilitan enormemente este proceso y ya supone un gran esfuerzo.

Pero los centros 112 de Europa todavía no se han aprovechado de estas nuevas tendencias. Hasta este momento, solo ofrecen la posibilidad de comunicarte con los servicios de emergencias a través de llamadas de voz, y a través de sus apps gratuitas, el envío de la ubicación actual del ciudadano y, si es necesario, de fotografías para agilizar una primera intervención.

Ante esta situación, los proveedores de PSAPs deberían contemplar la integración de las tecnologías en tiempo real, para recibir no solo voz, sino también texto en tiempo real, ficheros multimedia, videollamadas u otra información relevante. Y de este modo, proveer un acceso universal como a ciudadanos con algún tipo de discapacidad que les impide realizar una llamada al 112 en condiciones.

A lo largo de este documento, con el objetivo de mejorar la respuesta de los servicios de emergencias, se describe el estudio de nuevas alternativas tecnologías en tiempo real y el desarrollo de una solución software con baja complejidad de integración y cuyo rendimiento cumpla con los requisitos impuestos por los centros 112.

Palabras clave — mensajería instantánea, app, Centro 112, PSAP

Abstract

Abstract — TODO: Resumen en inglés, 250-500 palabras.

Key words — TODO: Palabras clave en inglés, separadas por coma.

Índice general

Índice de figuras

Índice de tablas

1

INTRODUCCIÓN

Internet ha sufrido una evolución radical desde sus inicios. A día de hoy, la Web tiende a ser un espacio interactivo en el cual la comunicación entre un cliente y un servidor web es bidireccional y la información se mueve en tiempo real.

Este gran cambio puede reflejarse en los teléfonos móviles, con la explosión de las apps de mensajería instantánea. Estas apps permiten el envío de mensajes de texto gratuitos a través de Internet. También ofrecen opciones de voz y video, y la posibilidad de compartir archivos.

Pero las apps que permiten comunicarnos con los servicios de emergencias no han evolucionado con la misma rapidez que la tecnología en tiempo real. Estas apps hasta ahora, solo permiten enviar de forma automática la ubicación actual a las centros de emergencias y, si es necesario, enviar fotografías para agilizar una primera intervención.

1.1. Motivaciones del proyecto

El mundo del desarrollo web avanza rápido, cada día surgen nuevas herramientas y nuevas tendencias que debemos implementar si queremos seguir siendo competitivos dentro del mercado. El presente de las aplicaciones web, se podría afirmar que es la respuesta inmediata, lo que en tecnología se conoce como tiempo real.

Estos últimos años, las empresas han apostado por remodelar sus productos para aprovecharse de los beneficios de las aplicaciones web en tiempo real. Esto se debe a que integrar la funcionalidad de tiempo real supone un gran esfuerzo, pero ahora existen herramientas que nos facilitan enormemente este proceso.

Node.js es una de estas herramientas que en los últimos tiempos ha alcanzado una popularidad innegable, hasta llegar a ser un componente indispensable en el desarrollo de aplicaciones web.

Por otra parte, la mensajería instantánea, que ha evolucionado desde los años 90, hoy en día se ha sofisticado y adoptado como parte del uso cotidiano. Las compañías, las organizaciones políticas y otras entidades están utilizando cada vez más la mensajería instantánea como medio para comunicarse con los clientes e incluso, entre compañeros de trabajo.

Pero los centros 112 de Europa todavía no se han aprovechado de estas nuevas tendencias que han revolucionado el Internet. Las apps de emergencias, solo ofrecen la posibilidad de comunicarte con los servicios de emergencias a través de llamadas de voz.

Siendo conscientes de esto, algunos centros de emergencias, buscan incluir aquellas novedades tecnológicas que sean fáciles de incorporar a corto plazo. Con el objetivo de ofrecer una mejor respuesta de los servicios de emergencias y una nueva alternativa de comunicación en tiempo real.

1.2. Objetivos del proyecto

A día de hoy, los ciudadanos usan todos los días comunicaciones basadas en IP, como la mensajería instantánea, y sería interesante la posibilidad de comunicarse con los servicios de emergencias utilizando estos medios. Pero la organización del 112 y los servicios de emergencias solo son accesibles mediante llamadas telefónicas de voz.

Motivo más que suficiente para que los PSAPs cambien su tecnología para ser parte del futuro; es decir, mejorar sus sistemas de emergencias con la integración de las tecnologías en tiempo real, para recibir no solo voz, sino también información de ubicación, texto en tiempo real, ficheros multimedia o videollamadas.

Por esta razón, el trabajo plantea el estudio de las nuevas soluciones y tecnologías disponibles que puedan mejorar los sistemas de emergencias. Y posteriormente, llevar a cabo el desarrollo de un solución software en tiempo real con un baja complejidad de integración. No obstante, al tratarse de un nuevo servicio en los sistemas de emergencias es de vital importancia realizar un análisis del rendimiento y el coste del mismo.

2

ESTADO DEL ARTE

Este capítulo vamos a comprender mejor cómo es el ámbito de los servicios de emergencias, sobre todo el estado actual de las aplicaciones de emergencias para móviles donde se enmarca el proyecto. Además, de las diferentes etapas o componentes que afectan en la integración de uno nuevo servicio a estas aplicaciones de emergencias.

2.1. Servicios de emergencias

Desde 1998, los países de la Unión Europea tienen la obligación de garantizar que los usuarios de teléfonos fijos y móviles puedan llamar sin coste alguno al número de emergencia 112 para comunicar una incidencia de cualquier tipo.

Si la llamada se hace a través de un teléfono fijo, el centro de atención de emergencias conocerá desde dónde se ha realizado la llamada. Pero si se llama desde un teléfono móvil, sólo se podrá saber la zona desde donde más o menos se hace la llamada, en ningún caso el punto exacto en el que se encuentra quien requiere la ayuda.

2.1.1. Número de emergencia europeo

El gran aumento de los viajeros dentro de la Unión Europea hizo que el Consejo de la Unión Europea decidiera introducir un número de emergencia común en todos los estados para evitar la necesidad de recordar diferentes números nacionales dependiendo de la ubicación.

Este número de emergencia europeo fue el 112 y, desde entonces, se encuentra disponible de forma gratuita, 24 horas al día, los 365 días al año, en cualquier lugar de la Unión Europea. Un ciudadano puede marcar el 112 para comunicarse con los servicios de emergencia, incluida también la policía, los servicios de asistencia médica y el cuerpo de bomberos.

La llamada de emergencia a un centro 112 puede realizarse aunque no se disponga de cobertura del operador de red que estemos usando, pero sí de algún otro, pues se utiliza la red GSM (Global System for Mobile communications) que haya disponible. El teléfono móvil realiza la llamada de voz igualmente si se desconoce el PIN e incluso si el teléfono no tiene introducida la tarjeta SIM en el terminal o la pantalla está bloqueada.

2.1.2. EENA

La Asociación del Número de Emergencia Europeo o EENA (acrónimo en inglés de European Emergency Number Association) cree que tener un número de emergencia común en todas partes de Europa está beneficiando directamente a los ciudadanos y visitantes pero desafortunadamente, este número que puede salvar vidas es en gran parte desconocido.

EENA es una organización sin ánimo de lucro con base en Bruselas y establecida en 1999 con el objetivo de promover servicios de emergencia de calidad a través del número de emergencia 112 en toda Europa. Esta organización proporciona

una plataforma de intercambio de información y experiencias entre los servicios de emergencia, autoridades públicas, investigadores y la industria de la tecnología con el objetivo de mejorar la respuesta a las situaciones de emergencia de acuerdo a los requisitos de los usuarios.

EENA, basándose en diferentes fuentes, ha diseñado los modelos 112. Los modelos 112 no cubren todo el modelo de manejo de llamadas, sino que intentan resaltar sus características principales. Los modelos 112 no presentan todos los modelos sobre la organización de los PSAPs (Public Safety Answering Point) en Europa, pero presentan los conceptos principales con descripciones simplificadas.

Modelo donde las EROs atienden las llamadas

En este modelo las llamadas a números nacionales y al 112 son redirigidas a las Organizaciones de Respuesta de Emergencia (ERO, acrónimo en inglés de Emergency Response Organisation). Si se requiere la intervención de una ERO diferente, la llamada y/o los datos sobre la situación de emergencia se envían a la ERO más adecuada. En una variante, dos EROS son colocadas y contactadas a través del mismo número.

Modelo de filtrado de llamadas y envío de recursos

En este modelo las llamadas a números nacionales y al 112 son redirigidas a las Organizaciones de Respuesta de Emergencia (ERO, acrónimo en inglés de Emergency Response Organisation). Si se requiere la intervención de una ERO diferente, la llamada y/o los datos sobre la situación de emergencia se envían a la ERO más adecuada. En una variante, dos EROS son colocadas y contactadas a través del mismo número.

Modelo de recogida de datos y envío de recursos

La diferencia respecto al modelo 2 es el papel que juegan las organizaciones independientes. Los operadores clasifican la llamada y hacen un envio paralelo de las llamadas a las EROs. En algunos casos, los especialistas de las EROs están disponibles para dar soporte a los operadores. Las EROs se encargan del envío de los recursos de intervención.

Modelo de recogida de datos y envío de recursos en la misma sala de control

Este modelo también se realiza en dos niveles. Los operadores y las EROs se encuentran en el mismo lugar. Los operadores están a cargo de clasificar la llamada y, en el caso que sea necesario, hacer un envio paralelo de las llamadas a las EROs más apropiadas. En algunos casos, los especialistas de las EROs están disponibles para dar soporte a los operadores. Las EROs se encargan del envío los recursos de intervención.

Modelo donde las EROs y el PSAP son independientes

En este modelo, los operadores se hacen cargo tanto de la atención de las llamadas como del envío de los recursos de intervención. En algunos casos, los especialistas de las EROs están disponibles para dar soporte a los operadores. El mismo PSAP se encarga de la clasificación de las llamadas, la recogida de datos y el envío de los recursos de intervención a la incidencia.

Modelo donde las PSAPs están interconectados

Los PSAP de diferentes regiones se pueden interconectar. Si no hay disponible un operador, la llamada puede ser redirigida a otro PSAP.

2.1.3. Centro 112 de Madrid

El teléfono 112 se activó en la Comunidad de Madrid el 1 de enero de 1998 para atender, tratar y evaluar todas las llamadas de emergencia de los madrileños. Ese año recibió alrededor de un millón de llamadas. A día de hoy, la cifra supera los 86,6 millones, lo que lo consagra como una gran central de gestión de emergencias que ofrece confianza y seguridad a los ciudadanos.

