GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CREACIÓN DE UN SISTEMA DE DISTRIBUCIÓN DE VOCES PARA LA APP DE ANDROID AhoTTS

Alumno/Alumna: Arpón Bikandi, Asier

Director/Directora (1): Saratxaga Couceiro, Ibon

Curso: 2019-2020

Fecha: Bilbao, 18 de Julio de 2020

Abstract

The *Aholab* research group has developed a mobile application called *AhoTTS*, which allows the synthetization of text to voice. This is performed by two voices in Spanish and Basque (one voice for each language) that come pre-installed in the application. On the other hand, there is the service of *AhoMyTTS* by which anyone can synthesize their voice and donate it or save it for a private use.

With this project both services have been unified, adding to the application the possibility of downloading the variety of voices donated by the *AhoMyTTS* users and also the private voices that one has stored.

An application has been created in the server that distributes the voices requested and works independently, which allows the development of different client applications and in different platforms.

Now, *AhoTTS* includes the possibility of seeing the public voices catalogue downloaded from the server and also the private ones if the user has been previously validated. Once the list is been obtained, a voice can be selected for its download and installation.

Key words: *AhoTTS*, *AhoMyTTS*, synthesize, voice

Resumen

El grupo de investigación de *Aholab* ha desarrollado una aplicación de móvil llamada *AhoTTS* que permite la síntesis de texto a voz. Esto se realiza tanto en castellano como en euskera mediante dos voces (una para cada idioma) que vienen preinstaladas en la aplicación. Por otro lado, está el servicio de *AhoMyTTS,* con el que cualquiera puede sintetizar su voz y donarla o guardarla para un uso privado.

Con este proyecto se han unido estos dos servicios, añadiéndole a la aplicación la posibilidad de descargar la variedad de voces donada por los usuarios de *AhoMyTTS* y también las voces privadas que uno tenga guardadas.

Se ha creado una aplicación en el servidor que distribuye las voces solicitadas y funciona de forma independiente, lo que permite desarrollar aplicaciones cliente diferentes y en distintas plataformas.

Ahora, *AhoTTS* da la posibilidad de ver el catálogo de voces públicas descargado del servidor y también las voces privadas si se valida el usuario previamente. Una vez obtenida la lista de voces se puede seleccionar una para su descarga e instalación.

Palabras clave: *AhoTTS*, *AhoMyTTS*, sintetizar, voz

Laburpena

Aholab-eko ikerkuntza taldea *AhoTTS* izeneko mugikorreko aplikazioa garatu du, testutik ahotserako sintesia ahalbidetzen duena. Hau, euskaraz zein gaztelaniaz egin daiteke aplikazioan aurretik instalatuta datozen bi ahotsen bitartez (bata hizkuntza bakoitzerako). Beste alde batetik, *AhoMyTTS* zerbitzua dago, zeinekin edonork bere ahotsa sintetizatu dezake eta dohaintzan eman edo erabilera pribatu baterako gorde.

Proiektu honekin bi zerbitzu hauek bateratu nahi dira, aplikazioari *AhoMyTTS*-ko erabiltzaileek dohaintzan emandako ahots anitzak deskargatzeko aukera emanez, eta baita norberak modu pribatuan gordetakoak.

Zerbitzarian aplikazio bat sortu da eskatutako ahotsak banatzen dituena eta modu independentean lan egiten duena. Honek bezero aplikazio ezberdinak garatzea ahalbidetzen du eta plataforma ezberdinetarako.

Orain, *AhoTTS-k* zerbitzaritik deskargatutako ahots publikoen katalogoa deskargatzeko aukera ematen du, aurrez aurretik zerbitzaria baieztatua izan bada. Ahotsen lista lortu izandakoan, bat aukeratu daiteke bere deskarga eta instalakuntzarako.

Hitz gakoak: *AhoTTS*, *AhoMyTTS*, sintetizatu, ahotsa

Índice

Índice de ilustraciones

Índice de tablas

Índice de Acrónimos

API	Application Programming Interface
APK	Android Application Package
HTTPS	HyperText Transfer Protocol Secure
iOS	iPhone Operating System
ID	Identificador
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
REST	Representational State Transfer
TTS	Text-to-Speech
XML	Extensible Markup Language

MEMORIA

1 Introducción

En la actualidad, la tecnología se encuentra en cualquier lugar facilitando el día a día de las personas, desde semáforos que ayudan a regular el tráfico hasta aplicaciones para el teléfono móvil que permiten pagar en bares y tiendas. En muchos casos sirve de ayuda en labores complejas o permite a un individuo realizar tareas que por sí mismo no podría.

Un claro ejemplo de la última situación mencionada es el sintetizador de voz, sistema que permite la conversión de texto en habla, también conocido como TTS (Text-to-speech). Cualquier asistente de voz de los teléfonos móviles es un ejemplo de un TTS, y es cada vez más importante debido a la proliferación del uso de los asistentes inteligentes. Sin embargo, existe un uso de esta tecnología de vital importancia para un grupo concreto de la población: aquellos que sufren trastornos de la voz.

Para aquellas personas que han sufrido una pérdida parcial o total de la voz, los sintetizadores de voz sirven de intermediarios, permitiéndoles seguir comunicándose mediante el habla. Puede que uno de los ejemplos más notorios de esta situación sea el del físico teórico Stephen Hawking, quien, debido a una enfermedad llamada esclerosis lateral amiotrófica (ELA), terminó perdiendo la capacidad del habla. Cuando esto sucedió, adquirió un sintetizador de voz que le permitía, mediante leves contracciones de sus mejillas, componer palabras y frases para ser reproducidas a voz.

Debido a este tipo de discapacidades orales nació el proyecto *AhoMyTTS*[1], una iniciativa del grupo de investigación "Aholab Signal Processing Laboratory"[2] de la Universidad del País Vasco. Este proyecto busca dotar a este grupo de personas de la posibilidad de tener un dispositivo TTS con una voz personalizada.

Ilustración 1: icono del grupo de investigación Aholab

Actualmente, el proyecto dispone de una página web donde cualquier persona puede acceder y donar su voz. De esta forma se va creando un banco de voces públicas que sirve de catálogo para otros usuarios que hayan perdido la voz. Gracias a esta iniciativa, personas con esta clase de enfermedades podrían utilizar más voces además de las pocas genéricas existentes. Asimismo, un usuario que ha donado su voz puede obtener su propia voz personalizada, de tal forma que una persona en conocimiento de una futura pérdida del habla (por una operación quirúrgica o por una enfermedad) puede grabar su voz para su posterior uso.

[1] Página web de *AhoMyTTS*: https://aholab.ehu.eus/ahomytts
[2] Información añadida sobre *Aholab*: https://aholab.ehu.eus/aholab/

Partiendo de la aplicación de *AhoTTS* existente, este proyecto desarrolla una comunicación con una nueva aplicación en el servidor que se encarga de distribuir las voces solicitadas. Además, permite validar el usuario dándole acceso a las voces privadas obtenidas en *AhoMyTTS.*

El presente documento consta primero de una memoria en la que se incluye este apartado de introducción. Después, un contexto que sitúa el proyecto con respecto a la tecnología existente relacionada con sintetizadores de voz y los objetivos que tiene esta implementación. Esta memoria también incluye un apartado que explica los beneficios, otro donde se analizan distintas alternativas para la aplicación y un análisis de los posibles riesgos durante el desarrollo del proyecto. El último apartado de la memoria es el diseño, donde se detallan las funciones de la aplicación.

El siguiente bloque, la metodología, se constituye por dos apartados. El primero describe las tareas realizadas explicando los pasos que se han seguido y el segundo, es un diagrama de Gantt.

Para finalizar el documento, se encuentra el presupuesto, las conclusiones del proyecto y la bibliografía.

2 Contexto

En lo referente a la tecnología de los sintetizadores de voz, grandes empresas como Google o Samsung tienen desarrollados sus propios TTS. En el caso de Google, se pueden encontrar aproximadamente 30 idiomas[1] distintos para usar, por otro lado, Samsung incorpora 45[2] idiomas, pero exclusivamente para los usuarios de sus dispositivos.

Ninguna de las dos compañías previamente mencionadas incluye el euskera entre sus idiomas, la realidad es que por el momento solo existe un TTS del idioma disponible públicamente, el perteneciente al del anteriormente mencionado grupo de investigación *Aholab*. El grupo tiene dos servicios en marcha por el momento: *AhoTTS* y *AhoMyTTS*.

2.1 AhoTTS

AhoTTS es una aplicación para dispositivos móviles tanto Android como iOS gratuita que permite le síntesis de texto a voz. Con la aplicación vienen preinstaladas dos voces, una en euskera y otra en castellano. La interfaz es la mostrada en la Ilustración 2.

Ilustración 2: interfaz aplicación AhoTTS

En la versión en versión en Android el sintetizador se instala como un dispositivo integrado en el sistema operativo y puede ser utilizado por cualquier aplicación que pueda usar síntesis (Adobe Reader, Google Books, etc.).

2.2 AhoMyTTS

El proyecto *AhoMyTTS*, mediante su web, ofrece la posibilidad de crear una voz sintética adaptada a la voz del usuario que después se puede utilizar desde diversas aplicaciones.

Por un lado, los usuarios pueden donar su voz sirviendo esta para crear un banco de voces públicas. De esta forma, los beneficiarios no se verían todos obligados a usar las mismas voces genéricas.

Por otro lado, es posible grabar la voz y no donarla para hacer un uso privado de la voz sintética adaptada.

El uso de *AhoMyTTS* está enfocado para aquellas personas, que, por diversas razones (como enfermedades), han perdido la capacidad de hablar. Esta prestación permite grabar la voz del usuario para posteriormente instalarla en un dispositivo móvil y usarla como la voz del TTS.

Actualmente, ese servicio se ofrece de una manera individualizada (en iOS), obligando a una persona a instalar manualmente la voz del beneficiario en su teléfono, Tablet… Mantener esta asistencia es posible, aunque inconveniente, si las peticiones de uso no exceden cierto límite. No obstante, el proceso en Android es automático, pero no es posible escoger una voz del catálogo de voces públicas.

El interés en este producto ha tenido una crecida tan abrupta que hizo que el proyecto se viralizara, apareciendo en los medios de comunicación de todo el estado. Esto obligó en enero de 2020 a parar la página para evitar problemas de sobrecarga de los sistemas. Este hecho demuestra el gran interés en los beneficios que aporta *AhoMyTTS*, no obstante, debido a la restricción que supone la instalación individual de las voces, se dificulta en gran medida la tarea de alcanzar a un mayor número de la población.

Viendo la crecida de usuarios que ha tenido el servicio, surge la motivación de este proyecto por facilitar a los beneficiarios el uso de un catálogo de voces más amplio que las preinstaladas con la aplicación. Además, se desea que el catálogo sea dinámico de forma que al añadir voces nuevas estén automáticamente disponibles para los usuarios. Ya que *AhoMyTTS* dispone de su banco de voces tanto privadas como públicas, unir ambos servicios contribuiría a ofrecer un servicio más atractivo y de más fácil gestión.

Es interesante solucionar también el inconveniente de necesitar una persona que realice la instalación de la voz y automatizar este cometido de forma que desde *AhoTTS* un usuario pueda descargar la voz deseada sin intermediarios.

Tanto Android como iOS disponen de sus respectivas aplicaciones por lo que conviene que el diseño del mecanismo del servidor permita la interoperabilidad con ambas plataformas.

3 Objetivos y alcance

Mediante el desarrollo de esta aplicación se pretenden lograr dos objetivos principales, ambos relacionados con las funciones que se desean implementar en la aplicación para cubrir las necesidades mencionadas en el apartado anterior. Un objetivo es permitir desde la aplicación actual (AhoTTS) la descarga de voces de un catálogo público, y, el otro objetivo es permitir también a un usuario concreto la descarga de sus voces privadas.

