

Desarrollo de un front-end para DuoCode

Julián F. Calleja da Silva

Johana Gabriela Ferreira Yagua

José Carlos Valera Villalba

Grado en Ingeniería Informática

Facultad de Informática

Departamento de Sistemas Informáticos y Computación



Trabajo fin de grado

Dirigida por

Enrique Martín Martín

Adrián Riesco

Madrid, 2015

Abstract

Resumen

DuoCode bla b

Índice

1. Introducción con ejemplos	1
2. Hola	1
2.1. Introducción	1
3. Introducción	2
3.1. Revisión del estado del arte	2
3.2. Influencias tecnológicas	3
3.3. Propuestas y objetivos	4
4. Requisitos y base de datos	6
4.1. Requisitos	6
4.2. Base de datos	6
5. Manual de instalación	12
5.1. Instalación paso a paso	12
A. Especificación de requisitos	18
A.1. localhost/duocode/rest/temas	18
A.2. localhost/duocode/rest/temas/idTema	18
A.3. localhost/duocode/rest/lecciones/	20
A.4. localhost/duocode/rest/lecciones/idLeccion	20
A.5. localhost/duocode/rest/ejercicios	22
A.6. localhost/duocode/rest/ejercicios/idEjercicio	22
A.7. localhost/duocode/rest/enunciados	23
A.8. localhost/duocode/rest/enunciados/idEnunciado	24
A.9. localhost/duocode/rest/lenguajes/	25
A.10.localhost/duocode/rest/candidatos/	25
A.11.localhost/duocode/rest/candidatos/idCandidato	26
A.12.localhost/duocode/rest/usuarios/	27
A.13.localhost/duocode/rest/usuarios/idUsuario	27
A.14.localhost/duocode/rest/envios	28

1. Introducción con ejemplos

Hola **negrita** *cursiva*

Tres

- Primera cosa
- Segunda cosa

1.

2.

Cosa 1. Esto que vamos a comentar ahora es algo muy importante porque blabla Esto que vamos a comentar ahora es algo muy importante porque blabla

Cosa2

Ejemplo 1

hola	<i>hola</i>	<i>hola</i>
<i>adios</i>	<i>adios</i>	<i>adios</i>

2. Hola

2.1. Introducción

Para la sección 2.1 usaremos el libro [8]

```
if (condicion){  
    int x = 0;  
}  
else{  
    int x = 1;  
}
```

3. Introducción

Este proyecto empezó con el objetivo de permitir a los alumnos que lo desarrollan profundizar en tecnologías y paradigmas que no se aprenden a lo largo del grado o que se dan como meras referencias. Todo ello sin olvidarnos de los valiosos conocimientos ya afianzados que seguro serán de utilidad. Nos planteamos realizar una *front-end* para un proyecto de aprendizaje colaborativo de lenguajes de programación en base a otros que ya sabes, llamado DuoCode. A continuación pasaremos a explicar el estado actual del sector, luego detallaremos las tecnologías que nos han influenciado y usaremos, y finalmente definiremos los objetivos del proyecto.

3.1. Revisión del estado del arte

En la actualidad hay numerosas maneras de aprender online. Para empezar, existen recursos típicos donde el que quiere aprender lee y realiza poca interacción. El líder por antonomasia de esta modalidad es Wikipedia [7], una enciclopedia libre y editada colaborativamente en múltiples idiomas. Ante dudas concretas hay sitios de preguntas y respuestas como Stack Overflow [6], donde programadores se ayudan mutuamente en base a cuestiones concretas. Asimismo muchas universidades usan plataformas digitales como Moodle [4], que permiten publicar apuntes de las diferentes asignaturas. También existen cursos online, los llamados MOOC (acrónimo en inglés de Massive Open Online Course) donde puedes aprender a distancia materias. Estos requieren que se suba material como vídeos con explicaciones detalladas. Suelen estar enfocados a la obtención de algún tipo de diploma o certificación. Requieren un esfuerzo constante y duradero por parte del profesorado, que tiene que actualizar el material académico disponible y corregir los distintos trabajos de los alumnos.

Otra manera de enfocar la enseñanza, sobre todo cuando quieres ampliar algo sobre la que ya tienes una base, es a base de ejercicios. Durante bastantes años se han usado los *jueces online* o *correctores automáticos*. Hay un análisis general sobre este tema, donde se estudia la viabilidad de un proyecto colaborativo para enseñar lenguajes de programación [8]. En él se explica como normalmente este tipo de sistemas, aplicados a lenguajes de programación, se basan en la ejecución de unos casos de prueba para comprobar la corrección de los programas del usuario. La desventaja de estos sistemas es que exige que los instructores desarrollen previamente estas pruebas, tarea poco grata.