El Centro 112 de Madrid está diseñado bajo un criterio multiservicio que permite integrar operativamente a todos los organismos de emergencia, mediante los acuerdos precisos para definir los procedimientos que determinan cuándo hay que activar cada servicio. Todo ello, con independencia de que se encuentren integrados físicamente en el Centro 112, como sucede con los principales, o que sus sedes estén ubicadas fuera del Centro.

El Centro 112 de la Comunidad de Madrid, ha incorporado en su Sistema Integrado de Gestión de Emergencias (SIGE112) una novedosa aplicación móvil permitiendo la localización del llamante mediante las coordenadas desde donde se está realizando la llamada.

El trabajo de los profesionales, junto con el desarrollo tecnológico, sirve para optimizar la asistencia y para que la gestión del 112 de la Comunidad de Madrid haya obtenido la norma ISO 22320, siendo el único de España en tener este certificado de calidad.

Modelo 112

Los procedimientos determinan también los intercambios de información necesarios para conocer en todo momento el desarrollo de la gestión de la incidencia. Este tipo de modelo, determina que el trabajo se desarrolle a dos niveles:

1. Recepción, atención y gestión de la llamada. Este proceso corresponde a Madrid 112 y tiene por finalidad, partiendo de la información de cada llamada, activar los servicios precisos que tienen que resolver la emergencia.
2. Movilización y gestión de recursos. Corresponde a los organismos de intervención directa en la emergencia la activación de los recursos adecuados para la resolución de la emergencia. La actividad operativa que corresponde a Madrid 112 se desarrolla fundamentalmente en la sala de operaciones.

El modelo de Madrid 112 está diseñado con criterios de escalabilidad que nos permiten ir incorporando los avances de las nuevas tecnologías.

Plataforma tecnológica

La plataforma tecnológica goza de una serie de atributos, al objeto de brindar un soporte eficaz a la gestión de las emergencias, tales como:

- Capacidad, para atender la potencial demanda de emergencias de los más de 6 millones de ciudadanos existentes en el ámbito territorial de la Comunidad de Madrid. Número al que se deben añadir las personas en tránsito y turistas. El Centro 112 goza de un excedente de capacidad, espacial, operativa y tecnológica para poder hacer frente a casos extremos de catástrofe o gran emergencia, a sabiendas de que lo que funciona bien en el régimen ordinario, lo hará también en el extraordinario.
- Seguridad, entendida como el aseguramiento de la prestación continuada del Servicio 24 h/día, 365 días/año. Este aseguramiento se sustenta en que el suministro de energía está garantizado mediante doble acometida exterior y la existencia de grupos electrógenos con capacidad de soportar y redundar las necesidades energéticas del Centro. El aseguramiento de las comunicaciones y la información se soporta en la redundancia de elementos críticos de los sistemas, y en la existencia de un centro de respaldo del principal.

- Flexibilidad e integrabilidad, características que toman cuerpo en la existencia de un sistema de integración radiotelefónica, capaz de integrar redes de radio de distinta tecnología que conviven en la Comunidad de Madrid y facilitando la interoperabilidad de todos los efectivos de emergencia que actúan en nuestra Comunidad.

2.2. Servicios de emergencias de nueva generación

En 20 países es posible acceder los servicios de emergencia a través de SMS al 112, incluido en España. También se puede en Bélgica, Croacia, Estonia, Finlandia, Francia, Irlanda, Letonia, Lituania, Luxemburgo, Rumanía, Eslovenia, Suecia y Reino Unido. También se puede, pero sólo a través de un número más largo, en Austria, Chipre, Dinamarca, Italia, Malta y Portugal.

A día de hoy, los ciudadanos tienen la necesidad de comunicarse con los servicios de emergencias a través de los medios que utilizan diariamente, y la organización del 112 y los servicios de emergencias están muy dispersos en toda la Unión Europa y aun más en el mundo entero.

Los distintos PSAPs deben afrontarse a ciertos cambios, que tienen un impacto directo en sus organizaciones, para resolver las dificultades en recopilar información relevante relacionada con la emergencia, determinar la ubicación precisa de la persona que llama y proveer acceso universal e inclusivo.

Para que el 112 siga funcionando en toda la Unión Europa de manera equivalente a la actual se necesita un mínimo de estandarización.

2.2.1. AML

En vista de que muchos teléfonos móviles han tenido información de ubicación muy precisa durante varios años, el operador de PSAP de Etapa 1 del Reino Unido, British Telecom, junto con sus socios, establecieron un proyecto, conocido como AML (son las siglas de Advanced Mobile Location).

AML permite que la tecnología de teléfonos inteligentes pase datos de ubicación basados en GNSS o WiFi a servicios de emergencias a través de SMS o HTTPS. En la gran mayoría de los casos, AML proporciona ubicaciones exteriores e interiores con una precisión de menos de 50m y 25m de radio respectivamente.

Hace mucho tiempo que Android integró AML en todos sus dispositivos. En cambio, Apple integró AML en la actualización de iOS 11.3 en marzo de 2018 y funciona en un perímetro de menos de 100 metros en el 63

Como una consideración de desarrollo importante, AML se diseñó de modo que

no interfiera con la llamada de emergencia por voz, por lo que si esta solución se replica en otros países de la Unión Europea, los desarrolladores deben confirmar que tanto el teléfono como la red móvil pueden admitir el establecimiento de ubicación GNSS o Wifi y la transmisión de SMS al PSAP a través de la red GSM durante una llamada de voz de emergencia estándar.

Funcionamiento de AML

Un teléfono móvil de hoy en día habilitado con AML reconoce cuando se realiza una llamada de emergencia y, si en este instante no está activado, activa el GNSS del dispositivo para recopilar la información de ubicación del usuario que llama. El teléfono luego envía un SMS (Short Message Service) automático a los servicios de emergencia (o PSAPs) con la ubicación actual de donde se encuentra el ciudadano, antes de volver a desactivar el GNSS.

SMS ofrece la mejor cobertura geográfica, especialmente en áreas remotas, y además, los SMS de emergencia generalmente no se cobran. El servicio también puede usar Wi-Fi, dependiendo de cuál sea mejor en ese momento dado.

Beneficios de AML

Según EENA, AML es 4.000 veces más preciso que la localización GSM tradicional, con un total del 85 % de las llamadas localizadas dentro de un radio de menos de 50 metros, mientras que con la localización mediante la red móvil, el radio puede tener varios kilómetros.

Las ventajas más llamativas de AML son:

- Una vez implementado, el usuario no tiene que descargar ninguna app o realizar alguna acción adicional, sino que todo se hace de forma automática.
- El sistema ya acumula bastantes casos de éxito.
- La solución no ignora la información de Cell-ID que ya existía, sino que la complementa con información de GNSS o información de Wifi tomada del teléfono.
- Las redes móviles o los proveedores de dispositivos móviles no necesitaron una inversión significativa.

Pero AML todavía tiene algunas limitaciones, la principal es que no se ha extendido de forma global. El sistema ya funcionaba desde el 2016 en Reino Unido, Lituania, Austria y en Estonia, mientras que en otros países de la Unión Europea esperan desplegarlo o en están en fase de prueba.

Android ELS

Android incluye AML, desde la versión Gingerbread OS en adelante, a través del Servicio de Ubicación de Emergencia (ELS, por sus siglas en inglés, Emergency Location Service). ELS fue anunciada por Google en Julio de 2016 y ha sido una de las noticias más importantes de la industria en los últimos años.

Android ELS ayuda a los operadores de redes móviles, a los proveedores de infraestructura de emergencia y a los gobiernos a proporcionar información de ubicación más precisa a los PSAPs durante una emergencia. Cuando los servicios de emergencia reciben una llamada, deben conocer la ubicación de la persona que llama para enviar ayuda y salvar vidas.

Android ELS es compatible con más del 99 % de los dispositivos Android existentes (versión 4.0 y posteriores) a través de Google Play Services. Este servicio está disponible hoy en día y continúan interactuando activamente con los países y socios para hacer que ELS esté más disponible.

2.2.2. NG112

Actualmente, los servicios de emergencia solo son accesibles mediante llamadas telefónicas de voz, por lo que, tienen que cambiar su tecnología para ser parte del futuro; es decir, tener más en cuenta las comunicaciones basadas en Internet.

NG112 (Next Generation 112), un proyecto de EENA, busca la integración de las nuevas tecnologías en los servicios de emergencias, para recibir no solo voz, sino también información de ubicación, texto en tiempo real, ficheros multimedia, videollamadas y otra información relevante.

Una de las razones, por las que desarrolló NG112, fue que los ciudadanos usan comunicaciones basadas en IP todos los días y quieren comunicarse con los servicios de emergencia utilizando estos métodos.

Beneficios de NG112

En contraposición a las limitaciones del modelo actual de 112, NG112 ofrece algunas ventajas:

- Una infraestructura basada en el protocolo IP que implementa estándares abiertos y asegura interoperabilidad entre fronteras, agencias y proveedores.
- Métodos sólidos de adquisición y representación de información de localización con independencia del tipo de red.

- Permite al ciudadano acceder a los servicios de emergencia desde cualquier parte, y desde cualquier dispositivo, usando distintos tipos de medios (texto, voz o video) y aplicaciones.
- Nuevas funciones de mapeado y enrutamiento de llamadas que reemplazan los tradicionales códigos de área.

2.2.3. PEMEA

EENA empezó en 2016 a desarrollar un proyecto que busca que las apps que conectan a los ciudadanos con los servicios de emergencia deben funcionar por completo en cualquier lugar de la Unión Europea. Para lograr esto, las apps deben estar interconectadas de manera estandarizada. No se trata de crear una aplicación única, sino que múltiples aplicaciones móviles sean capaces de funcionar allá de donde estén.

EENA, junto con la empresa francesa Deveryware y la empresa italiana Beta 80, presentó a finales del mes de abril de 2018, el proyecto PEMEA (Pan-European Mobile Emergency App). PEMEA es un estándar ETSI, desarrollado bajo el proyecto H2020 NEXES, que busca la interconexión de apps de emergencias.

El objetivo de PEMEA es que una persona que llama por una emergencia pueda usar cualquier app de emergencias en cualquier lugar de Europa. Permitiendo a cualquier PSAP recibir información vital de la persona que llama, como idiomas o discapacidades, y una ubicación precisa para una respuesta de emergencia más rápida y efectiva.

Además, PEMEA define roles y responsabilidades así como formatos de intercambio de datos y un modelo general de seguridad de manera que los PSAPs puedan asegurarse de la veracidad de la información que les está siendo facilitada a través de la app, y a su vez los usuarios de la app pueden estar seguros de que la información que facilitan no está siendo usada indebidamente.