Por otro lado, existen objetivos secundarios ligados al correcto funcionamiento de la aplicación, como el nivel de seguridad implementado, el aprendizaje durante el desarrollo…

3.1 Objetivos principales

3.1.1 Descarga de voces donadas del banco de *AhoMyTTS*

El primer objetivo que se desea cumplir es generar un sistema de distribución de voces que permita la descarga desde el banco de *AhoMyTTS* en cualquier momento sin la necesidad de una persona intermediaría.

Para ello, se añadiría una interfaz al TTS el cual permitiría visualizar en una lista las distintas voces disponibles y posibilitando la descarga de cualquiera de ellas.

3.1.2 Descarga de voces privadas de *AhoMyTTS*

Este objetivo va estrechamente relacionado con el anterior, ya que debe implementar las mismas funciones, pero añadiéndole nuevas como la validación del usuario. Es necesario implementar este método para confirmar la identidad de un cliente y permitir así el acceso a las voces privadas únicamente al propietario de ellas.

3.2 Objetivos secundarios

3.2.1 Seguridad

Para el objetivo de descargar voces privadas es imprescindible desarrollar un sistema de autenticación. Esto conlleva que la aplicación transmitirá unos parámetros de usuario y contraseña por la red para validar a un usuario físico. Este objetivo requiere que la comunicación y el tratamiento de la información se haga de forma que permita asegurar, en la medida de lo posible, que el acceso a las voces esté limitado al propietario de ellas y que la información de la contraseña no sea accesible.

3.2.2 Aprendizaje durante el desarrollo

Otro objetivo secundario es el aprendizaje de los diferentes lenguajes y entornos utilizados durante el desarrollo de este proyecto, Android Studio, Java, PHP (Hypertext Preprocessor)… Trabajar en la mejora de la aplicación debería aportar nuevos conocimientos en los campos recién mencionados.

4 Beneficios

Este proyecto aspira a proporcionar diversos beneficios que pueden clasificarse en tres grupos. Por un lado, están las ganancias sociales, que son las más destacadas. Por otro lado, el interés económico, que en este caso no es muy alto. Por último, son varios los beneficios técnicos que el proyecto ofrece.

4.1 Beneficios sociales

Como se ha mencionado anteriormente, hay diversos casos en los que una persona puede quedarse sin voz, y, en la mayoría de situaciones, las alternativas de las que se disponen son limitadas. Gracias a este proyecto, se les ofrece a todas esas personas con dificultades en el habla la posibilidad de disponer de una gran variedad de voces, para no tener que usar siempre la que por defecto se incorpora en el dispositivo. Además, no solo es posible incorporar una voz donada, sino que, a sabiendas de una futura pérdida (como puede ser el caso de los pacientes de ELA) se puede adaptar la voz y posteriormente incorporarla al dispositivo de una forma sencilla.

Este último servicio ya se realiza en la actualidad, pero mientras que para Android se genera una APK (Android Application Package) particularizada con su voz que el usuario puede descargar e instalar, para iOS es necesario que un profesional instale la voz en el dispositivo de forma individual para cada cliente. El problema reside en que convierte el servicio en algo costoso y que requiere de una persona trabajando de manera personalizada para cada caso. Sin embargo, con estas mejoras en la aplicación este factor humano se elimina, haciendo mucho más sencilla la tarea de incorporar voces personalizadas.

4.2 Beneficios económicos

El proyecto no busca unos beneficios económicos, pues se desarrolla para el grupo de investigación de la Universidad del País Vasco *Aholab* y la actual aplicación está disponible de manera gratuita en la tienda de aplicaciones móviles de Android. Después del desarrollo de este proyecto, la aplicación seguirá ofreciéndose de manera gratuita y es por eso que no se contemplan beneficios económicos.

Sin embargo, permite eliminar la intervención humana necesaria para iOS previamente explicada, por lo que genera un ahorro de coste. Adicionalmente, en un futuro podrían comercializarse voces no gratuitas que supondrían beneficio económico.

4.3 Beneficios técnicos

En cuanto a los beneficios técnicos, gracias a las ampliaciones en la aplicación, ahora aporta los nuevos servicios previamente comentados, la descarga de voces públicas y de voces privadas. Además, todo queda automatizado sin la necesidad de una persona que haga la instalación en el móvil. *AhoTTS* y *AhoMyTTS* se fusionan pasando de tener dos desarrollos separados a uno conjunto lo que facilita el mantenimiento. Estas nuevas funcionalidades al implementarse con una interfaz REST (Representational State Transfer) permite expandirse posteriormente a otras plataformas con facilidad, plataformas como iOS, Windows, Linux…

5 Especificaciones

Previamente, los dos objetivos principales que se desean lograr mediante este proyecto han sido mencionados, para conseguir cada uno de ellos es necesario hacer diferentes implementaciones.

A continuación, se listan las funcionalidades que se deben introducir en la aplicación y se explica brevemente su cometido. Se dividen en cuatro partes, la primera describe las interacciones cliente-servidor o, dicho de otra forma, lo que constituye el protocolo de aplicación. Los siguientes dos apartados serán las especificaciones del cliente y las del servidor. Por último, se encuentran las especificaciones generales.

En cada apartado encontramos cuatro interacciones que se corresponden a la validación del usuario, la obtención del catálogo de voces, la descarga de una voz y la descarga de una demo de la voz. Todas las interacciones se hacen mediante el protocolo HTTPS[3] (HyperText Transfer Protocol Secure) y utilizando el formato POST para incluir los distintos parámetros en su cuerpo.

5.1 Especificaciones del protocolo de aplicación

5.1.1 Validación de usuario

A la hora de validar la autenticidad de un usuario, el cliente manda un mensaje de "login request" en el que se incluyen los siguientes parámetros:

- Tipo
- Usuario
- Contraseña

El parámetro tipo sirve para identificar en el servidor la interacción de la que se trata. A continuación, el servidor responde un mensaje de "login response" que contiene el resultado de la validación. En la siguiente ilustración se puede ver el proceso.

[3] Información sobre el protocolo HTTPS:
https://support.google.com/webmasters/answer/6073543?hl=es

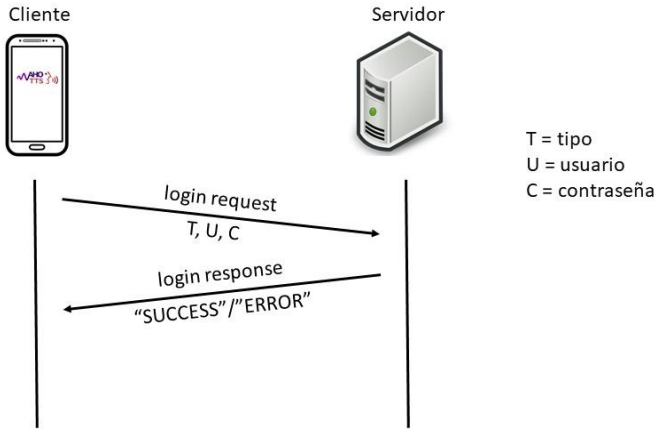

Ilustración 3: interacción de validación de usuario

5.1.2 Obtención de catálogo de voces

Para solicitar el catálogo de voces se manda un mensaje de "voicelist request" con los campos a continuación:

- Tipo
- Usuario (opcional)
- Contraseña (opcional)

Los campos de usuario y contraseña son opcionales ya que no son necesarios para obtener la lista de voces públicas. El servidor devuelve un "voicelist response" con el catálogo en formato JSON (JavaScript Object Notation). Si se ha mandado información sobre el usuario válida, la lista incluirá las voces privadas.

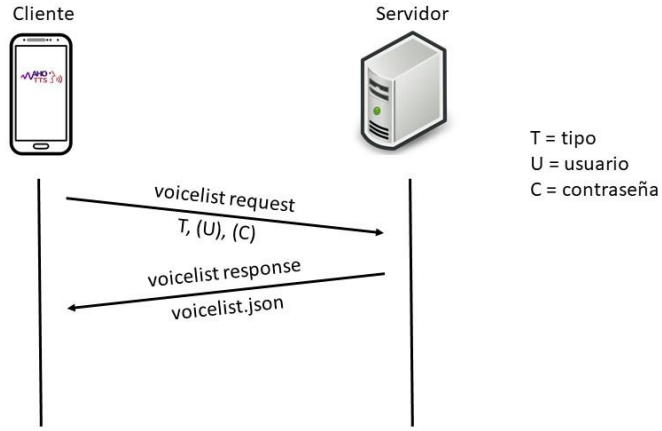

Ilustración 4: interacción de obtención de catálogo de voces

5.1.3 Descarga de una voz

En el momento de descargar una voz, la cual tiene un identificador único, se construye un mensaje de "voice request" para el servidor con los siguientes componentes:

- Tipo
- ID (identificador) de voz
- Usuario (opcional)
- Contraseña (opcional)

Como se puede observar se ha añadido el parámetro que identifica la voz y al igual que en la interacción anterior, el usuario y la contraseña son opcionales. El servidor devolverá un "voice response" en el que se incluirá la voz solicitada.

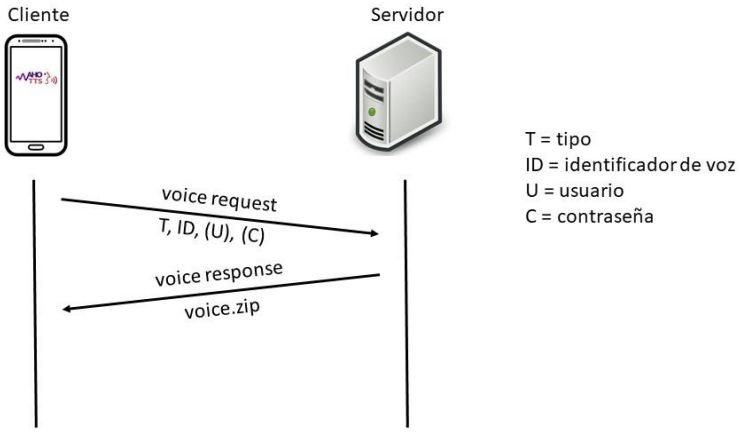

Ilustración 5: interacción de descarga de voz

5.1.4 Descarga de una demo de voz

Para cada voz pública almacenada en el servidor hay un audio que sirve de demo para el cliente. Para descargar este audio se manda un mensaje de "demo request" con los siguientes parámetros:

- Tipo
- ID de voz

El servidor al recibir la información responde enviando el audio correspondiente tal y como se muestra en la Ilustración 6.

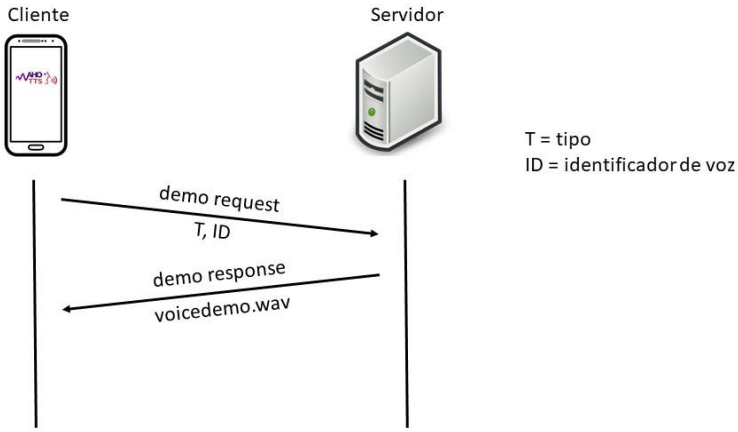

Ilustración 6: interacción descarga de demo de voz

5.2 Especificaciones del cliente

El desarrollo sobre la aplicación de *AhoTTS* se realiza únicamente para Android. Estas modificaciones se realizarán en la pantalla de configuración del motor de síntesis de voz dentro del menú de ajustes estándar del sistema operativo.

En dicha pantalla de configuración, se colocan dos botones que permiten ir a las interfaces de validación de usuario y de descarga de voces respectivamente.

5.2.1 Validación de usuario

En el lado del cliente, para realizar la interacción de validación de usuario es necesario implementar una interfaz que recoja de la pantalla la información del usuario y contraseña. Una vez hecho esto se genera el mensaje POST con estos dos parámetros más el de tipo.

5.2.2 Obtención de catálogo de voces

En el momento que se desee solicitar el catálogo, la aplicación cliente genera el mensaje descrito en el apartado 5.1.2. Al recibir la respuesta, se muestra la lista en la pantalla con el siguiente contenido:

- ID de voz
- Nombre
- Idioma

5.2.3 Descarga de una voz

El catálogo mostrado en pantalla es seleccionable de forma que permite elegir la voz deseada. Al realizar la selección, se recoge el ID de la voz para construir el mensaje para el servidor. Cuando se recibe la voz es descomprimida e instalada sobre la voz anterior de su correspondiente idioma.

5.2.4 Descarga de demo de una voz

Si el usuario desea escuchar una demo de la voz antes de descargarla, dispone de esa opción. Al hacerlo se genera el mensaje para el servidor y cuando se recibe la respuesta con el audio se reproduce.

5.3 Especificaciones del servidor

El servidor se ejecuta sobre Apache e interactúa con DRUPAL (plataforma sobre la que se apoya *AhoMyTTS*) para obtener la validación de usuario.

Todos los mensajes del cliente al servidor llevan el parámetro "tipo", mediante este dato se identifica cual es la operación que quiere realizar el cliente.

5.3.1 Validación de usuario

Al recibir este mensaje, el servidor recoge los datos de usuario y contraseña para validarlos. Dichos usuarios son los pertenecientes a *AhoMyTTS*, y como tales estarán en DRUPAL. Una vez comprobado, se envía un mensaje al cliente con el resultado.

5.3.2 Entrega de catálogo de voces

Cuando llega la solicitud al servidor, carga primero el catálogo de voces públicas y después comprueba los parámetros de usuario y contraseña. En caso de haberlos, comprueba su validez y carga el catálogo privado si es correcta la autenticación. Una vez cargados transforma el formato a JSON y se envía al cliente.

El catálogo contiene la siguiente información por cada voz:

1. Identificador
2. Nombre
3. País
4. Idioma

5.3.3 Envío de una voz

Al igual que con la solicitud anterior, el servidor comprueba si se aporta información sobre el usuario y la contraseña. De ser así, carga la lista de voces privadas para comprobar si la solicitada se encuentra entre ellas y en caso de no estar se carga de las públicas. Una vez cargada se envía la voz en formato ZIP[4].

5.3.4 Envío de demo de una voz

La última interacción posible es la de enviar una demo en audio de una voz el cliente. Cuando el servidor recibe una solicitud de este tipo, recoge el identificador para buscar su audio correspondiente. Por último, carga la demo y la manda al cliente.

[4] ZIP: es un formato de compresión de datos sin perdida

5.4 Especificaciones generales

5.4.1 Eficiencia del código

Este requerimiento de eficiencia hace referencia a la limpieza del código, no repitiendo trozos, sino programando de una forma ordenada y comprensible. De esta manera se consigue facilitar la comprensión del código a una persona ajena que deba entenderlo posteriormente, como podría ser otro programador.

También se incluye en este apartado la eficiencia en el sentido de utilizar el mínimo de recursos posibles del ordenador para ejecutar las distintas operaciones.

5.4.2 Multilingüismo

Diseñar la aplicación de forma que observe el idioma en el que se está configurado el dispositivo y muestre los distintos mensajes o botones en la lengua correspondiente, evitando así el uso de un único idioma estático.

6 Análisis de alternativas

Es posible realizar de más de una forma diferente varios puntos en el proyecto. En este apartado se identifican esos puntos y se analizan en función de diversos parámetros, para así decidir cuál es el más apropiado.

En este caso, son tres los factores a decidir. El primero, el lenguaje de programación para utilizar en el servidor; el segundo, el método de inicio de sesión que se implementa; y el tercero y último, el mecanismo de seguridad que se utiliza en el procedimiento de descarga.

6.1 Lenguaje de programación del servidor

6.1.1 Problemática y análisis de opciones

A la hora de elegir un lenguaje de programación, son varias las características que se deben de tener en cuenta, como el propósito de la aplicación, el entorno en el que se monta… En este caso hablamos de la parte del programa del servidor, y, para las funciones que desempeñan estos equipos, dos de los lenguajes más utilizados son Java y PHP.

Ambos lenguajes aportan ciertas ventajas el uno sobre el otro y es por eso que hay que analizar cuál de los dos es más recomendable para este proyecto en concreto.

6.1.2 Parámetros de análisis

A continuación, se plantean los diferentes parámetros en función de los cuales se evalúa cuál de los dos lenguajes escoger.

1. **Infraestructura necesaria.** Ambos lenguajes deben ejecutarse posteriormente sobre un servidor que gestione las entradas y salidas de las comunicaciones. En cada uno de los casos, los requerimientos que tienen pueden ser distintos.
2. **Escalabilidad.** Se refiere a la capacidad que tiene el lenguaje para añadir funcionalidades y ampliar el tamaño del sistema sin comprometer la calidad normal del mismo.
3. **Simplicidad.** Se entiende por simplicidad, la facilidad que presenta el lenguaje para entender su funcionamiento y el uso del mismo.
4. **Herramientas proporcionadas.** Son las herramientas que proporcionan cada uno de los lenguajes de forma nativa, las cuales facilitan la realización del programa.

6.1.3 Solución

Después de ver los distintos parámetros, se valora la ponderación que corresponde a cada uno de ellos en función de las necesidades del proyecto. Cada lenguaje recibe una nota del 0 al 10 (siendo 10 la mejor) en cada uno de los puntos.

En este caso, la infraestructura necesaria es un factor clave, ya que el ordenador donde se monta el programa está en la universidad, y ya cuenta previamente con un servidor Apache. Este tipo de servidor es válido para PHP, pero no suficiente para Java (ya que además requiere extensiones), y, por lo tanto, en este último caso exige instalar más herramientas que soporten

este lenguaje. Esto, además, es más complejo al tratarse de una universidad que exige ciertos trámites para instalar este tipo de modificaciones.

En cuanto a la escalabilidad, Java ofrece más facilidades para realizar programas grandes y complejos, no obstante, este proyecto exige poca complejidad por parte del servidor, por consiguiente, este parámetro se considera de baja relevancia en la ponderación.

Los últimos dos puntos van unidos, puesto que el hecho de disponer de muchas herramientas supone una mayor complejidad a la hora de saber usarlas correctamente, como ocurre con Java. PHP, por el contrario, ofrece menos herramientas, pero a la vez es más fácil de usar en tareas sencillas.

En la Tabla 1 se ven las distintas calificaciones, y, junto a los parámetros, la ponderación de cada uno de ellos.

Parámetros	Java	PHP
Infraestructura (40%)	6	10
Escalabilidad (10%)	9	5
Simplicidad (30%)	6	8
Herramientas (20%)	9	6
Total	**6,9**	**8,1**

Tabla 1: tabla de decisión del lenguaje de programación

Viendo los resultados, se considera más apropiado el lenguaje de PHP para este proyecto.

6.2 Método de inicio de sesión

6.2.1 Problemática y análisis de opciones

Actualmente, en *AhoMyTTS* ya existe la posibilidad de iniciar sesión y donar voces. Lo cual quiere decir que ya existe una base de datos con la información necesaria de los usuarios para poder hacer una autenticación del usuario.

Ante esto, se plantean dos posibilidades para la implementación del inicio de sesión en el servidor: por un lado, se puede hacer una gestión nueva de los usuarios independiente del servicio de *AhoMyTTS*, y, por otro lado, se puede utilizar la misma base de datos ya existente.

6.2.2 Parámetros de análisis

Los parámetros que se utilizan para analizar ambas opciones son los siguientes:

1. **Atractivo para el usuario.** Este parámetro hace referencia al atractivo que presenta el método de desarrollo en cuanto a pesadez, requerimiento de tiempo del usuario…

2. **Gestión de información.** Esto refiere a que una opción puede suponer mayor dificultad o tiempo a la hora de gestionar la información, para un administrador o para el desarrollador del programa.
3. **Facilidad de desarrollo.** Ambas posibilidades tienen diferentes dificultades o costes.

6.2.3 Solución

Por lo que concierne al atractivo para el usuario, el hecho de tener la gestión del usuario de forma independiente, obliga a registrarse dos veces si se desea usar ambos servicios. De la otra forma, mediante un único registro, se dispondría de todos los servicios, lo que resulta más atractivo y sencillo de utilizar.

Respecto a la gestión de la información, las dos opciones tienen puntos a favor. Si se hace de forma independiente, realizar una gestión más personalizada de los usuarios puede resultar más sencilla, en cambio, utilizando un único sistema, cambios que afecten a ambas partes derivan en una tarea más simple.

Por último, implementar sistemas independientes es más laborioso, ya que hay que crear una nueva base de datos u otro sistema que almacene la información de los usuarios, mientras que utilizando la parte de *AhoMyTTS,* no haría falta su creación.

En la Tabla 2 se muestran las calificaciones de ambas opciones.

Parámetros	Independiente	Conjunta
Atractivo (40%)	6	8
Gestión (30%)	6	6
Facilidad (30%)	6	8
Total	**6**	**7,4**

Tabla 2: tabla de decisión del método de inicio de sesión

Viendo los resultados, se demuestra que es más recomendable para este proyecto utilizar la base de datos existente para el inicio de sesión de la aplicación móvil.

6.3 Seguridad en el procedimiento de descarga

6.3.1 Problemática y análisis de opciones

En lo referente a la descarga de voces privadas, existen distintos mecanismos de seguridad que se pueden introducir para evitar diversos tipos de ataques. Se plantean dos opciones para este proyecto, por un lado, en cada comunicación con el servidor enviar el usuario y la contraseña. Por otro lado, mantener la sesión de un usuario abierta durante un tiempo

determinado, y, al crear esta comunicación, generar un token único que se comparta con la aplicación, el cual asegura la validez del usuario en el tiempo que dure la sesión.

6.3.2 Parámetros de análisis

En este caso se plantean únicamente dos criterios de evaluación:

1. **Nivel de seguridad.** Como bien indica su nombre, este parámetro hace referencia al nivel de seguridad que proporciona cada uno de los métodos.
2. **Tiempo de implementación.** Implica el tiempo necesario para llevar a cabo el desarrollo del programa de cada una de las opciones

6.3.3 Solución

A nivel de seguridad, la opción del token es superior, puesto que el usuario y la contraseña se comparten una única vez y eso supone menos riesgo. De todas formas, en este proyecto el número de veces que se debe reenviar la información del usuario y contraseña es mínimo, y, por lo tanto, no implica una diferencia notable en la seguridad.

En relación con el tiempo de implementación, reenviar la información apenas supone tiempo. Sin embargo, introducir un generador de token y el código para mantener la sesión requiere más desarrollo.

Atendiendo las evaluaciones anteriores, se obserban los resultados en la Tabla 3.

Parámetros	Reenvío	Token
Seguridad (50%)	7	8
Tiempo (50%)	9	7
Total	**8**	**7,5**

Tabla 3: tabla de decisión de seguridad en el procedimiento de descarga

Viendo estos resultados, se implementa el método de reenvío de usuario y contraseña.