PEMEA se lanzó oficialmente el 11 de septiembre de 2018 en Madrid (España). De hecho, el beneficio para España es doble, ya que los usuarios no se verán obligados, como hasta ahora, a descargar diferentes apps de emergencias cuando viajan de una Comunidad Autónoma a otra dentro del propio país.

Cómo funciona PEMEA

La arquitectura PEMEA lleva a cabo los siguientes pasos a la hora de atender una llamada al 112:

1. El usuario de la app de emergencias llama al 112.

2. El AP (Application Provider) autentica al usuario, formatea los datos de la llamada antes de enviarlos al PSP (PSAP Service Provider).

3. El PSP recupera información vital del usuario a partir de fuentes de confianza y los proporciona al PSAP.

4. Si el usuario está en itinerancia, los datos se envían al ASP (Aggregating Service Provider) para fines de enrutamiento.

5. El ASP proporciona enrutamiento de los datos para el PSP o PSAP más adecuado, o envía un error.

6. Finalmente, el PSAP obtiene la información enviada y el usuario recibe la ayuda necesaria.

2.3. Apps de emergencias

En un principio, la razón de añadir las apps de emergencias fue el uso de las capacidades de localización de alta precisión que incluyen los teléfonos móviles de hoy en día, para que dicha información pudiese ser enviada a los PSAPs.

Una app de emergencias puede ser muy útil si ocurre una emergencia y hace falta pedir ayuda. Algunas ya existentes permiten enviar de forma automática la localización a las unidades de atención de llamadas e incluso mandar fotografías para agilizar una primera intervención si es necesario. Pero los sistemas de comunicación con los centros de gestión de las llamadas al 112 no han evolucionado con la misma rapidez que la tecnología en tiempo real.

Aunque actualmente existen cientos de apps de emergencias, éstas no están integradas entre sí. Un viajero está obligado a descargarse la app que usan en la ciudad de destino, porque el centro de emergencias de esta ciudad no está habilitado para recibir esa solicitud.

2.3.1. My112

My112 es una app con la que podemos comunicarnos con el servicio de emergencias de ciertas comunidades autónomas de España. La app está disponible tanto en Android como en iOS y, por supuesto, de forma gratuita y sin anuncios. Es sencillo de usar: antes de llamar al 112 para cualquier emergencia, usamos la app para enviar nuestra ubicación actual, incluso si queremos, como información adicional, enviar también fotografías de lo que sucede. De este modo, enviamos mayor información de la emergencia a los PSAPs.

My112 no está disponible en toda Europa, ni siquiera en toda España. Sólo funciona si la utilizamos en determinadas comunidades autónomas. Actualmente, My112 es compatible con los centros 112 de Madrid, Castilla y León, Islas Baleares, Cataluña, Cantabria y Melilla.

En realidad, la app no está desarrollada ni por el organismo que regula el 112 ni por ninguna entidad oficial, sino por Telefónica. Telefónica ha trabajado en coordinación con los centros de Emergencias 112 de las distintas comunidades.

Avisos de emergencias en tiempo real

My112 recibe avisos de emergencias en tiempo real cercanas a tu posición e información actualizada de las mismas en el momento de producirse. Cuando se produzca un aviso de emergencia en la zona donde te encuentres, recibirás una notificación con información asociada.

Estos avisos, para evitar la saturación, sólo se reciben si la persona se encuentra en la "zona cercana". Los servicios de emergencia seleccionan un área en un mapa desde el panel de administración y todos los que estén dentro recibirán las alertas enviadas a ese grupo.

Pulsando sobre un aviso, se accede a la vista del mapa donde se puede observar geográficamente el área afectada por el aviso, nuestra posición con respecto al mismo y el texto del aviso lanzado desde el Centro 112.

En caso de no disponer de datos la aplicación enviará nuestra posición al 112 mediante un mensaje de texto SMS.

2.4. Apps de mensajería instantánea

Las apps de mensajería instantánea son el motivo de que muchos usuarios deciden dar el salto a los smartphones, ya que la posibilidad de enviar mensajes sin coste adicional resulta muy atractivo. Casi todo el mundo usa las apps de mensajería instantánea como principal medio de comunicación.

Hoy por hoy, WhatsApp es el rey indiscutible, sobre todo en España donde tiene una cuota de mercado realmente alta, dejando muy poco espacio para otros competidores. Pero a pesar de todo, si por cualquier motivo no os gusta WhatsApp existen muchas alternativas de calidad para utilizar la mensajería instantánea.

Todas parten sobre la misma base, chat de texto, voz e incluso videollamada, y luego tienen sus particularidades, interfaces de usuario muy distintas, una base de usuarios mayor o menor, y diferentes herramientas enfocadas a la seguridad o la privacidad.

2.4.1. Skype

Skype es el cliente de mensajería instantánea de Microsoft, que además se puede usar para comunicarse a través de voz y vídeo. Al ser multiplataforma, está disponible en Windows, Mac y Linux, así como en aplicaciones para tablets y smartphones.

Skype fundamentalmente lo asociamos con las videoconferencias, donde podemos comunicarnos simultáneamente con hasta 10 usuarios. La posibilidad de llamadas de voz a través de Skype, incluso a teléfonos convencionales si tenemos créditos o estamos en un plan de pago hacen que sea una opción muy atractiva para tener instalada en el teléfono móvil.

Su apuesta por Skype Qik para compartir mensajes de vídeo entre amigos, es una opción interesante con la que trata de ganar terreno entre los usuarios.

2.4.2. Facebook Messenger

Facebook Chat, lanzada en 2008 por Facebook, fue la punta de lanza que renovó al mundo de la mensajería instantánea. En 2011 dio el salto definitivo en las apps con el lanzamiento de Facebook Messenger. Ahora no solo permite compartir archivos, sino también mensajes de voz, llamadas y videollamadas entre otras características como las reacciones ante las imágenes que recibimos de otros usuarios.

Este es el cliente de mensajería instantánea de Facebook, que hoy en día se ofrece únicamente en dispositivos móviles (Android, iPhone, Windows Phone y BlackBerry) o mediante la página oficial de Facebook. Hace tiempo que se volvió independiente, y es uno de los más importantes en todo el mundo.

2.4.3. WhatsApp

El gran revulsivo en el ámbito de la mensajería instantánea, que ha llevado esta solución a ganar popularidad en los smartphones. Es una aplicación que ganó popularidad rápidamente y que fue adquirida por Facebook en 2014. Funciona en iOS, Android, BlackBerry, Windows Phone, Nokia, Symbian y Tizen. El nombre de usuario se define a través del número de teléfono, y ofrece soluciones para poder trabajar con él desde un PC.

WhatsApp es para muchos la app de mensajería instantánea más importante del mundo. WhatsApp se lanzó en 2009 y hasta la fecha continúa dando sorpresas tras cada actualización. Mensajes instantáneos, llamadas de voz, videollamadas, notas de voz, compartir archivos y chats grupales son las características que hacen a WhatsApp la app líder en el mundo.

2.5. Tecnologías de comunicación en tiempo real

TODO

2.5.1. IM

La mensajería instantánea se remonta a la década de los 60, cuando el MIT desarrolló una plataforma que permitía que hasta 30 usuarios pudieran iniciar sesión a la vez y enviarse mensajes entre ellos. El concepto creció en popularidad a medida que la tecnología avanzaba, y ahora damos por sentado la mensajería instantánea y la consideramos parte de nuestra vida cotidiana.

La mensajería instantánea o IM (Instant Messaging) es un servicio de comunicación en tiempo real que permite una conversación, casi siempre con alguien ya conocido, durante la cual un dispositivo móvil o sobremesa está conectado a otro con el fin de intercambiar texto. Utiliza el protocolo IP, siendo un servicio más que se proporciona a través de Internet.

La mensajería instantánea ha evolucionado desde el concepto de las salas de chat en línea públicas de los años 90 y 2000, y se ha vuelto bastante sofisticada y muy común. Algunas compañías usan este servicio como parte de sus herramientas de productividad y comunicación.

Con el auge de las redes sociales y servicios como Facebook y Twitter, así como el cambio a dispositivos móviles como teléfonos inteligentes y tablets, la mensajería instantánea ha perdurado y evolucionado.

La mensajería instantánea crece "de forma imparableçomo primera forma de comunicación, imponiéndose incluso a la comunicación en persona. El uso de la mensajería instantánea es especialmente significativo entre los jóvenes.

Algunos productos de mensajería instantánea son básicos, que funcionan esencialmente como mensajes de texto. Otros sistemas de mensajería instantánea ofrecen opciones avanzadas que le permiten hacer más que enviar mensajes de texto, como la posibilidad de compartir fotos, enviar y recibir archivos.

Cómo funciona la mensajería instantánea

La mensajería instantánea se basa en pequeños programas, conocidos como clientes, que dos personas independientes instalan, y esos programas se conectan para transmitir mensajes escritos entre sí.

Para que dos personas se puedan comunicar usando IM, cada uno debe tener instalado uno de estos programas, que se conectan entre sí para enviar mutuamente

mensajes de texto u otro contenido multimedia.

La forma en que la comunicación ocurre se puede describir de la siguiente manera:

1. Usando un cliente de IM, tecleas tu usuario y contraseña.
2. El cliente se conecta a un servidor usando Internet y algún protocolo de comunicación, que es usualmente específico para el servicio que estés usando.
3. El servidor verifica tu identidad y crea un registro temporal de tu conexión y los contactos que tienes en tu lista.
4. El servidor verifica quiénes de tu lista de contactos está en línea y le da esa información al cliente, que a su vez hará lo necesario para mostrarlos. Asimismo, les indicará a los clientes de esos contactos que tú estás en línea.
5. Seleccionas una persona a la que le enviaras un mensaje. Tecleas tu mensaje y lo envías. En este momento tu software cliente sabe a qué IP y puerto enviar el mensaje y el cliente de tu contacto le muestra el mensaje.
6. La otra persona te escribe un mensaje, repitiendo el proceso y así llevando a cabo una conversación.
7. Cuando cierras tu cliente, el servidor se da cuenta de que estás fuera de línea y le comunica a los clientes de tus contactos que ya no estás en línea. El servidor destruye el registro temporal que se había creado cuando te conectaste.

2.5.2. SIP

TODO

2.5.3. WebRTC

TODO

2.6. Estrategias de comunicación en tiempo real

Para muchos, lo primero que se nos viene a la mente cuando hablamos de aplicaciones web en tiempo real, es el uso de WebSocket, sin embargo, como verás a continuación, la respuesta a la implementación de comunicación en tiempo real, no siempre deben ser usando WebSocket.

A continuación vienen algunas de las estrategias que los desarrolladores web han implementado para poder establecer una comunicación constante con el servidor, que les permita mantener la información actualizada, justo cuando sucede.