7 Análisis de riesgos

Durante la realización del proyecto cabe la posibilidad de encontrarse con imprevistos que obstaculicen el desarrollo de este, por ello, es importante analizar previamente los posibles riesgos.

Primeramente, se crea una lista de los percances que pueden surgir, junto con la probabilidad de que sucedan y la repercusión que tendrán en el proyecto. Después, se evalúa cuáles son los más peligrosos y, por último, se diseña un plan de prevención para evitar o disminuir los riesgos más elevados.

7.1 Identificación de riesgos

A continuación, se presentan los diferentes hechos que se considera que puedan darse lugar durante la realización del proyecto. En cada riesgo se identifica la probabilidad de que ocurra y la repercusión que supone, siendo cinco las posibles calificaciones para ambos criterios: muy bajo, bajo, medio, alto o muy alto.

7.1.1 Definición errónea de las especificaciones (A1)

Durante el diseño de la aplicación, es posible plantear una forma de llevar a cabo las especificaciones que no sea correcta y genere incompatibilidades entre diferentes partes del programa o aparezcan zonas en el código dónde falte información. Dicho problema obliga a alterar el diseño, lo cual, en algunos casos, puede ser una tarea sencilla y, en otros, puede exigir un gran cambio en funciones ya realizadas.

- Probabilidad de que suceda: probabilidad media.
- Repercusión: repercusión media.

7.1.2 Desconocimiento de las herramientas (A2)

El desconocimiento de las herramientas abarca cualquier tipo de retraso sufrido derivado de la falta de nociones de entornos como el *Android Studio* o de los distintos lenguajes de programación usados. La falta de conocimiento puede suponer grandes retrasos, puesto que puede llevar a errores básicos, pero de difícil identificación.

- Probabilidad de que suceda: probabilidad muy alta.
- Repercusión: repercusión alta.

7.1.3 Fallo del servidor (A3)

Una parte de este proyecto está desarrollada en el servidor que, en este caso, se encuentra en la universidad. Cabe la posibilidad de que haya algún fallo en el equipo derivado de, por ejemplo, un corte de luz en la universidad que impida realizar pruebas del funcionamiento de los distintos servicios. Tales percances no suelen ser muy probables, pero, al ser un equipo que se encuentra en la universidad, reiniciarlo o arreglarlo en caso de avería puede llevar más tiempo que en otros casos.

- Probabilidad de que suceda: probabilidad baja.
- Repercusión: repercusión media.

7.2 Evaluación de riesgos

Para decidir cuál de los riesgos requiere más atención, a continuación, en la Tabla 4, se tiene en cuenta tanto la probabilidad de que sucedan como él impacto que tendrían sobre el proyecto.

Como se puede apreciar, cuanto más próximo esté un riesgo a la esquina inferior derecha, más peligro supone para el proyecto, y, por lo tanto, cuanto más alejado menos preocupante será.

		Repercusión				
		Muy baja	Baja	Media	Alta	Muy alta
Probabilidad	Muy baja					
	Baja			**A3**		
	Media			**A1**		
	Alta					**A2**
	Muy Alta					

Tabla 4: evaluación de riesgos

Como se ha podido observar, el desconocimiento de las herramientas es el factor más crítico en cuanto a riesgo para el proyecto, y, por ello, en el próximo apartado se contempla un método o procedimiento para disminuirlo.

7.3 Respuesta ante los riesgos

Este riesgo se produce, como bien indica su nombre, por la falta de conocimientos de las diferentes herramientas o lenguajes de programación que se utilizan durante el desarrollo de la aplicación.

En el apartado de análisis de alternativas se ha concluido que el lenguaje que se utiliza para el servidor es PHP y la aplicación se sigue desarrollando en Java junto al entorno *AndroidStudio*.

Una vez analizadas las distintas herramientas que se utilizan, se plantean dos formas de minimizar tanto la posibilidad de futuros obstáculos en el proyecto, como la repercusión que pueden tener.

1. **Aprendizaje previo.** Antes de la iniciación del proyecto, se toma un tiempo de aprendizaje mediante las herramientas a disposición (cursos online, tutoriales de internet, libros de programación…). De esta forma, se reduce el desconocimiento y es más sencillo evitar errores.
2. **Identificación de un programador experimentado.** Previo al inicio del proyecto, convenientemente, se busca en el entorno (profesores, compañeros…) personas que conozcan en profundidad las distintas herramientas del proyecto, y se pone en contacto con ellas. Mediante esto, se consigue disponer de una persona de ayuda en caso de quedarse bloqueado en algún punto durante el desarrollo de la aplicación.

8 Diseño

El apartado de diseño se divide en dos subapartados. El primero, consta de un diseño de alto nivel, en el cual se explica el modo de implementación de las especificaciones desde una perspectiva más general. El segundo, por el contrario, es un diseño da bajo nivel donde se analizan las diferentes funciones, métodos, clases, etc. que se han empleado en la programación.

8.1 Diseño de alto nivel

El proceso de desarrollo de la aplicación se basa en el siguiente esquema:

- Descarga de voces públicas
  - Establecimiento de la comunicación
  - Obtención de lista de voces públicas
  - Visualización de lista
  - Descarga de voz publica
  - Instalación de voz
  - Descarga de una demo de voz
- Descargar de voces privadas
  - Validación de usuario
  - Integración de la lista de voces privadas
  - Integración de descarga de voces privadas

Las diferentes pantallas y las interacciones entre ellas se detallan en el apartado “Diseño de bajo nivel”.

8.1.1 Descarga de voces públicas

En la Ilustración 7 se pueden ver los distintos componentes que forman este servicio. En el lado del cliente está el dispositivo móvil con la aplicación de *AhoTTS* que inicia la comunicación, y, en el otro lado, el servidor con la aplicación de PHP[3], la lista de voces (XML[5]) y las voces.

[5] XML = Extensible Markup Language

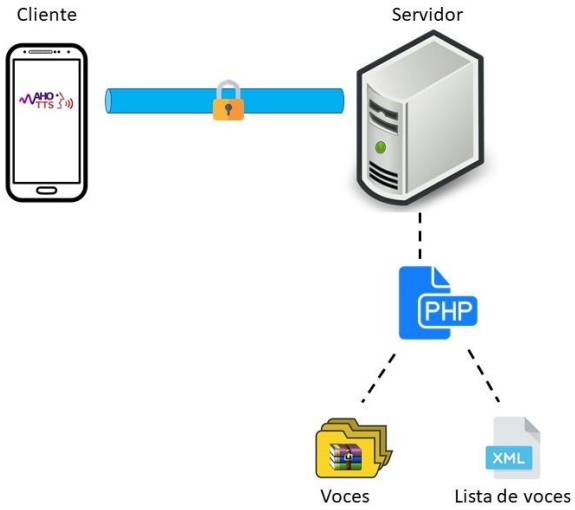

Ilustración 7: esquema descarga voces públicas

Para desarrollar esta función se deben cumplir todas las tareas del apartado anterior, empezando por la comunicación.

8.1.1.1 Establecimiento de la comunicación

Para poder efectuar cualquier tipo de operación de envío o recepción de información, el primer paso a realizar es abrir un canal de comunicación entre el cliente y el servidor.

Cliente

El cliente abre la comunicación con el servidor utilizando el protocolo HTTPS ya que la información debe ir cifrada.

Servidor

El servidor está preparado para recibir comunicaciones de HTTPS, en concreto se utiliza el formato POST respetando la arquitectura REST (Representational State Transfer)[4].

8.1.1.2 Obtención de lista de voces públicas

Para obtener el catálogo de voces públicas los pasos realizados por el cliente y el servidor son los siguientes:

Cliente

El cliente primero debe establecer la comunicación y una vez realizada envía un "voicelist request" al servidor. El parámetro que incluye es el de "tipo" indicando la interacción de la que se trata, en este caso, "tipo=1"[6]. Al recibir la respuesta del servidor el cliente cierra la conexión.

Servidor

[6] En el apartado 8.2.2 se incluye una tabla que resume las distintas interacciones en función del parámetro "tipo"

Cuando el servidor recibe la solicitud, carga la lista almacenada en formato XML y la transforma a JSON. Este cambio se realiza porque el segundo formato ocupa menos espacio, y, por consiguiente, es más conveniente para el envío de información. Por otro lado, XML es más visual, es por ello que se almacena con esa configuración. Una vez hecha la transformación el servidor envía la lista al cliente

En la Ilustración 8 podemos ver todo el proceso descrito.

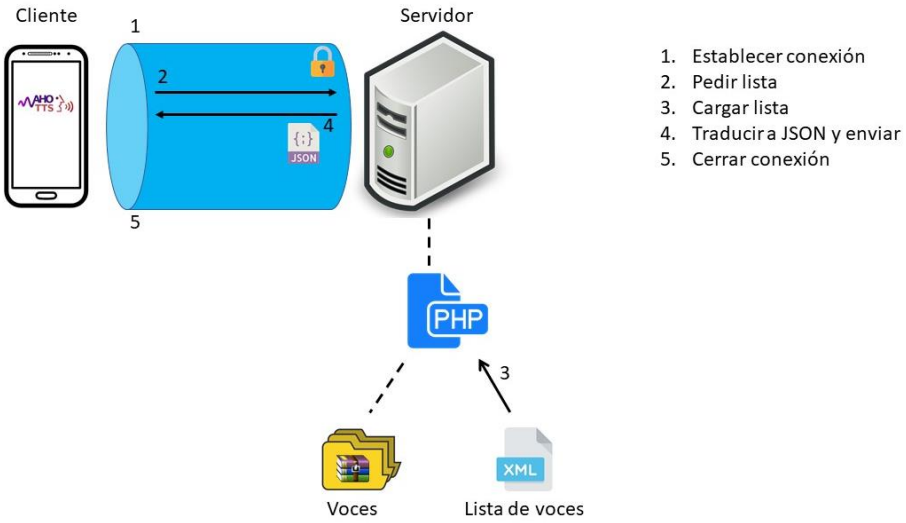

Ilustración 8: esquema comunicación descarga de lista de voces

8.1.1.3 Visualización de lista

Cliente

Una vez obtenida la lista en el cliente, se visualiza en pantalla con la siguiente información:

- Nombre de la voz
- Identificador
- Idioma

El sistema de visualización admite seleccionar elementos, ejecutando una función al hacerlo. Este diseño está pensado para posteriormente descargar la demo correspondiente a la selección. A la derecha de la lista se inserta un botón que permite descargar la voz. En la Ilustración 9 se puede observar el resultado.

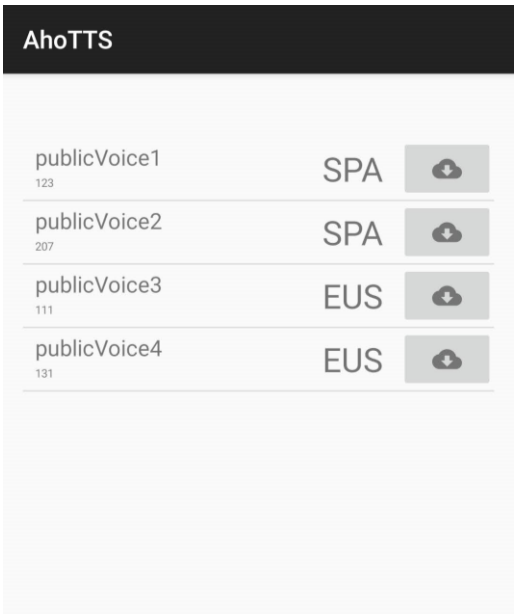

Ilustración 9: pantalla de la aplicación mostrando lista de voces

8.1.1.4 Descarga de una voz concreta

Para realizar el proceso de descarga de una voz, las operaciones realizadas por cliente y servidor son las siguientes:

Cliente

En el cliente, tal y como se ha mencionado en el apartado anterior, al generar la lista el usuario puede seleccionar la voz que desea descargar. Cuando se clica el botón de la derecha de una voz, comienza el proceso de descarga. Una vez más, lo primero es establecer la conexión, y, una vez realizada, el cliente envía una petición con el ID (identificador) de la voz elegida. El parámetro "tipo" también es especificado ("tipo=2"). Al recibir la voz se guarda y se cierra la conexión.