2.6.1. Polling

La más simple de todas las formas es el pooling. Es muy simple, para saber si algo pasa, tenemos que preguntar constantemente. La estrategia del pooling consiste en consultar al servidor en un periodo constante de tiempo, usualmente muy pequeño, digamos cada 3 segundos. En términos más técnicos, implementar pooling significa realizar peticiones al servidor cada par de segundos, para consultar información nueva.

Las ventajas del pooling son:

- Es extremadamente simple realizar peticiones constantes cada cierto periodo de tiempo.
- No requiere de tecnologías especiales más que AJAX, para poder hacer las consultas.
- Muy fácil de implementar, sólo colocas tus consultas en un intervalo y listo.

Así como la implementación es muy simple, las desventajas de usar pooling son también muy claras:

- Sobrecargas al servidor, es muy probable que muchas de las peticiones reciban como respuesta nada, en caso de que no haya información nueva que comunicar, sin embargo, las peticiones siguen ejecutándose y el servidor tiene que responderlas.
- Pequeña latencia. Si tu aplicación es de respuesta crítica, considera que con el pooling siempre habrá un ligero retraso entre el momento en el que la información se produce y el momento en el cliente se entera, si por ejemplo, mandas una petición cada 10 segundos, este será el tiempo máximo en que la información podría llegar retrasada.

2.6.2. Long polling

Al notar que muchas de las respuestas que se reciben las peticiones de una implementación con pooling, son respuestas vacías porque el servidor no tiene nada nuevo que comunicar, se introdujo una mejora a dicha estrategia, la llamaron long pooling.

El flujo de una implementación con long pooling es la siguiente:

1. El cliente envía una petición HTTP al servidor consultando información nueva.
2. El servidor tiene dos opciones:
   a) Si existe información nueva que reportar, la envía inmediatamente.
   b) Si no existe información nueva que reportar, mantiene esta conexión HTTP en espera y abierta, hasta que exista algo que reportar, entonces envía la información al cliente y cierra dicha conexión.
3. El cliente recibe respuesta de su mensaje con datos nuevos, ya sea tan pronto como la envió o luego de haber esperado por bastante tiempo a que hubiera algo nuevo que reportar.
4. El cliente manda una nueva petición hasta que la anterior fue contestada, así, esta nueva recibirá respuesta hasta que haya nuevos datos.

La clara diferencia entre el long pooling y el pooling es que en esta mejora de la estrategia anterior, no se envía peticiones constantes, más bien se envía una inicial y las siguientes sólo se envían hasta que hubo una respuesta previa, con datos actualizados. Esto reduce drásticamente la cantidad de peticiones que enviamos hacia el servidor.

Las ventajas del long pooling son las siguientes:

- Todas las del pooling, a final de cuentas es casi la misma metodología.
- Menos peticiones que el pooling, por lo tanto, un servidor con menos carga y más eficiente.
- Mejora significativa en qué tan rápido recibimos los datos nuevos, ya que para cuando estos se crean, ya hay una conexión esperando para que se envíen al cliente.

Las desventajas son más difíciles de identificar, pero sucede:

- Seguimos realizando y abriendo peticiones HTTP, aún cuando estas quizás nunca reciban respuesta.
- Algunos servidores no permiten que las conexiones HTTP permanezcan abiertas por mucho tiempo, por lo que cada que se cierran, debemos crear peticiones nuevas, aumentando la carga del servidor.

2.6.3. WebSocket

En 2010, justo en la cúspide de la popularidad del término HTML5, se introdujo al navegador la habilidad de establecer conexión a dos vías, directamente con el servidor, el protocolo WebSocket.

WebSocket es un protocolo que permite crear un canal de comunicación bidireccional (full-duplex) sobre una única conexión TCP. Está pensado para ser implementado en navegadores y servidores web, aunque no hay ningún impedimento a la hora de implementarlo en cualquier otro tipo de aplicación que siga el modelo cliente/servidor.

Es el mismo concepto de los clásicos sockets de UNIX, pero en la Web, y con la idea de facilitar la transferencia de datos en tiempo real entre un cliente y un servidor web. Siendo una comunicación bidireccional, el servidor web puede enviar la información directamente al cliente durante la conexión.

Las comunicaciones se realizan a través de los mismos puertos que utiliza HTTP con el fin de ofrecer compatibilidad con el software HTTP del lado del servidor ya existente. Es decir, cuando el protocolo trabaja directamente sobre TCP utiliza el puerto 80 y cuando lo hace sobre TLS utiliza el 443. No obstante, WebSocket es un protocolo independiente.

Funcionamiento básico

El protocolo se divide en dos partes: la negociación y la transferencia de datos. Como este coexiste con HTTP, la primera comunicación debe realizarse necesariamente a través de una petición HTTP. Por ello, la negociación de apertura comienza con una petición upgrade por parte del cliente, que tiene el siguiente aspecto:

Cabe destacar que la elección del método GET es una decisión arbitraria tomada por los autores del borrador que finalmente quedó plasmada en el RFC. Aun así, es el único método que contempla el estándar y, por tanto, el único que se debe utilizar. Por su parte, si todo va bien, el servidor responde con un estado 101 (switching protocols), que tiene el aspecto que sigue:

En ambos casos, tanto en la petición como en la respuesta, se incluyen una serie de cabeceras: obligatorias de HTTP/1.1 (Host), necesarias para establecer la negociación (Upgrade, Connection y Sec-WebSocket-*) o por cuestiones relacionadas con modelo de seguridad escogido para el protocolo (Origin).

Una vez el cliente y el servidor han cumplido con su parte de la negociación, y únicamente si no ha ocurrido ningún error, comienza la transferencia de información. A partir de ese momento cada parte puede enviar información a placer sin depender de la otra, cosa imposible de hacer con HTTP, AJAX, las tecnologías push en general

o técnicas más específicas como long polling.

Por último, cuando una de las partes decide que ya no hay nada más que transmitir, es posible cerrar la conexión mediante una negociación de cierre. Esta se inicia enviando un mensaje de control específico, al cual el otro extremo responde con otro mensaje de control para confirmar que el cierre es acordado. La negociación de cierre está pensada para ir acompañada del cierre de la conexión TCP.

2.6.4. SEE

A diferencia de los WebSockets, los Server-Sent Events (SSE) son un canal de comunicación unidireccional donde los eventos fluyen del servidor a cliente únicamente. Los eventos enviados por el servidor permiten que los clientes del navegador reciban un stream de eventos a través de una conexión HTTP sin polling [1].

Un cliente se suscribe a un "stream"de un servidor y el servidor enviará mensajes, event-stream, al cliente hasta que el servidor o el cliente cierre el stream. Depende del servidor decidir cuándo y qué enviar al cliente, por ejemplo, tan pronto como cambian los datos.

SSE es un estándar que describe cómo los servidores inicializan la transmisión de datos con el cliente una vez una conexión de cliente inicial se establece. Esta tecnología fue propuesta por el WHATWG (Web Hypertext Application Technology Working Group) y se implementó por primera vez en el navegador web Opera en el año 2006 [2].

También ofrece una API de JavaScript denominada EventSource implementada en la mayoría de los navegadores modernos como parte del estándar HTML5 de W3C. Hay polyfills disponibles para los navegadores que no son compatibles con la API de EventSource [3].

2.7. Protocolos de mensajería instantánea

En la actualidad existe una gran variedad de protocolos que son usados en la mensajería instantánea, algunos de ellos son propietarios y por tal motivo no ofrecen documentación ni dan acceso a sus fuentes, por otra parte, hay otros que son libres y están muy bien documentados a disposición de todos los usuarios.

Para poder mantener una comunicación a través de mensajería instantánea, es necesario hacer uso de un cliente que realice el servicio. En un primer momento cada servicio permitía conectarse únicamente con los usuarios que utilizaban ese mismo servicio. Más adelante veremos, que algunos protocolos hacen posible conectar varios

servicios desde una misma cuenta.

Con respecto a la seguridad, por norma general, los protocolos no implementan (o habilitaban por defecto) el cifrado de las comunicaciones, transmitiendo éstas en texto claro y quedando expuestas a numerosos ataques como el robo de información, suplantación de identidad, alteración de la información transmitida, entre otros.

2.7.1. MSNP

MSNP (Mobile Status Notification Protocol) es el protocolo de mensajería instantánea de Microsoft. El cliente oficial de mensajería instantánea era Windows Live Messenger. A pesar de la popularidad de Windows Live Messenger, en 2013 cancelaron el servicio forzando a sus usuarios a utilizar en su lugar Skype.

A pesar de que el protocolo MSNP no es de código abierto, a través de técnicas de ingeniería inversa se ha podido conocer su funcionamiento. En la arquitectura del protocolo hay presentes tres tipos distintos de servidores utilizados para distintos procesos, siendo éstos 'Dispatch Server' (DS), 'Notification Server' (NS) y 'Switchboard Server' (SS).

2.7.2. OSCAR

OSCAR (Open System for Communication in Real-time) es el protocolo de mensajería instantánea de AOL. Este protocolo es implementado por los dos principales clientes de la compañía, AIM e ICQ. Al igual que ocurría con MSNP, OSCAR no es de código abierto pero también se ha podido acceder a una gran parte de su funcionamiento gracias a técnicas de ingeniería inversa.

Al igual que MSNP, la arquitectura detrás de OSCAR consta de varios servidores con distintas finalidades, siendo los principales el 'Authorization Server' (AS) y el 'Basic OSCAR Service Server' (BOSS).

Las conexiones con los servidores se realizan a través de distintos canales (frames). Gracias a estos canales es posible realizar comunicaciones paralelas sin necesidad de conectarse a múltiples servidores.

2.7.3. YMSG

El protocolo YMSG (Yahoo! Messenger) fue publicado en junio de 1999 y era similar a MSNP. Una de las novedades que introdujo fue su excelente integración con la Web. Al igual que ocurre con el resto de protocolos comentados, YMSG tampoco es de código abierto.

Con respecto a la arquitectura, se trata de un sistema cliente-servidor pero que, al contrario que en los protocolos MSNP y OSCAR, el cliente se conecta con un servidor aleatorio y a través de él realizará el resto de transmisiones (autenticación, acceso a la lista de contactos, comunicación con otro usuario, etcétera).

2.7.4. XMPP

La especificación base de Jabber, más tarde XMPP, surgió en 1998 por Jeremie Miller, conocido como el primero de carácter abierto y tomado como protocolo por la comunidad open source en 1999, donde ha ido creciendo y evolucionando hasta la actualidad.

XMPP (eXtensible Messaging and Presence Protocol) es un protocolo abierto y extensible, que establece una plataforma para el intercambio de datos en XML y que se utiliza principalmente en servicios de mensajería instantánea.

Posee muchas implementaciones abiertas de servidores, clientes y librerías para las más diversas plataformas y lenguajes. Su funcionamiento topológico se basa en la clásica arquitectura cliente / servidor en la que no existen servidores centrales que gestionan el servicio sino que cualquier usuario puede crear su propio servidor XMPP.

Este protocolo es utilizado en aplicaciones y servicios conocidos como Google Talk, Tuenti, Facebook y WhatsApp (con algunas variaciones del mismo adaptadas al funcionamiento propio del servicio).

2.8. Aplicaciones web

Una aplicación web se basa en una arquitectura cliente/servidor, donde tanto el cliente (interfaz de usuario) como el servidor (servidor web) y el protocolo mediante el que se comunican (HTTP) están estandarizados y no han de ser creados por el desarrollador de aplicaciones.

La arquitectura de una aplicación web suelen ajustarse a un modelo de tres capas. Este modelo supera las limitaciones de las arquitecturas ajustadas a un modelo de dos capas, introduciendo una capa intermedia (capa de proceso), entre la capa de presentación y capa de datos. Cada capa es un proceso separado y bien definido corriendo en plataformas separadas.

Las capas que pueden identificarse en la arquitectura de una aplicación web son:

- Capa de presentación o cliente web. El cliente web es un programa con el que interacciona el usuario para solicitar a un servidor web el envío de los recursos

que desea obtener mediante HTTP. El cliente web recoge la información del usuario a través de una interfaz de usuario y la envía a la capa de proceso, después recibe los resultados de la capa de proceso y presenta los resultados al usuario.

- Capa de proceso o servidor web. El servidor web es un programa que está esperando permanentemente solicitudes de conexión, mediante el protocolo HTTP, por parte de la capa de presentación. En los sistemas Unix suele ser un demonio y en los sistemas Microsoft Windows un servicio. El servidor web recibe una petición por parte de la capa de presentación, éste interactúa con la capa de datos para realizar operaciones y envía los resultados procesados a la capa de presentación.
- Capa de datos o servidor de base de datos. El servidor de base de datos proporciona y almacena datos relevantes para la aplicación. Además, también puede proporcionar la lógica de negocio u otra información administrada por el servidor web.

La interfaz de usuario no es requerida para comprender o comunicarse con el servidor de base de datos. La separación de roles en tres capas, hace más fácil reemplazar o modificar una capa sin afectar a las capas restantes.

2.8.1. Servidor web

Los servidores web son intrínsecos al funcionamiento de las aplicaciones web, lo que exige la necesidad de un mayor énfasis en la arquitectura del servidor web, incluida la capacidad física del servidor: almacenamiento, memoria, potencia de cómputo y rendimiento, además de los niveles de la aplicación. Esto podría estar en cualquier lugar, ya sea dentro del servidor, a través de la red o los sistemas operativos.

Los requisitos de una solución determinan el alcance de las arquitecturas de servidores web; por ejemplo, las soluciones pueden ser aplicaciones simples o de múltiples niveles.

Los diferentes tipos de arquitectura de servidor web incluyen:

Java

En virtud de ser un lenguaje de programación versátil, Java es popular en el entorno de desarrollo empresarial.

Independientemente de la complejidad o la naturaleza de la aplicación, una arquitectura de un servidor web en Java es la plataforma preferida por los

desarrolladores para crear soluciones y entregarlas según las expectativas.

Una de las ventajas distintivas de esta arquitectura es la capacidad de combinar y confiar en las herramientas nativas de Java y los frameworks para crear aplicaciones que abarcan todo el espectro, desde las aplicaciones más sencillas hasta las más complejas.

Node.js

Hace 24 años que Netscape creó JavaScript: un lenguaje de programación creado para manipular las páginas web mediante scripts dentro de su navegador web.

En 2008 Google lanza su navegador Chrome, junto con su motor de JavaScript V8. Un año después Ryan Dahl usaría V8 como base para crear Node.js y cambiar la forma en que se conocía JavaScript.

Node.js es un entorno open source de desarrollo de software o programación con el objetivo de cubrir ciertas necesidades de los programadores a la hora de trabajar con Javascript en el lado del servidor.

Node.js utiliza un modelo de entrada/salida sin bloqueo controlado por eventos, de esta manera lo hace un entorno ligero y eficiente.

Su propuesta se basa en el tratamiento de conexiones de forma unificada a partir de un único hilo complementado con un bucle de eventos de tipo asíncrono. De este modo las peticiones que se vayan haciendo reciben un tratamiento en forma de eventos y pertenecen a este único bucle.

Este nuevo replanteamiento proporciona un lenguaje con la capacidad de gestionar una gran cantidad de solicitudes y conexiones con la máxima eficiencia.

2.8.2. Servidor de base de datos

Los servidores de base de datos surgen en la década de los 80 con motivo de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes de una manera segura y debe proporcionar servicios de forma global y, en la medida de lo posible, independientemente de la plataforma.

Un servidor de base de datos, también conocido como DBMS (acrónimo en inglés de DataBase Management System), es un software que permite la organización de información mediante el uso de tablas, índices y registros. Los servidores de bases de datos se utilizan en todo el mundo en una amplia variedad de aplicaciones.

La información puede organizarse en tablas o en documentos. Cuando organiza-

mos información en un Excel, lo hacemos en formato tabla y, cuando los médicos hacen fichas a sus pacientes, están guardando la información en documentos. Lo habitual es que las bases de datos basadas en tablas sean bases de datos relacionales y las basadas en documentos sean no relacionales, pero esto no tiene que ser así siempre.

En realidad, una tabla puede transformarse en documentos, cada uno formado por cada fila de la tabla. Solo es una cuestión de visualización. Lo que pasa es que a menudo en una base de datos no relacional una unidad de datos puede llegar a ser demasiado compleja como para plasmarlo en una tabla.

Bases de datos SQL

Las bases de datos SQL son el modelo estándar de toda la vida y también el más utilizado en el mundo tecnológico. SQL es un lenguaje de peticiones estructuradas bastante robusto pero muy poco flexible.

En el ámbito informático se habla mucho de ACID, cuyas siglas vienen de las palabras en inglés: atomicidad, consistencia, aislamiento y durabilidad. Son propiedades que las bases de datos relacionales aportan a los sistemas y les permiten ser más robustos y menos vulnerables ante fallos.

La base de datos relacional más usada y conocida es MySQL junto con Oracle Database, seguida por Microsoft SQL Server, PostgreSQL y SQLite.

Algunas ventajas de las bases de datos SQL son:

- Es una tecnología ampliamente conocida y los perfiles que lo conocen son mayoritarios y más económicos.
- Mayor soporte y mejores herramientas debido al largo tiempo que llevan en el mercado.
- Los datos deben cumplir requisitos de integridad tanto en tipo de datos como en compatibilidad.
- La atomicidad de las operaciones en la base de datos. Esto significa que si hay un error durante la petición a cualquier nivel de la operación, se devuelve al punto inicial sin comprometer los datos que fueron utilizados durante el proceso.

Bases de datos NoSQL

Como su propio nombre indica, las bases de datos NoSQL (Not only SQL) son las que, a diferencia de la relacionales, no tienen un identificador que sirva de

relación entre un conjunto de datos y otros. La información se organiza normalmente mediante documentos y es muy útil cuando no tenemos un esquema exacto de lo que se va a almacenar.

Las bases de datos NoSQL son un modelo que se ha puesto muy de moda entre los desarrolladores full-stack porque no requiere un alto conocimiento académico de bases de datos y su curva de aprendizaje y practicidad lo hacen bastante atractivo para proyectos rápidos.

NoSQL se compone generalmente de bases de datos, compuestas a su vez por colecciones que poseen documentos; también hay otras tecnologías NoSQL que poseen columnas y estructuras diferentes.

La indiscutible reina del reciente éxito de las bases de datos NoSQL es MongoDB seguida por Redis, Elasticsearch, CouchDB y Apache Cassandra.

Algunas ventajas de las bases de datos NoSQL son:

- Su naturaleza descentralizada permite una alta escalabilidad. NoSQL es muy utilizada de una amplia forma en aplicaciones con Big Data.
- Son mucho más abiertas y flexibles. Permiten adaptarse a necesidades de proyectos mucho más fácilmente que los modelos de Entidad-Relación.
- Se pueden hacer cambios de los esquemas sin tener que parar la base de datos.
- Escalabilidad horizontal: son capaces de crecer en número de máquinas, en vez de en cantidad de recursos de hardware en una sola máquina.
- No necesita altos recursos para ejecutarse. Cualquier servidor con la mínima cantidad de recursos puede correr una base de datos no relacional.
- Optimización de consultas en bases de datos para grandes cantidades de datos.

2.8.3. Serialización de datos

El universo de servicios y aplicaciones web disponibles hoy en día en Internet es tan inmenso como heterogéneo y muchos de ellos comparten información entre sí. Entonces, ¿cómo logramos que todos estos sistemas, siendo diferentes uno del otro, puedan transmitirse datos? Sencillo, gracias a los formatos de serialización de datos. De no ser por estos estándares creados para representar datos, esta tarea sería un verdadero infierno.

Los sistemas necesitan formatos robustos, que permitan transmitir y compartir información compleja entre sistemas diferentes, con estructuras jerárquicas y atributos variables pero a su vez sean fáciles de leer por un humano. Es aquí donde entran estándares como XML, JSON y YAML.

Los tres formatos de serialización mencionados tienen la misma extensión que su nombre (.xml, .json y .yaml respectivamente). Así que es más fácil de recordar. De hecho, las extensiones de archivo son arbitrarias para los tres estándares de serialización de datos. Es útil para la aplicación y los usuarios saber qué formato de archivos, tipo de contenido y estructura de datos se están utilizando.

Gracias a estos formatos, podemos contar con servicios y aplicaciones que hacen más fácil la vida de desarrolladores y usuarios.

XML

XML (del inglés eXtensible Markup Language) es un lenguaje de marcado, al igual que HTML, que define un conjunto de reglas para codificar información de manera que sea legible por un ser humano y por un ordenador.

XML es una evolución que se inició en el lenguaje GML creado por IBM. XML se usa ampliamente para transmitir información entre servicios web y para definir archivos de configuración. Uno de los lenguajes de programación que le da más soporte es Java.

Una de las fortalezas de XML es el soporte a Unicode, lo que permite escribir la información en cualquier idioma del mundo y otra es el amplio soporte que tiene en la actualidad. Sin embargo, ha sido duramente criticado por su verbosidad y complejidad; mapear una estructura básica XML usando tipos de datos de un lenguaje de programación o bases de datos a veces puede ser muy difícil y poco descriptivo.

Los documentos de texto, hojas de cálculo, páginas web y bases de datos son algunos de los campos de aplicación del XML. El metalenguaje aparece como un estándar que estructura el intercambio de información entre las diferentes plataformas.