Servidor

El servidor, al ver el ID y la petición, recoge la voz de una ubicación en el equipo no accesible desde el exterior de él. Este diseño no es importante con las voces de uso público, y está realizado de esa forma con vistas a las descargas privadas, a fin de mantener un planteamiento similar y seguro. Las voces se almacenan y se envían en formato ZIP (formato de compresión de ficheros).

En la siguiente ilustración se puede ver el esquema.

Ilustración 10: esquema comunicación descarga de voz

8.1.1.5 Instalación de voz

Cliente

Una vez descargada la voz en ZIP, se descomprime y sobrescribe la voz previamente instalada del idioma correspondiente. Los distintos pasos de descarga y borrado se realizan en ese orden, ya que, si hubiera un problema durante la descarga, no se borraría la voz previa dejando el sintetizador sin poder reproducir.

8.1.1.6 Descarga de una demo de voz

Cliente

Al clicar en una voz de la lista (no en el botón) se inicia la descarga de la demo. La aplicación cliente inicia la conexión con el servidor y solicita el audio mandando un "demo request" que incluye el parámetro "tipo=3". Cuando recibe la respuesta, la voz se almacena y reproduce en el dispositivo móvil.

Servidor

Cuando recibe una solicitud de descarga de demo, el servidor recoge el identificador de la voz, busca el directorio correspondiente donde se almacena el audio y se lo envía al cliente.

8.1.2 Descarga de voces privadas

En este punto se implementa la validación de usuario y, por lo tanto, el esquema varía, pero no excesivamente. Como se puede observar en la Ilustración 11, se han añadido los servicios de *AhoMyTTS* para hacer la autenticación (la API que ofrece DRUPAL[5] para PHP) y obtención de la información de las voces de cada usuario.

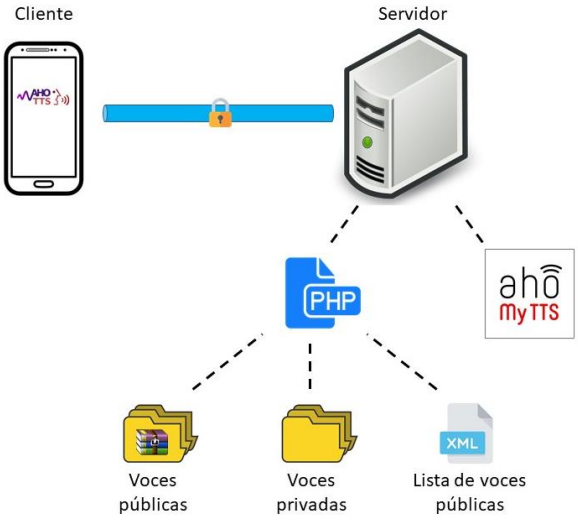

Ilustración 11: esquema descarga voces privadas

8.1.2.1 Validación de usuario

La validación de usuario se analiza a continuación dividiendo la parte del cliente y del servidor.

Cliente

En la aplicación cliente, el usuario dispone de la interfaz mostrada en la Ilustración 12 para introducir la información necesaria (usuario y contraseña).

Ilustración 12: pantalla inicio de sesión

Al presionar en el botón de "iniciar sesión", comienza el proceso de autenticación. Como se puede observar en la Ilustración 13, el primer paso es el establecimiento de la conexión. Una vez

realizado, el cliente envía la información al servidor de usuario y contraseña, además del parámetro "tipo" que en este caso es "tipo=0".

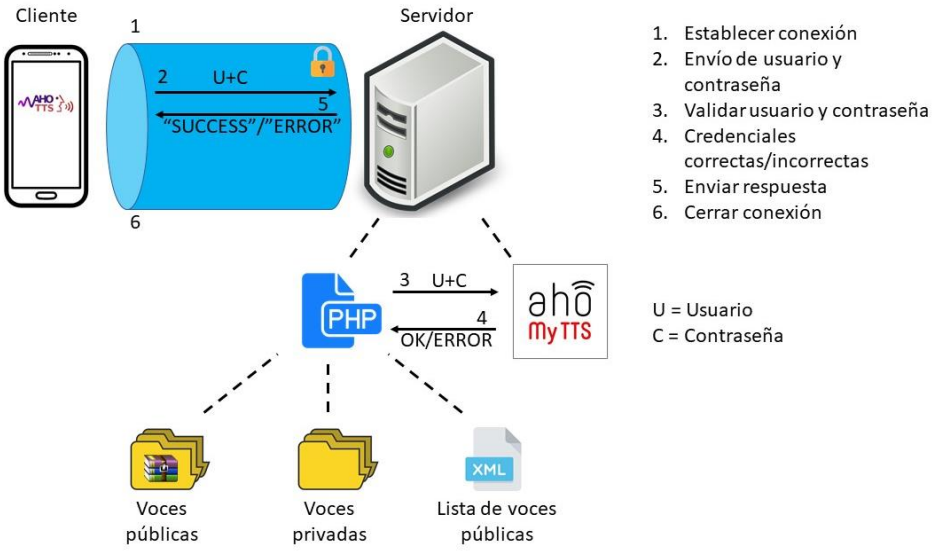

Ilustración 13: esquema comunicación validación de usuario

Por último, se recibe la respuesta del servidor que puede ser de dos tipos:

- Validación correcta: "SUCCESS"
- Validación errónea: "ERROR"

Si la validación es correcta se realizan ciertos cambios en la interfaz, bloqueando la entrada de texto del usuario y contraseña, y sustituyendo el botón de iniciar sesión por uno de cerrar sesión.

Servidor

Al recibir la información del cliente, el servidor recoge los parámetros de usuario y contraseña para mediante la API de DRUPAL validarla. Si la operación tiene éxito le enviará un mensaje de "SUCCESS" al cliente, si no, el mensaje será de "ERROR".

8.1.2.2 Integración de la lista de voces privadas

Cliente

Con estos cambios, el cliente añade ahora los parámetros de usuario y contraseña en la comunicación (si existen).

Servidor

Partiendo del funcionamiento previo de la obtención de lista, ahora se añaden algunas operaciones nuevas que permiten enviar también la lista de voces privadas de un usuario. Con estos cambios, si el servidor recibe los parámetros de usuario y contraseña, después de cargar

la lista pública pasa a validar esas credenciales. Si la información es correcta, solicita la lista de voces que corresponden a ese usuario y combina ambas listas.

A continuación, se transforma la lista a formato JSON y se envía al cliente. En las Ilustración 14 se pueden apreciar las modificaciones implementadas.

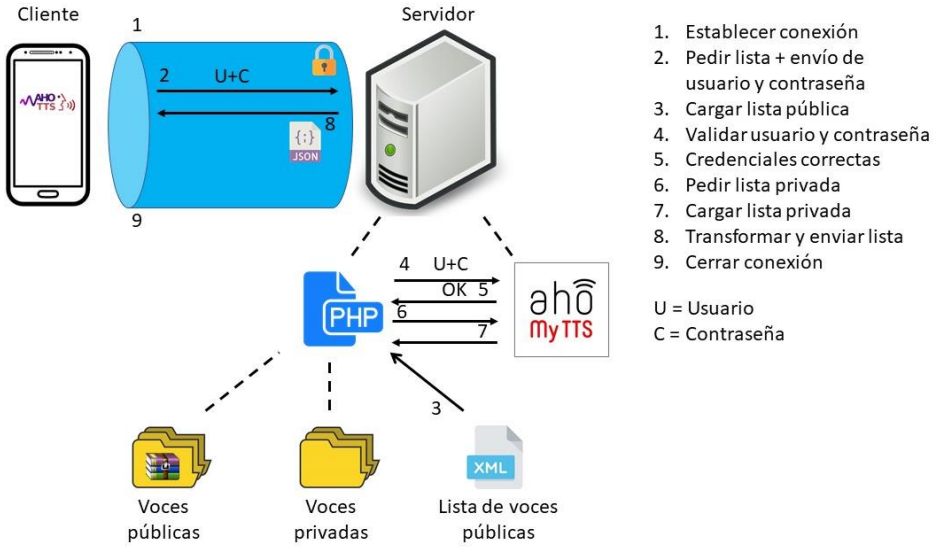

Ilustración 14: esquema comunicación descarga de lista de voces modificado

8.1.2.3 Integración de descarga de voces privadas

Cliente

Al igual que en el apartado anterior, al descargar una voz se añaden algunas modificaciones para validar al usuario. Si se dispone de información del usuario y contraseña se envía al servidor independientemente de si es pública o privada.

Una vez enviado, recibe la respuesta con la voz en formato ZIP, se descomprime y se instala al igual que en la versión anterior.

Servidor

Una vez reciba la información, el servidor la comprueba con *AhoMyTTS*. Si es correcta, le pide la lista de voces privadas que corresponden a ese usuario. Este paso es importante para comprobar que la voz solicitada corresponde realmente a dicho cliente.

Las voces privadas no están almacenadas en formato ZIP y, por lo tanto, antes de enviarla se comprime en una carpeta temporal (tmp) de la cual se borra posteriormente.

Por último, la voz se carga y se envía al cliente.

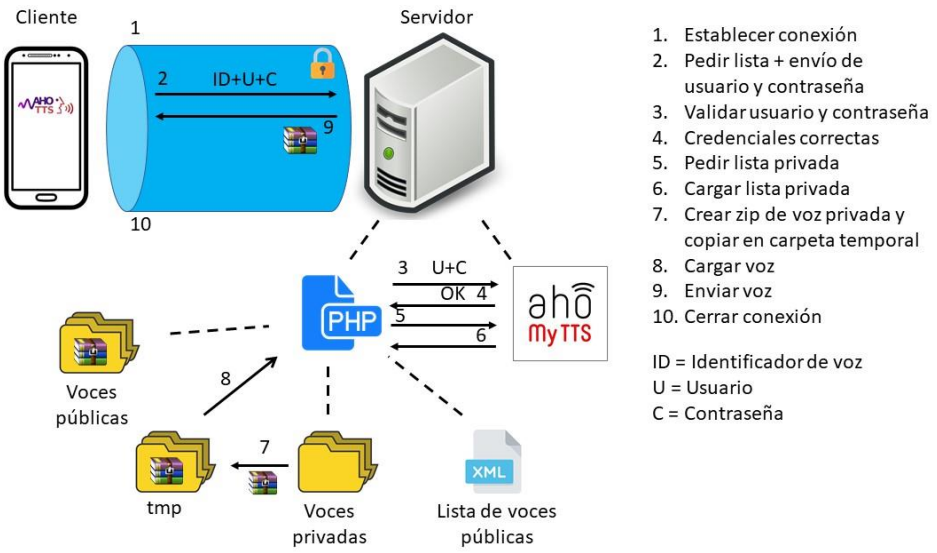

Ilustración 15: esquema comunicación descarga de voz modificado

8.2 Diseño de bajo nivel

Una vez explicado el diseño de la aplicación de manera algo general, a continuación, se muestran las distintas clases, métodos, atributos, funciones… usados en los programas cliente y servidor.

8.2.1 Cliente

Se presentan únicamente las clases utilizadas durante este proyecto, excluyendo las existentes al inicio del mismo siempre y cuando no afecten a las nuevas funciones implementadas.