JSON

Es un formato para intercambio de datos, un estándar abierto que utiliza texto plano para codificar información en la forma atributo-valor. Su nombre proviene del inglés JavaScript Object Notation y aunque en sus inicios fue considerado como una parte de JavaScript, siempre ha sido independiente del lenguaje de programación y se encuentra disponible para los lenguajes más populares.

JSON es ampliamente usado para intercambio de información entre servicios web y REST APIs. Es usado especialmente en entornos donde el tamaño del flujo de datos es de vital importancia. Su simplicidad y facilidad de implementación le otorgan un gran desempeño y lo convierten en una de las alternativas ideales al momento de reemplazar XML.

YAML

Su nombre proviene del inglés YAML Ain't Another Markup Language. Es otro formato para el intercambio de información que tiene como objetivo facilitar el mapeo de estructuras de datos más complejas (como las listas) en un documento de texto plano legible por un ser humano. Si bien es un formato joven, sus características le han hecho ganarse un lugar importante en la web, junto con XML y JSON.

YAML es más estricto que los anteriores pero también más simple. Estas características le otorgan elegancia y claridad, haciéndolo ideal para tareas que involucren intervención de un humano.

La simplicidad también le otorga velocidad pero, a diferencia del JSON, no es usado en servicios web sino en archivos de configuración, depuración u otros fines en los que la facilidad de lectura juegan un rol importante.

3

EVALUACIÓN DE RIESGOS

El objetivo principal que busca la ingeniería de software es convertir el desarrollo de software en un proceso formalizado, con resultados predecibles, que permitan obtener un producto final de alta calidad y satisfaga las necesidades y expectativas del cliente.

Por esta razón, una fase importante es la representación simplificada del proceso para el desarrollo y evaluación de la solución software de este proyecto, desde una perspectiva que señala los beneficios y riesgos de las decisiones que tomamos en los diferentes marcos de trabajo.

Como en un modelo de desarrollo de software, este capítulo rescata etapas como la especificación de requisitos, el diseño, la implementación y la evaluación, teniendo en cuenta los objetivos principales de este proyecto.

3.1. Identificación de los requisitos no funcionales

Los requisitos no funcionales tienden a ser aquellos que reflejan la calidad del producto. Especifican los criterios que pueden usarse para juzgar el funcionamiento de un sistema como por ejemplo la disponibilidad, accesibilidad, usabilidad, mantenibilidad, seguridad o rendimiento.

3.1.1. Mensajería instantánea

En la actualidad la mensajería instantánea cada vez tiene una mayor presencia en las empresas. Esta situación hace necesaria la adopción de una serie de requisitos a la hora de autorizar el uso de estos servicios. A continuación se detallan algunos de éstos requisitos:

- Confidencialidad de la comunicación. Los mensajes transmitidos a través del servicio de mensajería instantánea sólo pueden ser leídos por el receptor, estableciendo las medidas de seguridad que sean necesarias para evitar que terceros puedan acceder a su contenido.
- Integridad del mensaje. Los mensajes no deben poder ser alterados.
- Identificación. Los mensajes deben contener la información de su emisor y la persona a la que va dirigido, evitando que el mensaje sea entregado a un destinatario equivocado.
- El emisor no debe poder modificar ni eliminar un mensaje ya enviado.

3.1.2. Centro 112 de Madrid

A día de hoy, el Centro 112 de Madrid, está consagrado como una gran central de gestión de emergencias que ofrece confianza y seguridad a los ciudadanos. Por lo que nos puede servir como punta de partida en la identificación de los requisitos no funcionales.

Los aspectos más destacados del Centro 112 de Madrid son:

- Funciona 24 horas al día y 365 días al año.
- Cuenta con 241 profesionales, a los que se suman otros 200 de otros cuerpos y servicios de emergencias que también tienen presencia en el mismo centro de operaciones.

- Atiende más de 4,5 millones de llamadas al año, según las últimas publicaciones, y el tiempo medio de respuesta, desde que se establece la llamada al 112 y el operador responde, es de tan solo 8 segundos. Y, que en 70 segundos, el operador ha enviado la emergencia al servicio correspondiente.
- La sala de equipos está equipada con herramientas de integración de la telefonía en los puestos de trabajo (PCs), para facilitar el trabajo al operador de emergencias mediante el uso exclusivo de pantalla, teclado y ratón.
- Tiene incorporado en su Sistema Integrado de Gestión de Emergencias (SIGE112) la app My112, permitiendo la localización del llamante mediante las coordenadas desde donde se está realizando la llamada.
- Su modelo 112 está diseñado con criterios de escalabilidad.

Teniendo en cuenta esta información relevante, podemos definir los siguientes requisitos no funcionales que la solución software debe cumplir en todo momento:

- Funcionamiento: durante 24 horas seguidas.
- Participantes en la comunicación: entre dos o más personas.
- Tiempo medio de respuesta: 8 segundos.
- Duración media de la comunicación: 70 segundos.
- Comunicaciones simultáneas: alrededor de 250.

3.2. Diseño del protocolo de mensajería instantánea

Desde hace tiempo se sabe que XML es un formato obsoleto y fallido para la serialización de datos intercambiables. Aunque, proporciona todas las características que XMPP necesita, XML no deja de tener su parte de detractores. De hecho, hace algunos años, esto llevó al intento de proporcionar una codificación binaria para XMPP llamada Binary XMPP.

Desafortunadamente, la codificación binaria carecía de las principales ventajas de XML en su legibilidad humana, por lo que la búsqueda de mejores codificaciones nos lleva a JSON.

Por otro lado, si en nuestro proyecto no fuese crítico la comunicación inmediata, sería una buena idea respaldarse en la simplicidad del long pooling. Algunos desarrolladores prefieren usar long pooling porque las tecnologías modernas como son los WebSockets son más complejas y difíciles de configurar. Cuando la simplicidad no es tan importante como el rendimiento, hay que considerar el uso de WebSockets.

3.2.1. Mensajería instantánea grupal

Hoy en día, hay aplicaciones para casi cualquier cosa, incluidas aplicaciones como WhatsApp que te permiten conectarte con dos o más personas al mismo tiempo. Una de las características destacadas de estas aplicaciones de mensajería instantánea es que permiten que grupos grandes hablen y conversen al mismo tiempo. Conocidas como mensajería instantánea grupal, estas aplicaciones las ofrecen de forma predeterminada sin costo.

Dada la naturaleza competitiva de la industria, hay varias aplicaciones que realizan una función similar. Con la multitud de aplicaciones que existen, puede ser bastante difícil elegir una aplicación como base para nuestro desarrollo. Teniendo esto en cuenta, es esencial que nuestra aplicación permita como mínimo el envío de mensajes de texto en un grupo.

3.2.2. WebSocket

Como se ha visto, WebSocket permite establecer comunicaciones bidireccionales en tiempo real en la Web, posibilidad que antes solo existía de forma simulada y bastante costosa mediante técnicas como long polling.

Optar por WebSocket permite reducir la saturación de cabeceras que ocurriría si se utilizase HTTP en su lugar, especialmente para aplicaciones web que requieren un gran volumen de comunicaciones. Además, evita que cada aplicación web utilice una solución de integración diferente, con los problemas de compatibilidad que ello conlleva.

También, se ha visto que su funcionamiento es extremadamente sencillo: se establece una conexión, se envían/reciben mensajes y se cierra la conexión. Al funcionar bajo los mismos puertos que HTTP evita problemas relacionados con cortafuegos, facilitando así el funcionamiento de productos basados en SOA, entre otros.

Por otra parte, es necesario gestionar y mantener un gran número de conexiones que han de permanecer abiertas mientras ambas partes sigan interactuando. Esto puede llegar a ser un problema en determinados casos, teniendo en cuenta que el número máximo de conexiones simultáneas que admite un puerto TCP es de 64.000 y que, además, mantener las conexiones abiertas requiere memoria del servidor.

Por tanto, WebSocket es la mejor solución para aplicaciones web que necesitan actualizaciones constantes en tiempo real como chats, juegos multijugador en línea o retransmisiones interactivas en directo.

3.2.3. JSON

Originalmente, el lenguaje XML era increíblemente flexible y fácil de escribir, pero su inconveniente era que era detallado, difícil de leer para los humanos, muy difícil de leer para los servidores, y tenía mucha sintaxis que no era del todo necesaria para comunicar información.

Hoy en día, XML está muerto para la serialización de datos en la web. A menos que esté escrito en HTML o SVG, ambos hermanos en XML, probablemente no verás XML en otros lugares. Algunos sistemas obsoletos todavía lo usan hoy, pero usarlo para pasar datos tiende a ser excesivo para la web.

La transferencia de datos es mucho más fácil cuando los datos se almacenan en una estructura que está familiarizada a los lenguajes orientados a objetos por lo que es muy sencillo importar datos desde un fichero JSON a Perl, Ruby, JavaScript, Python, y otros muchos lenguajes. En XML, tendríamos que transformar los datos antes de importarlos. Por esta razón, JSON es un formato de fichero superior para las Web APIs.

JSON se usa en todas partes hoy en día. El formato es fácil de escribir tanto por humanos como por máquinas, y se convirtió en el estándar de facto para transferir datos de un sistema a otro. Casi todos los lenguajes de programación tienen una funcionalidad incorporada para leerlos y escribirlos.

Otra de las grandes ventajas que presenta JSON es el rendimiento, ya que los JSON son considerablemente más livianos en peso y mucho más rápido en su procesamiento. Pero como ya vimos, el rendimiento tiene un coste y es la robustez del mensaje como tal.

Un objeto JSON es un objeto válido en JavaScript por lo que es el formato perfecto para este lenguaje. La mayoría de los navegadores web modernos incluyen funciones nativas para codificar y decodificar JSON, lo que le da un punto de ventaja en lo que se refiere a desempeño y disminuyen los riesgos de seguridad.

3.3. Desarrollo de la aplicación web en tiempo real

Una aplicación web en tiempo real requiere un esfuerzo de ingeniería muy grande. En un modelo tradicional son los clientes los que preguntan al servidor, lo que requiere poco trabajo por parte de éste. Las peticiones llegan y el servidor responde. Fácil y sencillo. El servidor no procesa más datos de los necesarios para dar la información que piden los clientes.

Cuando pasamos al modelo del tiempo real, las cosas cambian. Ahora el servidor tiene que preocuparse de dar la información a los clientes cuando ésta se genera. Para ello necesita un registro de clientes que tienen que recibir esa información, así

que no sólo tiene que procesar los datos relativos a la información que va a dar sino que además tiene que preocuparse de decidir con quién conectar y cuándo hacerlo. No sólo eso, también tiene que gestionar las conexiones y desconexiones del sistema. Es decir, un método mucho más complejo que el tradicional.