Como se puede observar en la Ilustración 16, hay 8 clases divididas en 3 grupos. El primer grupo corresponde con las actividades, las cuales son las encargadas cada una de una interfaz de usuario, o pantalla. El segundo grupo son clases implementadas dentro de otras clases, dos para desarrollar conexiones con el servidor y una que sirve de ayuda para la visualización del catálogo de voces (más adelante se describirán en detalle). El último grupo, son clases que sirven de ayuda para ejecutar diversas funciones.

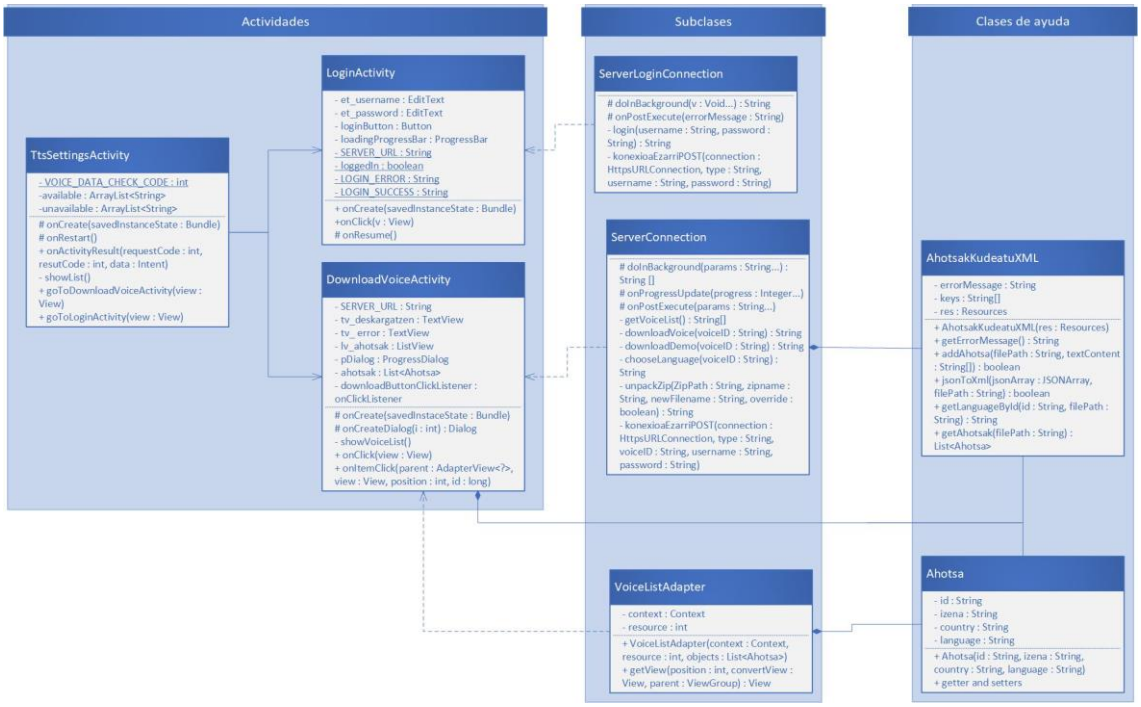

Ilustración 16: diagrama de clases de AhoTTS

8.2.1.1 TtsSettingsActivity

La primera pantalla es la correspondiente a la clase "TtsSettingsActivity" mostrada en la Ilustración 17 y contiene 2 botones:

- **Iniciar sesión.** Al pulsar este botón se ejecuta la función "goToLoginActivity" la cual lleva a la pantalla de "LoginActivity"
- **Descargar voz.** Al pulsar este botón se ejecuta la función "goToDownloadVoiceActivity" la cual lleva a la pantalla de "DownloadVoiceActivity"

Además de los botones se pueden ver las voces instaladas en el momento las cuales se generan con la función "showList". Esta función se ejecuta dentro de "onResume" el cual es un método especial que se lanza después de la creación de la actividad, después de volver a la pantalla si se había dejado en segundo plano…

Ilustración 17: pantalla "TtsSettingsActivity"

8.2.1.2 LoginActivity

Siguiendo con el apartado de inicio de sesión, en esta actividad se implementa la validación de usuario. Para ello la interfaz de usuario dispone de dos casillas para rellenar el usuario y contraseña, además de un botón el cual lanza la función "onClick" que a su vez genera un objeto de su subclase "ServerLoginConnection".

ServerLoginConnection

Al crear y ejecutar el objeto de esta clase, se inicia el proceso de conectar con el servidor y validar el usuario. Esta clase se ejecuta en segundo plano permitiendo al usuario seguir interactuando con la aplicación mientras cumple con su cometido.

Lo primero que hace es lanzar el método "doInBackground" dentro del cual se pueden programar las acciones que se desean ejecutar en segundo plano. En el caso de la aplicación se ejecuta el método "login" que se encarga de comunicarse con el servidor, enviarle la información mediante "konexioaEzarriPOST" y recibirla.

Al terminar, se dispara el método "onPostExecute" el cual analiza si la validación ha concluido de forma satisfactoria o no. En caso de ser correcta la validación, el botón pasa a ser "cerrar sesión" que como su nombre bien indica, permite cerrar la sesión iniciada, y se bloquean las entradas de texto. Por el contrario, si el resultado es erróneo, simplemente se visualizará un mensaje de error. Ambos casos se pueden observar en las siguientes ilustraciones.

Ilustración 18: pantalla validación de usuario correcto

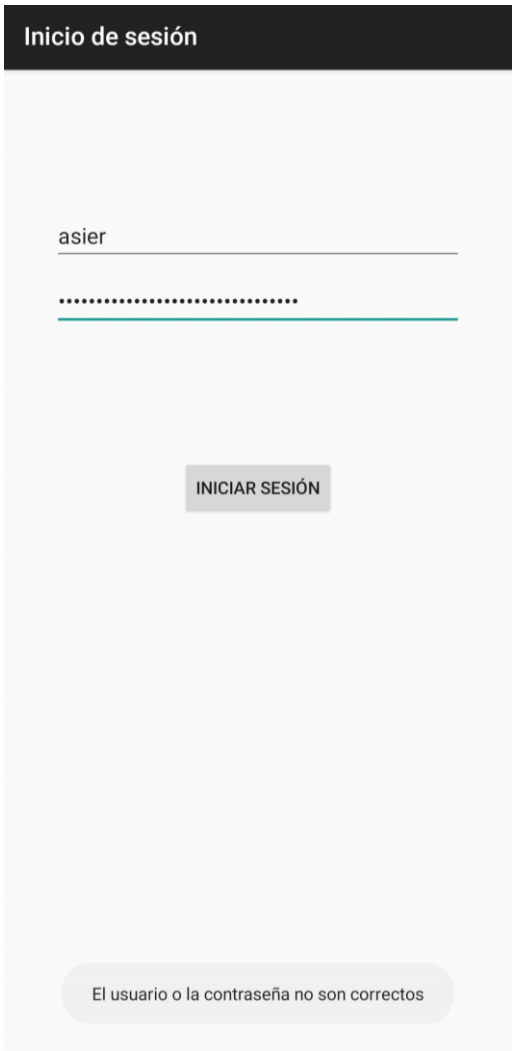

Ilustración 19: pantalla validación de usuario erróneo

Por último, la función "onResume" se encarga de analizar el estado de la sesión siempre que se entra en la actividad para modificar la pantalla al estado de la Ilustración 18 o la Ilustración 19.

8.2.1.3 DownloadVoiceActivity

La actividad "DownloadVoiceActivity" es la que contiene la gestión del resto de servicios: la obtención del catálogo, la descarga de voces y la descarga de una demo. Esas tareas se realizan principalmente por la subclase "ServerConnection" la cual, al igual que en la actividad de inicio de sesión, trabaja en segundo plano.

Además del método ejecutado en todas las actividades al ser creadas ("onCreate"), esta clase tiene los siguientes cuatro métodos:

- **onCreateDialog.** Sirve para mostrar un mensaje de diálogo que expresa el porcentaje de descarga completado tal y como se muestra en la siguiente ilustración.

Ilustración 20: pantalla diálogo de descarga

- **showVoiceList.** Este método muestra en pantalla el catálogo de voces con la configuración apreciable en la Ilustración 21. Para poder utilizar esa configuración personalizada se usa la subclase "VoiceListAdapter" de la cual se hablará posteriormente.

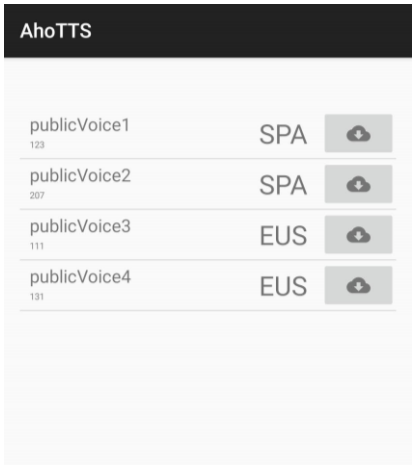

Ilustración 21: pantalla catálogo de voces

- **onClick.** Es el método que se ejecuta al pulsar el botón a la derecha de cada elemento de la lista. Al hacerlos se inicia el proceso de descarga de la voz seleccionada a través de la clase "ServerConnection"
- **onItemClick.** Es el método que se ejecuta al pulsar un elemento de la lista, el cual inicia la descarga y reproducción de la demo de la voz seleccionada.

Antes de explicar el funcionamiento de esta subclase, se hablará de las dos clases de apoyo de las que se nutren la actividad y sus las dos clases que incorpora.

Ahotsa

Esta clase representa los datos que aporta el catálogo de voces, es decir, sus atributos corresponden a la información que se obtiene de cada voz de la lista. De esta forma, se pueden utilizar objetos de esta clase permitiendo una fácil manipulación de los datos.

AhotsakKudeatuXML

En apartados previos se ha mencionado que el catálogo se almacena en el servidor con formato XML y después es enviado en JSON. Por tanto, al recibirlo el cliente en JSON es interesante volver a transformarlo a XML para guardarlo.

Esta clase sirve para la gestión de la información de esa lista de voces. Por un lado, el método "jsonToXml" recibe la lista en formato JSON, la transforma a XML y la guarda en un fichero valiéndose del método "addAhotsa". Este último hace la función de añadir al final del documento la voz pasada como parámetro. El fichero se guarda en el directorio raíz de la aplicación[7] (diferente en los distintos dispositivos Android).

Por otro lado, "getLanguageById" sirve para obtener el lenguaje de la voz cuyo identificador se mande. Por último, "getAhotsak" y "getErrorMessage" se utilizan para obtener la lista de voces completa y los posibles errores durante las operaciones respectivamente.

ServerConnection

Al igual que en la clase "ServerLoginConnection", "ServerConnection" dispone de dos métodos en torno a los cuales gira u funcionamiento.

Por un lado, está "doInBackground" el cual se ejecuta en segundo plano y en ella se realizan las comunicaciones con el servidor. Al iniciar este método en función de los parámetros que reciba, realizará la función de descarga del catálogo ("getVoiceList"), la función de descarga de voz ("downloadVoice") o la descarga de la demo ("downloadDemo"). Todas ellas utilizan la función de "konexioaEzarriPOST" para iniciar la comunicación con el servidor y mandarle la información pertinente.

- **getVoiceList.** En este método se implementa el establecimiento de la conexión y se recibe el catálogo de voces en formato JSON. Una vez obtenido, por medio de la clase "AhotsakKudeatuXML" se almacena de nuevo en formato XML.

[7] El directorio raíz de la aplicación de *AhoTTS* se obtiene con el método "getFilesDir().getPath()" que ofrece android

- **downloadVoiceList.** Realiza el establecimiento de la conexión y solicita la voz cuyo identificador se le pasa como parámetro. Una vez recibida, se almacena junto a las voces instaladas en el directorio raíz de la aplicación seguido de "data_tts/voices" y, puesto que está en formato ZIP, se ejecuta la función "unpackZip" que descomprime el fichero y lo mueve al destino elegido. En este caso, el destino es el lugar donde se encuentra la voz del idioma correspondiente el cual se obtiene mediante el método "chooseLanguage" que devuelve directamente la ruta. Esta ruta es la previamente mencionada y en función del idioma se añade "aholab_es" o "aholab_eu" (castellano y euskera respectivamente).
- **downloadDemo.** Es muy similar a los métodos anteriores, lo primero, establece la conexión y solicita la demo de una voz concreta. Una vez recibida la demo se almacena en el dispositivo y se cierra la conexión. El almacenamiento de realiza en el mismo directorio que el catálogo de voces, en el directorio raíz de la aplicación.

Por otro lado, "onPostExecute" realiza una pequeña labor después de cada una de las diferentes operaciones:

- Después de descargar el catálogo, se utiliza la clase "VoiceListAdapter" para preparar la lista a visualizar y se muestra la información de las voces en pantalla.
- Al terminar de descargar una voz simplemente se muestra un mensaje indicando la correcta descarga de la voz.
- Cuando la demo se ha obtenido correctamente se reproduce.

En todos los casos previamente se recogen posibles mensajes de errores y se muestran de haberlos.

VoiceListAdapter

Es una clase que se utiliza para personalizar una lista de elementos como la del catálogo de voces. Para la aplicación se ha decidido utilizar el formato de la Ilustración 22 en el cual se pueden ver a la izquierda el nombre de la voz y su identificador, y a la derecha el idioma al que pertenece y el botón de descarga.

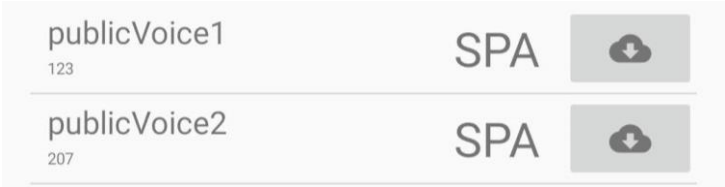

Ilustración 22: formato de lista en pantalla

8.2.2 Servidor

La aplicación principal del servidor es el fichero "server.php" el cual gestiona todas las comunicaciones recibidas. Además del código principal en el fichero se incluyen otros programas de PHP y una función que realiza el envío de la información del fichero cuya ruta

recibe: “sendVoice”. Todos los ficheros de PHP se encuentran en el directorio raíz web[8] dentro de la carpeta php.

La estructura se basa principalmente en el parámetro “tipo” que se incluye dentro de todos los mensajes de las distintas operaciones. Como se ha mencionado previamente, todas las comunicaciones utilizan los mensajes POST de HTTPS y en caso de recibir uno que no emplee ese tipo se devolverá un aviso de error.

Todos los mensajes enviados al servidor mantienen el mismo esquema. Dentro del cuerpo del mensaje POST, se envían los parámetros necesarios para la solicitud en formato clave-valor utilizando JSON.

En función del valor de la clave “tipo” recibido se realiza una operación u otra, en la siguiente tabla se resumen las distintas opciones:

Valor	Operación
0	Validación de usuario
1	Envío del catálogo de voces
2	Envío de voz
3	Envío de demo

Tabla 5: resumen operaciones en función del valor "tipo"

Al inicio de la recepción de cualquier mensaje POST, el programa recoge los parámetros de usuario y contraseña, y los valida con el servicio de DRUPAL. Para ello se utiliza la función “user_authenticate” que ofrece su API. El resultado de la operación es el identificador de ese usuario si los datos son correctos o “FALSE” en caso de error.

A continuación, se describen las 4 operaciones y las funciones que se utilizan en ellas.

8.2.2.1 Validación de usuario

Al recibir una solicitud de validación de usuario, puesto que el cliente ya se ha verificado al recibir el mensaje, se comprueba si el resultado ha sido positivo y se manda:

- En caso de validación correcta: “SUCCESS”
- En caso de validación errónea: “ERROR”

El formato del mensaje enviado al servidor es el siguiente:

[8] El directorio raíz web es “/var/www/users/asier”

HTTPS body (POST)	
Clave	**Valor**
tipo	0
username	<nombre de usuario>
password	<contraseña>

Tabla 6: parámetros mensaje al servidor - validación de usuario

8.2.2.2 Envío del catálogo de voces

Cuando llega la petición del catálogo el programa carga la lista de voces públicas en un Array. Dicha lista se encuentra en el directorio raíz web. Una vez hecho, si la validación de usuario ha sido correcta, ejecuta la función "getUserVoiceList" la cual devuelve el catálogo de voces privadas del cliente cuyo identificador se haya pasado. Por último, se añade la lista al Array y se transforma a JSON para seguido enviarlo.

Como ya se ha mencionado anteriormente, todas las voces tienen las claves de: "id", "izena", "country" y "language" en ese orden. A continuación, se muestra lo que correspondería al catálogo en formato XML con dos voces.

```
<voiceList>
      <voice>
            <id>123</id>
            <izena>publicVoice1</izena>
            <country>ESP</country>
            <language>spa</language>
      </voice>
      <voice>
            <id>207</id>
            <izena>publicVoice2</izena>
            <country>ESP</country>
            <language>spa</language>
      </voice>
</voiceList>
```

El formato del mensaje enviado al servidor es el siguiente:

HTTPS body (POST)	
Clave	**Valor**
tipo	1
(username)[9]	<nombre de usuario>
(password)	<contraseña>

Tabla 7: parámetros mensaje al servidor - obtención del catálogo de voces

8.2.2.3 Envío de voz

En esta operación, el primer paso es ejecutar la función "prepareVoice" si el usuario ha sido validado de forma correcta. Dicha función recibe los parámetros del identificador de usuario y de la voz. Con ello compara la voz solicitada con las del catálogo privado del usuario, y, en caso de encontrarse en él, se prepara para el envío:

- Se copia la voz a una carpeta temporal para la descarga. Esta carpeta se encuentra en el directorio raíz de la aplicación[10] seguido de "/ahotsak/tmp".
- Se comprime la voz a formato zip para su envío
- Devuelve la ruta de la voz

Una vez recibida la ruta, se ejecuta la función "sendVoice" que se encarga de mandar la voz.

Si la voz no se encuentra en el catálogo privado, se genera la ruta de la voz pública y se ejecuta la función "sendVoice". Dicha ruta es el directorio raíz de la aplicación, seguido de "/ahotsak" y el idioma correspondiente. Es decir, si la voz buscada es la 123 en castellano, la ruta sería: directorio base + "/ahotsak/spa/123.zip".

El formato del mensaje enviado al servidor es el siguiente:

HTTPS body (POST)	
Clave	**Valor**
tipo	2
vozID	<identificador de la voz>
(username)	<nombre de usuario>
(password)	<contraseña>

Tabla 8: parámetros mensaje al servidor - descarga de voz

[9] Los parámetros entre paréntesis indican que no son obligatorios
[10] El directorio raíz de la aplicación es "/home/aholab/asier"

8.2.2.4 Envío de demo

Este servicio solo esta implementado para las voces públicas y por lo tanto no se realiza ninguna comprobación del usuario. En función del identificador de la voz, se genera la ruta de la demo y se utiliza la función "sendVoice" para mandarla. La ruta de la demo es, directorio raíz de la aplicación + "/ahotsak/demo" + idioma de la voz.

El formato del mensaje enviado al servidor es el siguiente:

HTTPS body (POST)	
Clave	**Valor**
tipo	3
vozID	<identificador de la voz>

Tabla 9parámetros mensaje al servidor - descarga de demo

METODOLOGÍA

9 Planificación

El desarrollo de este trabajo se ha dividido en distintas etapas a las que se les ha dado el nombre de paquetes de trabajo (PT). Dentro de cada paquete hay varias tareas (T). Durante este apartado se analizan las distintas fases del proyecto, teniendo en cuenta el tiempo de realización de cada una y los recursos humanos usados.

9.1 Grupo de trabajo y recursos materiales

El grupo de trabajo de este proyecto se compone de dos personas únicamente:

Código	Nombre	Cargo	Rol
IS	Ibon Saratxaga	Ingeniero Senior	Supervisión del proyecto
IJ	Asier Arpón	Ingeniero Junior	Realización del proyecto

Tabla 10: grupo de trabajo del proyecto

Aparte de eso, a continuación, se nombran los recursos materiales utilizados durante el trabajo.

Código	Material	Cantidad
PCP	Ordenador personal	1
PCS	Servidor	1
TLF	Teléfono móvil	1

Tabla 11: materiales del proyecto

9.2 Descripción de tareas

PT1 - Seguimiento del proyecto y documentación

Este paquete de trabajo se realiza durante el desarrollo de todo el proyecto, ya que consta del seguimiento del mismo y de su documentación.

- **T1.1 Seguimiento del proyecto.** En esta tarea se efectúa un seguimiento del proyecto a lo largo de todo el tiempo que dure. Además, se realizan múltiples reuniones para corregir los errores que surgen y hacer una mejor supervisión.
- **T1.2 Documentación.** Es la redacción de todas las tareas realizadas durante el proyecto y del documento final a presentar.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	18h
IJ	90h

Tabla 12: uso de recursos PT1

PT2 - Definición del proyecto

La definición del proyecto es la primera parte, dónde se deciden algunos puntos como los objetivos.

- **T2.1 Definir objetivos y especificaciones.** En esta tarea se decide hasta dónde se quiere llegar en el proyecto y cuáles son las especificaciones que se implementarán.
- **T2.2 Análisis de alternativas.** Ciertos aspectos del proyecto en los que hay duda sobre su método de realización, tecnología de implementación, etc. Se hará un análisis para decidir cuál es la mejor opción.
- **T2.3 Análisis de riesgos.** En esta tarea se evalúan los posibles riesgos en la realización del proyecto. Después se observa cuáles son los más peligrosos y se proponen medidas para minimizarlos.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	2h
IJ	20h

Tabla 13: uso de recursos PT2

PT3 - Diseño de módulos de desarrollo

Una vez definidos los distintos aspectos se pasa a diseñar los módulos que se desarrollan en el proyecto.

- **T3.1 Diseño de módulo de descarga de voces públicas.** Se diseña el módulo de descarga de las voces públicas.
- **T3.2 Diseño de módulo de descarga de voces privadas.** Se diseña el módulo de descarga de voces privadas.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	2,4h
IJ	24h

Tabla 14: uso de recursos PT3

PT4 - Desarrollo de módulo de descarga de voces públicas

Después de diseñar los módulos comienza el desarrollo de la aplicación.

- **T4.1 Establecimiento de la comunicación.** Se desarrolla el establecimiento de la comunicación entre el cliente y el servidor.
- **T4.2 Obtención de la lista de voces públicas.** Se desarrolla el programa que permite descargar el catálogo de voces desde el servidor.
- **T4.3 Visualización de lista.** Se implementa el programa que visualiza la lista en pantalla.
- **T4.4 Descarga de voz pública.** Se implementa el servicio de descarga de voces públicas.
- **T4.5 Instalación de voz.** Después de la descarga de la voz se implementa la instalación de la misma.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	7,2h
IJ	72h

Tabla 15: uso de recursos PT4

PT5 - Pruebas unitarias y corrección de errores I

En este punto se ejecutan pruebas del programa y se corrigen errores.

- **T5.1 Pruebas unitarias.** Se realizan pruebas sobre el programa realizado para buscar posibles fallos en el desarrollo.
- **T5.2 Corrección de errores.** Se arreglan los errores cometidos en el desarrollo del módulo previo.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	2,4h
IJ	24h

Tabla 16: uso de recursos PT5

PT6 - Desarrollo de módulo de descarga de voces privadas

Una vez solventados los errores del módulo anterior comienza el desarrollo de la descarga de voces privadas.