Con toda esa complejidad del tiempo real, alguna ventaja tenía que darnos. La primera es para los servidores. En una aplicación web tradicional, por ejemplo, si de repente los 200.000 clientes de un servicio deciden hacer una petición todos a la vez, el servidor puede bloquearse por saturación. Sin embargo, si son los servidores los que gestionan las conexiones estas situaciones no pueden ocurrir: el servidor distribuye de forma normal las conexiones de forma que no haya problemas de saturación, y evitando los ataques DDoS involuntarios.

La siguiente ventaja es bastante obvia: la instantaneidad de la información. Cuando son las aplicaciones las que tienen que preguntar al servidor, normalmente lo hacen a intervalos usando pooling. Si vamos encadenando varias aplicaciones, los intervalos se acumulan y podemos tener retardos bastante grandes. Con una aplicación web en tiempo real, estos intervalos desaparecen y la información llega con un retraso mínimo.

Otra ventaja es que genera menos tráfico de red. Por ejemplo, si un cliente de correo comprueba cada cinco minutos si tiene mensajes nuevos, cada cinco minutos va a estar generando tráfico haya o no haya mensajes nuevos. Es decir, que estará creando bastante tráfico inútil. Sin embargo, si es el servidor el que avisa al cliente, sólo se produce tráfico cuando hay correo nuevo. De este modo, el tráfico generado es siempre tráfico útil.

3.3.1. REST

REST es una tecnología que transporta datos por medio del protocolo HTTP. Es tan flexible que permite transmitir prácticamente cualquier tipo de datos, ya que el tipo de datos está definido por la cabecera Content-Type, lo que nos permite mandar formatos como XML o JSON, binarios (imágenes, documentos), texto plano, entre otros más.

REST se caracteriza por no tener estado. Es decir, el servidor no es capaz de recordar el estado de la anterior solicitud REST que pudo, o no, hacer un cliente. Por ello, el cliente tiene que enviar en cada solicitud todo el estado de su sesión.

REST viene para simplificar las cosas. Hoy REST y JSON se han convertido en la opción más sencilla y por tanto más recomendable para implementar una aplicación web.

3.3.2. Node.js

Gracias a que Node.js permite trabajar tanto desde el servidor como desde el cliente, es posible generar una transferencia de información mucho más rápida e inmediata. El resultado de todo esto es una reducción considerable en los periodos de trabajo.

Además al ser un lenguaje popular y empleado por profesionales de todo el mundo resulta fácil encontrar información y recursos en Internet. Está diseñado para incentivar el intercambio entre usuarios y programadores.

Es quizá la opción más competitiva para diseñar aplicaciones que gestionan grandes cantidades de información generadas por una comunidad elevada de usuarios.

Debido a sus altas prestaciones para gestionar y procesar grandes volúmenes simultáneos de información, Node.js es una opción estrella para el desarrollo de aplicaciones como chats online.

Beneficios de utilizar Node.js

Asumiendo que lo que interesa son las ventajas desde el punto de vista del desarrollo web, a continuación se definen las más significativas:

- Escalabilidad de manera sencilla. Node.js se diseñó con una arquitectura dirigida por eventos y con E/S asíncrona desde el minuto 0. Eso hace que sea muy escalable de forma sencilla y directa.

- Rendimiento. Gracias al motor V8 el uso de Javascript en Node.js supera a soluciones basadas en otros lenguajes "interpretados".

- Javascript. Node.js está basado en Javascript, que es un lenguaje que está de moda, en parte gracias al propio Node.js. Y además es el lenguaje de la web. El único soportado por todos los navegadores. Ahora que las aplicaciones web se han hecho muy interactivas y que es ingente la cantidad de código en Javascript que corre en los navegadores, es muy cómodo poder desarrollar con el mismo lenguaje en el servidor.

- Popularidad. Como he mencionado todo esto ha llevado a que Node.js y Javascript gocen de gran popularidad. Cada vez hay más interés y más desarrolladores. Y eso hace que la comunidad sea enorme, lo que es muy interesante en caso de necesitar ayuda.

Event Loop de Node.js

Si una empresa desea que su aplicación soporte más usuarios, necesitará agregar más y más servidores. Por todas estas razones, el cuello de botella en toda arquitectura de aplicaciones web es el número máximo de conexiones concurrentes que podía manejar un servidor.

Node.js resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo del sistema operativo para cada conexión (y de asignarle la memoria acompañante), cada conexión dispara una ejecución de evento dentro del proceso del motor V8 de Node.js.

Node.js emplea un único hilo y un event loop asíncrono. Las nuevas peticiones son tratadas como eventos en este bucle. Este es el motivo por el que las características asíncronas y los eventos de JavaScript encajan tan bien en la filosofía de Node.js.

Esto permite que Node.js sea capaz de gestionar múltiples conexiones y peticiones de forma muy eficiente, lo que lo hace apropiado para desarrollo y aplicaciones con un gran número de conexiones simultáneas.

NPM

En un proyecto Node.js, el código se organiza por módulos o paquetes, así que al momento de trabajar con él va a ser necesario agregar más módulos, es aquí donde entra la herramienta NPM.

Node Package Manager, o simplemente npm, es un gestor de paquetes, el cual hará más fácil nuestras vidas al momento de trabajar con Node.js, ya que gracias a él podremos tener cualquier librería disponible con solo una línea de código. NPM nos ayudará a administrar nuestros módulos, distribuir paquetes y agregar dependencias de una manera sencilla.

Express.js

Muchos desarrolladores de software han optado por utilizar JavaScript para construir aplicaciones web. Esto ha provocado un crecimiento exponencial de los frameworks web para Node.js con el objetivo de facilitar la creación rápida de prototipos y la construcción de aplicaciones web impresionantes, especialmente en el lado del servidor.

Los frameworks para Node.js se utilizan principalmente debido a su productividad, escalabilidad y velocidad, lo que los convierte en una de las primeras opciones para crear aplicaciones web para empresas.

Al usar un framework, puedes trabajar con un conjunto de herramientas,

directrices y prácticas recomendadas que lo ayudan a ahorrar tiempo. También puede ayudar a consolidar los estándares de código en un equipo de desarrolladores.

Express.js es un framework web rápido, flexible y minimalista para Node.js. Es simplemente una tecnología basada en Node.js que se comporta como un middleware para ayudar a administrar nuestros servicios y rutas.

Express.js ha demostrado, con el tiempo, que su popularidad vale la pena con sus métodos y funciones fáciles de usar. Probablemente sea el framework web para Node.js más popular disponible para la comunidad de JavaScript en GitHub con más de 41,000 estrellas.

Algunas de las numerosas ventajas de Express.js incluyen:

- Casi el estándar para el middleware web de Node.js.
- Totalmente personalizable.
- Curva de aprendizaje baja.

3.3.3. MongoDB

Es muy común entre los desarrolladores de aplicaciones web encontrarse en la situación de tener que elegir si se va a usar una base de datos SQL o NoSQL. La mayoría no se lo piensa demasiado y opta por la opción que mejor conocen y con la que más cómodos trabajan. Tampoco es una decisión catastrófica; en realidad, ya sea una base de datos SQL o NoSQL, se puede construir cualquier cosa.

Es importante saber en qué se diferencian y cuál deberíamos usar en cada caso, ya que un buen diseño de base de datos con la tecnología apropiada indudablemente aporta calidad al proyecto. Dependiendo de la naturaleza de la aplicación, interesa que la base de datos tenga unas características u otras.

La diferencia fundamental entre ambos tipos de base de datos radica en que las bases de datos NoSQL no hacen uso de un modelo relacional. Si tu proyecto necesita una escalabilidad importante, donde los recursos son escasos y no necesita respetar la integridad de los datos, entonces sí: NoSQL es la mejor opción.

A la hora de hacer una aplicación web, existen varias soluciones para cada uno de los grandes componentes que forman una aplicación completa. Si el servidor web está en Node.js, MongoDB es la base de datos NoSQL que tiene más éxito entre este tipo de aplicaciones.

Beneficios de MongoDB

Los siguientes son algunos de los beneficios y fortalezas de MongoDB:

- Esquema dinámico. como se mencionó, esto le brinda flexibilidad para el esquema de los datos sin modificar ninguno de los datos ya existentes.
- Escabilidad. MongoDB es escalable horizontalmente, lo que ayuda a reducir la carga de trabajo y la escala de tu negocio con facilidad.
- Capacidad de administración. la base de datos no requiere un administrador de base de datos. Ya que es bastante fácil de usar de esta manera, puede ser utilizado tanto por desarrolladores como por administradores.
- Velocidad. Es de alto rendimiento para consultas simples.
- Flexibilidad. puede agregar nuevas columnas o campos en MongoDB sin afectar las filas existentes o el rendimiento de la aplicación.

3.4. Plan de pruebas no funcionales

Una vez que tengamos probada y desplegada nuestra aplicación, se debe probar de forma automática las capacidades y debilidades del software y de la plataforma sobre la que está corriendo (infraestructura y dependencias), llevándola al límite, para comprobar su disponibilidad, estabilidad y resiliencia.

3.4.1. Pruebas no funcionales

Las pruebas no funcionales se refieren a aspectos del software que pueden no estar relacionados con una función específica o acción del usuario, como la escalabilidad o el comportamiento bajo ciertas restricciones o la seguridad. Estas pruebas determinan el punto de ruptura, el punto en el que los extremos de escalabilidad o rendimiento llevan a una ejecución inestable.

Podemos clasificar las pruebas no funcionales según el tipo de requisito no funcional que abarcan:

Pruebas de seguridad

Las pruebas de seguridad validan los servicios de seguridad de una aplicación e identifican posibles fallos y debilidades. Las pruebas de seguridad son esenciales para el software que procesa datos confidenciales para evitar la intrusión en el sistema por parte de hackers informáticos.

Muchos proyectos utilizan un enfoque de caja negra para las pruebas de seguridad, lo que permite a los expertos, sin conocimiento del software, probar la aplicación en busca de agujeros, fallos, exploits y debilidades.

Pruebas de usabilidad

Las pruebas de usabilidad son para verificar si la interfaz de usuario es fácil de usar y entender. Se refiere principalmente a la interacción con la aplicación. Este no es un tipo de prueba que se pueda automatizar; se necesitan usuarios reales, que sean monitoreados por diseñadores de interfaces de usuario expertos.

Pruebas de rendimiento

Las pruebas de rendimiento verifican la capacidad de respuesta, el rendimiento, la confiabilidad y/o la escalabilidad del sistema cuando está bajo una carga de trabajo significativa.

Por su naturaleza, las pruebas de rendimiento pueden ser bastante costosas de implementar y ejecutar, pero pueden ayudarte a entender si los nuevos cambios van a degradar o mejorar el rendimiento del sistema.

En aplicaciones web, las pruebas de rendimiento a menudo están estrechamente relacionadas con las pruebas de estrés, la medición del retraso y la capacidad de respuesta bajo una carga pesada. Por ejemplo, se pueden observar los tiempos de respuesta cuando se ejecuta una gran cantidad de peticiones de usuario, o ver cómo se comporta el sistema con una cantidad significativa de datos.

3.4.2. Diseño de pruebas de rendimiento

Las pruebas de rendimiento generalmente se ejecutan para determinar cómo se desempeña un sistema o subsistema en términos de capacidad de respuesta y estabilidad bajo una carga de trabajo en particular. Pero, también puede servir para investigar, medir, validar o verificar otros atributos de calidad del sistema, como la escalabilidad, la confiabilidad y el uso de recursos.

Pruebas de carga

Las pruebas de carga se encarga principalmente de comprobar que el sistema puede continuar operando bajo una carga de trabajo específica, ya sea en grandes cantidades de datos o en una gran cantidad de peticiones. Esta carga puede ser el número esperado de usuarios concurrentes, que utilizando la aplicación realizan un número específico de transacciones durante el tiempo que dura la carga.

Una prueba de carga puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Esto generalmente se conoce como escalabilidad de software.

Si también se monitorizan otros aspectos como la base de datos, el servidor de aplicaciones, etcétera, entonces esta prueba puede mostrar el cuello de botella de la aplicación.

Pruebas de estrés

La prueba de estrés empuja los límites funcionales de un sistema. Se realiza sometiendo el sistema a condiciones extremas, como volúmenes de datos máximos o una gran cantidad de usuarios simultáneos. Se utiliza normalmente para romper la aplicación.

Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Esto ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

También se utilizan para, llevado el sistema al colapso o degradación, comprobar su funcionamiento continuado por encima de su límite y, una vez liberado de la carga, evaluar su capacidad de resiliencia volviendo a su estado óptimo de funcionamiento.

Pruebas de estabilidad

Las pruebas de estabilidad o soak testing comprueban si el software puede funcionar continuamente en un período aceptable o superior; es decir, si la aplicación puede aguantar una carga esperada continuada. Generalmente esta prueba se realiza para determinar si hay alguna pérdida de memoria (memory leak) en la aplicación.

Las pruebas de estabilidad son una gran prueba, que a menudo se pasa por alto, y que muestra muchos aspectos críticos sobre una aplicación bajo carga de trabajo constante, cosas que ninguna otra prueba puede y es crítica para determinar si la aplicación es apta para la producción.

Hay que tener en cuenta los requisitos de la aplicación, la organización y la producción para determinar el largo periodo de tiempo que la prueba de estabilidad estará ejecutándose.

En una organización en la que los sistemas de producción se apagan y reinician todas las noches, la prueba de estabilidad no debe durar más de 24 horas. En cambio, si la aplicación se reinicia semanalmente como parte de un ciclo de mantenimiento, entonces la prueba de estabilidad debe durar al menos 7 días.

3.4.3. Herramienta para pruebas de rendimiento

Existen numerosas herramientas, tanto open source como privadas, para realizar las pruebas de rendimiento: NeoLoad, LoadRunner, LoadUI, WebLOAD, Artillery, etcétera. Una de las más populares es Apache JMeter.

Apache JMeter es una herramienta para llevar a cabo simulaciones sobre cualquier recurso de software. Es una herramienta creada por Apache y está completamente escrita en JAVA.

Apache Jmeter se suele usar para hacer pruebas de carga aunque también soporta aserciones para asegurar que los datos recibidos son correctos y muchas posibilidades a la hora de generar reportes.

4
DESARROLLO

TODO: Desarrollo del proyecto

4.0.1. Prueba de carga

TODO: Alcance del trabajo/proyecto

Protocolo WebSocket

TODO: Alcance del trabajo/proyecto

4.1. Motivaciones del proyecto

TODO: Estado del arte

4.1.1. Motivaciones del proyecto

TODO: Alcance del trabajo/proyecto

Motivaciones del proyecto

TODO: Alcance del trabajo/proyecto

Motivaciones del proyecto TODO: Alcance del trabajo/proyecto

5
RESULTADOS

TODO: Pruebas y resultados

6
CONCLUSIONES

TODO: Desarrollo del proyecto

6.1. Conclusiones técnicas

6.1.1. Prueba funcionales

TODO: Alcance del trabajo/proyecto

6.1.2. Aplicaciones web en Node.js

TODO: Alcance del trabajo/proyecto

6.1.3. Memory leaks

TODO: Alcance del trabajo/proyecto

6.2. Implantación real de los resultados

TODO: Alcance del trabajo/proyecto

6.2.1. Contenedor de Docker

TODO: Estado del arte

6.2.2. Instalación on-premise

TODO: Estado del arte

6.2.3. Motivaciones del proyecto

TODO: Alcance del trabajo/proyecto

Motivaciones del proyecto

TODO: Alcance del trabajo/proyecto

Motivaciones del proyecto TODO: Alcance del trabajo/proyecto

7
LÍNEAS FUTURAS

TODO: Desarrollo del proyecto

7.1. Automatización de las pruebas funcionales

Una persona puede ejecutar todas las pruebas mencionadas anteriormente, pero sería una tarea muy costosa y contraproducente. Como humanos, contamos con una capacidad limitada para realizar una gran cantidad de acciones de forma repetible y fiable. Sin embargo, una máquina puede hacerlo rápidamente y comprobar que un requisito del sistema funciona hasta 100 veces seguidas sin quejarse o desgastarse.

Para automatizar las pruebas, primero hay que implementar un programa mediante un framework de pruebas que se adapte correctamente a la aplicación. Jasmine, Mocha y Karma son ejemplos de frameworks de pruebas para JavaScript que a día de hoy están muy demandados.

Cuando las pruebas se pueden ejecutar mediante un script desde una línea de comandos o terminal, puedes hacer que se lleven a cabo de forma automática a través de un servicio de integración continua, como Jenkins. Estas herramientas supervisan los repositorios remotos para ejecutar ese script de pruebas cuando se hayan subido nuevos cambios en alguna rama del repositorio.

7.2. Chatbots

Hoy en día las apps de mensajería están convirtiéndose en uno de los servicios más utilizados por los usuarios de dispositivos móviles. Por esta razón, varios sectores están incorporando estos canales a sus modelos de negocio, un servicio que no estará controlado por personas, sino por robots.

En este sentido, tanto editores como apps de mensajería están centrando todos sus esfuerzos en el desarrollo de softwares de inteligencia artificial para crear chatbots, es decir, bots capaces de entablar conversaciones con los usuarios.

Un bot es un software de inteligencia artificial diseñado para realizar una serie de tareas por su cuenta y sin la ayuda del ser humano.

Los chatbots son un tema candente en estos días. Los conjuntos de herramientas disponibles han permitido a muchas empresas integrar con éxito la tecnología de chatbot en sus sistemas empresariales, ahorrando tiempo y dinero.

Una de las grandes ventajas de los chatbots es que, a diferencia de las aplicaciones, no se descargan, no es necesario actualizarlos y no ocupan espacio en la memoria del teléfono.

Otros lugares en los que han estado en funcionamiento en los últimos años ha sido en chats como Facebook Messenger o en aplicaciones de mensajería instantánea como Telegram o Slack. En estas últimas los chatbots estaban incorporados como si fueran un contacto más.

Los chatbots de inteligencia artificial, el futuro de las apps de mensajería.

7.2.1. Azure V3 Translator Text

Uno de los desafíos que enfrentan los equipos de desarrollo al crear un nuevo chatbot es cómo manejar la traducción de idiomas para un chatbot implementado globalmente. Debido a los aspectos dinámicos de una conversación de chat, la localización de aplicaciones es difícil, en el mejor de los casos.

Lo que necesita un chatbot global es una forma de detectar el idioma entrante del usuario, traducirlo al idioma que su bot entiende y luego traducir la respuesta del bot al idioma de entrada original del usuario. Todo esto debe suceder dinámicamente y sobre la marcha.

La API del Translator Text de los Servicios Cognitivos de Microsoft proporciona las herramientas para lograr esto.

Para realizar la detección y traducción de idiomas, necesitamos utilizar la API del Translator Text. La API del Translator Text puede realizar la detección automática

de idiomas, la traducción, la transliteración y las búsquedas de diccionarios bilingües. Admite más de 60 idiomas y Microsoft continúa agregando más.

Al escribir un chatbot que podría usarse en cualquier parte del mundo, las funciones le brindarán a su aplicación el soporte que necesita para comunicarse con cualquier usuario en su idioma nativo.

Bibliografía

[1] *Long Polling vs WebSockets vs Server-Sent Events*. URL: https://medium.com/system-design-blog/long-polling-vs-websockets-vs-server-sent-events-c43ba96df7c1.

[2] *What are Long Polling, Websockets, Server-Sent Events and Comet?* URL: https://www.geeksforgeeks.org/what-are-long-polling-websockets-server-sent-events-sse-and-comet.

[3] *Polling vs SSE vs WebSocket — How to choose the right one*. URL: https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9.

Apéndices

A

Ejemplos de bloques y comandos útiles en LaTeX

A.1. Ejemplo de sección

Citamos el acrónimo Field-Programmable Gate Array (FPGA).

Bitstream es una secuencia de bits.

La figura A.1 se utiliza en la portada.

Figura A.1: Logo de la Universidad Politécnica de madrid.

Código A.1: Algoritmo de ordenación Quicksort

```
#include <stdio.h>

void quick_sort (int *a, int n) {
    int i, j, p, t;
    if (n < 2)
        return;
    p = a[n / 2];
    for (i = 0, j = n - 1;; i++, j--) {
        while (a[i] < p)
            i++;
        while (p < a[j])
            j--;
        if (i >= j)
            break;
        t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
    quick_sort(a, i);
    quick_sort(a + i, n - i);
}
```

```
#include <stdio.h>

void quick_sort (int *a, int n) {
    int i, j, p, t;
    if (n < 2)
        return;
    p = a[n / 2];
    for (i = 0, j = n - 1;; i++, j--) {
        while (a[i] < p)
            i++;
        while (p < a[j])
            j--;
        if (i >= j)
            break;
        t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
    quick_sort(a, i);
    quick_sort(a + i, n - i);
}
```

La ecuación de Euler ($e^{\pm i\theta} = \cos\theta \pm i\sin\theta$) es citada frecuentemente como un

ejemplo de belleza matemática.

$$a^2 + b^2 = c^2 \tag{A.1}$$