- **T6.1 Validación de usuario.** Se desarrolla la funcionalidad de validar un usuario.
- **T6.2 Modificación de obtención de lista de voces.** Se modifica el desarrollo anterior de la tarea de obtención de lista de voces públicas para añadirle la posibilidad de descargar el catálogo privado también.
- **T6.3 Modificación de descarga de voz.** Se modifica el programa anterior para implementar la descarga de voces privadas.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	4h
IJ	40h

Tabla 17: uso de recursos PT6

PT7 - Pruebas unitarias y corrección de errores II

En este punto se ejecutan pruebas del programa y se corrigen errores.

- **T7.1 Pruebas unitarias.** Se realizan pruebas sobre el programa realizado para buscar posibles fallos en el desarrollo.
- **T7.2 Corrección de errores.** Se arreglan los errores cometidos en el desarrollo del módulo previo.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	2,4h
IJ	24h

Tabla 18: uso de recursos PT7

PT8 - Pruebas de integración y corrección de errores

Una vez realizadas las pruebas unitarias y solucionados los posibles errores comienzan las pruebas de integración.

- **T8.1 Pruebas de integración.** Se realizan pruebas en las que se comprueban la interoperabilidad de los diferentes módulos.
- **T8.2 Corrección de errores.** Se corrigen los posibles fallos que aparezcan en la tarea anterior.

Los recursos humanos utilizados durante este paquete de trabajo se resumen en la siguiente tabla.

R. Humano	Horas
IS	2,4h
IJ	24h

Tabla 19: uso de recursos PT8

10 Diagrama de Gantt

A continuación, se muestran dos diagramas de Gantt de las tareas mencionadas en el apartado anterior. El primero, la Ilustración 23, muestra las tareas de seguimiento del proyecto y documentación del primer paquete de trabajo, junto con el resto de paquetes de trabajo. El segundo, por el contrario, muestras todas las tareas del proyecto a excepción de las del primer paquete.

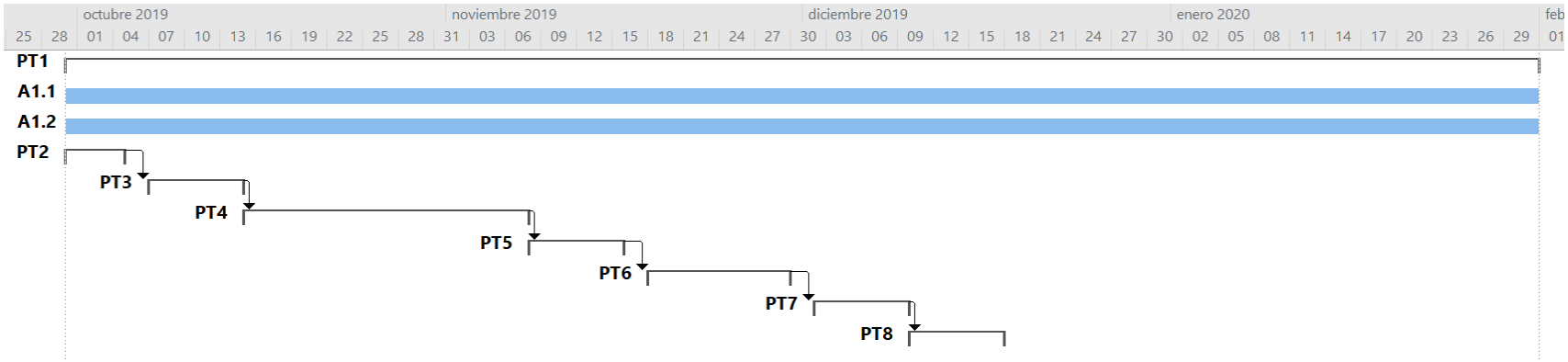

Ilustración 23: diagrama de Gantt I

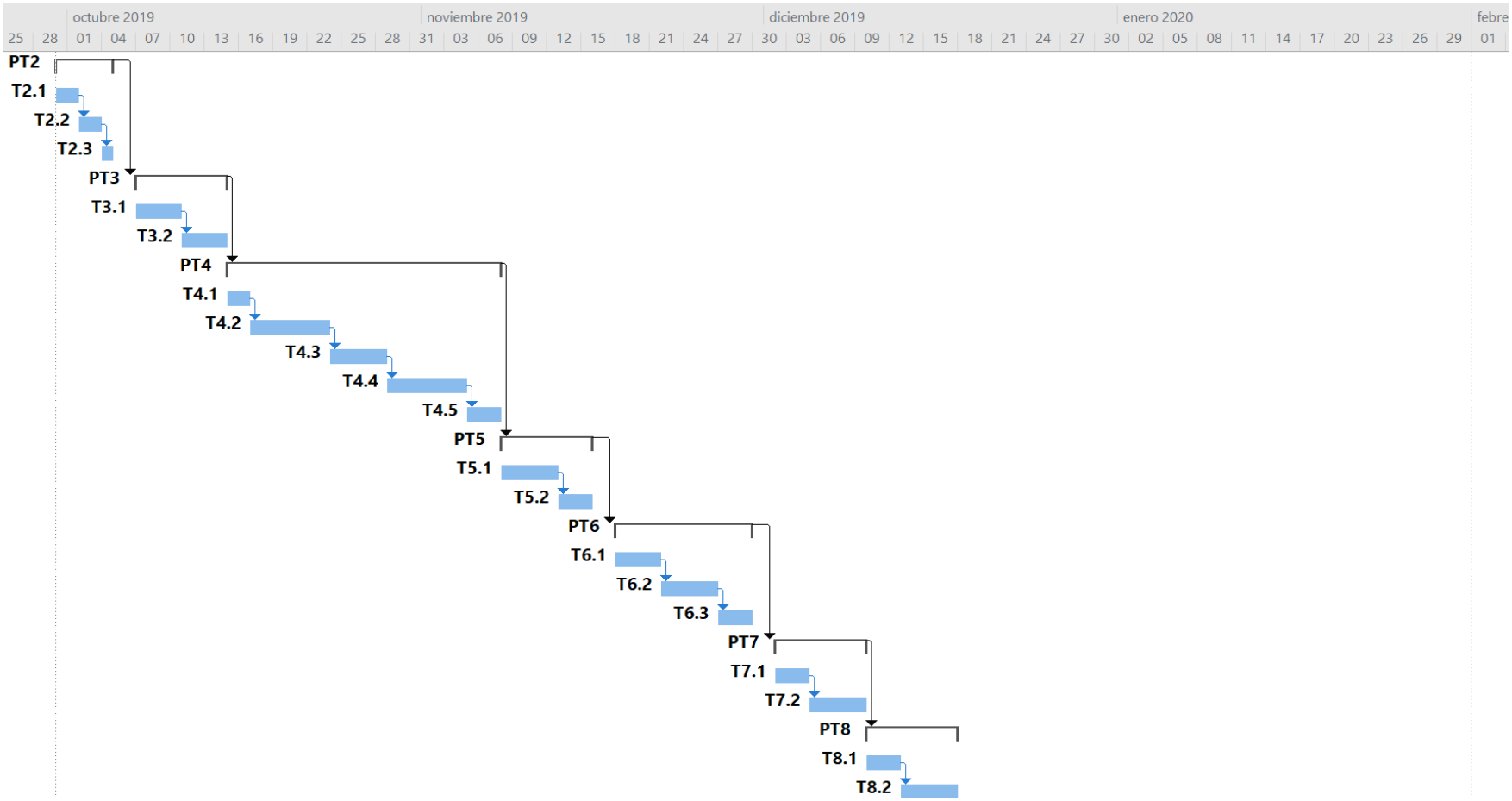

Ilustración 24: diagrama de Gantt II

ASPECTOS ECONÓMICOS

11 Presupuesto

En este apartado se tratan los aspectos económicos del proyecto. Por un lado, se necesitan recursos humanos que, como se ha mencionado previamente, se componen de un ingeniero senior y un ingeniero junior. Por otro lado, están los recursos materiales que se constituyen por dos ordenadores, uno personal y otro el servidor, y un teléfono Android.

En la tabla a continuación, se calcula el coste que tiene el personal en función del dinero por horas que cobran y el tiempo trabajado.

Código	Nombre	Tasa	Cantidad de horas	Total
IS	Ibon Saratxaga	50€/h	40,8h	2.040€
IJ	Asier Arpón	20€/h	318h	6.360€
			Total	**8.400€**

Tabla 20: costes recursos humanos

La siguiente tabla se corresponde a la amortización de los materiales usados en el proyecto en función de su vida útil y el tiempo de utilización.

Código	Nombre	Coste	Vida útil	Cantidad de tiempo	Total
PCP	Ordenador personal	750€	4 años	4 meses	62,5€
PCS	Servidor	8000€	8 años	4 meses	333,33€
TLF	Teléfono móvil	400€	3 años	1 mes	11,11€
				Total	**406,94€**

Tabla 21: costes recursos materiales

Por último, está la tabla que resume todos los gastos y tiene en cuenta los costes indirectos como un 10% de los indirectos.

Concepto	Total
Recursos humanos	8.400€
Recursos materiales	406,94€
Costes directos	8.806,94€
Costes indirectos (10%)	880,69€
Total	**9.687,63€**

Tabla 22: costes totales

Conclusiones

Los objetivos de este proyecto comenzaban con unificar los servicios de *AhoTTS y AhoMyTTS* brindando a la aplicación en Android la posibilidad de descargar un catálogo de voces públicas y escoger una para su descarga e instalación.

Por un lado, mediante la aplicación servidor creada en este trabajo es posible la distribución de voces almacenadas. Por otro lado, *AhoTTS* ahora implementa las funcionalidades necesarias para obtener el catálogo mencionado y posteriormente descargar la voz escogida. Gracias a dichos servicios se puede decir que se ha cumplido el primer objetivo propuesto.

Además del primer objetivo, se proponía un segundo que pretendía añadir la validación de un usuario en *AhoMyTTS* para así descargar además de las voces públicas, las de uso privado guardadas por el cliente.

En lo referente a este propósito, se podría decir que se ha conseguido debido a las modificaciones añadidas en los servicios de obtención de catálogo y descarga de voces tanto en el servidor como en el cliente. Estas alteraciones permiten la manipulación de la información relativa al usuario y contraseña, junto con la implementación de un servicio de validación de usuario cumpliendo con el segundo objetivo.

Al inicio del documento, se hablaba de aquellas personas que por diversos motivos como enfermedades carecían de la posibilidad de comunicarse mediante el uso de la voz. Con este proyecto se ha conseguido ofrecerles la posibilidad de usar una gran variedad de voces junto con el TTS de la aplicación móvil. Y no solo eso, el servicio está automatizado, lo que elimina el coste humano previo.

Así mismo, se han desarrollado ambas aplicaciones de forma independiente y utilizando estándares que permiten el desarrollo de otros programas cliente para distintas plataformas como iOS.

En conclusión, se han cumplido los objetivos planteados en el apartado 4, tanto los principales como los secundarios, cubriendo las necesidades observadas.

Bibliografía

[1] Google, "Idiomas y voces compatibles | Documentación de Cloud Text-to-Speech." https://cloud.google.com/text-to-speech/docs/voices?hl=es (accessed Jul. 16, 2020).

[2] "Samsung text-to-speech engine 3.0.04.4 para Android - Descargar." https://samsung-text-to-speech-engine.uptodown.com/android (accessed Jul. 17, 2020).

[3] "PHP: simplexml_load_file - Manual." https://www.php.net/manual/en/function.simplexml-load-file (accessed Jul. 21, 2020).

[4] "API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos." https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos (accessed Jul. 21, 2020).

[5] "user_authenticate | user.module | Drupal 7.x | Drupal API." https://api.drupal.org/api/drupal/modules%21user%21user.module/function/user_authenticate/7.x (accessed Jul. 21, 2020).