Saliéndonos del mundo de la programación, hay otros ejemplos, como Duolingo [3] que permiten aprender un idioma a partir de otro que ya sabes previamente. Sus características más relevantes son:

- Plantear el aprendizaje como un juego, haciendo que el usuarios gane puntos y experiencia según va avanzando. Además, incluye vidas, que hacen centrar la atención en la tarea presente para no tener que reiniciar el nivel.
- Guarda información sobre los fallos del usuario para intentar repetir esas preguntas y que aprenda los concepto de manera definitiva.
- Incluye prácticas con tiempo y la posibilidad de certificar el nivel del idioma mediante tests online.

Hay otras alternativas como Bussu [2], donde los usuarios hacen simultáneamente de

alumnos y profesores e interactúan entre ellos en una red social con el objetivo de aprender otro idioma.

3.2. Influencias tecnológicas

Antes de definir cómo llevar a la práctica este proyecto, quisimos ver desde alto nivel qué tipo de sistema queríamos desarrollar. Todas las asignaturas que nombraremos a continuación son las correspondientes al plan de estudios de la universidad [5]. Este plan de estudios sigue las pautas del Acuerdo del Consejo de Universidades publicado en el B.O.E [1].

Durante la carrera hemos aprendido, entre otras cosas, lenguajes de programación que nos permiten hacer software ejecutable en ordenadores de sobremesa, en asignaturas como Fundamentos de la Programación, o Tecnologías de la Programación. Aunque éste podría haber sido un enfoque válido, no era el que queríamos seguir. Nos parece que podríamos hacer algo mucho más rápido de probar y usar por primera vez, sin la necesidad de instalar pesado software adicional.

En otras asignaturas como Software Corporativo aprendimos a montar y configurar un sistemas de gestión de contenidos, también llamados CMS (acrónimo en inglés de Content Management System) y con ello montamos una Web. Este enfoque presenta más ventajas, pues permite a cualquiera que tenga un navegador de Internet acceder a nuestra plataforma. Sin embargo, los CMS actuales no permiten hacer cosas tan concretas y específicas como lo que queríamos hacer. Además, dejaríamos sin usar la mayoría de las características de estos y pensamos que no usaríamos todo el potencial que ofrecen estas plataformas.

También cursamos Aplicaciones Web, donde aprendimos a desarrollar una Web desde el principio. Realizar un desarrollo de este tipo nos parecía sumamente interesante, pues nos permitiría un alto grado de flexibilidad con la manera en que queremos realizar el proyecto, y permitiría a los potenciales usuarios probar y usar la plataforma de manera rápida. Además el uso de llamadas asíncronas nos puede permitir hacer la Web plenamente interactiva y fluida ante las acciones del usuario. Por esto decidimos utilizar HTML, CSS y JavaScript en nuestro proyecto. En esta asignatura también nos apoyábamos en conocimientos adquiridos en otras como Bases de Datos y Ampliación de Base de Datos, donde aprendimos a diseñar e implementar bases de datos relacionales y a consumirlas desde las aplicaciones Su uso permite establecer interconexiones (relaciones) entre los datos (guardados en las distintas tablas), y tiene como principal ventajas evitar duplicidades de los datos, y garantizar la integridad de los datos relacionados entre si. Con todos estos conocimientos podemos hacer un sitio Web interactivo y que guarde una gran variedad de datos. Por todo esto decidimos usar una base de datos de este tipo.

También sentíamos particular inclinación por hacer que el contenido estuviera disponible en dispositivos móviles, bien sea adaptando la Web o a través de una aplicación móvil. Algunos estábamos matriculados en el curso de realización de la asignatura en Programación de Aplicaciones para Dispositivos Móviles, y vimos una oportunidad para poner de manifiesto lo que se podría aprender en esa asignatura.

Por último investigamos la utilidad de implementar un servicio Web, una tecnología que utiliza un conjunto de protocolos y estándares para intercambiar datos entre aplicaciones. Lo más parecido que hemos dado a este enfoque es el uso de *sockets* en Ampliación de

Sistemas Operativos y Redes. Su uso permite abstraer y separar la manipulación de los datos de la representación de la interfaz del usuario, pudiendo incluso tener varias interfaces para la misma aplicación. Por todo ello decidimos usar este enfoque. Investigamos varias maneras de implementar el servicio Web:

- La llamada a procedimiento remoto (RPC) (del inglés, Remote Procedure Call) es un protocolo que permite a un ordenador ejecutar código de manera remota en otro. Vimos que hay diversas maneras de realizar un RPC, basándose en distintas especificaciones, siendo una de las más populares SOAP.
- SOAP es un acrónimo en inglés de Simple Object Access Protocol y es un protocolo que define como diferentes ordenadores pueden comunicarse a base del intercambio de mensajes XML. Sus principales desventajas son que se apoya en un descriptor de los servicios disponibles, que hay que actualizar con cada nueva funcionalidad añadida. Usa XML un lenguaje de marcado que ha perdido fuelle frente al más ligero JSON. Además no todos los lenguajes de programación ofrecen facilidades para el uso de este tipo de servicio Web.
- REST es un protocolo cliente/servidor sin estado que usa HTTP y los métodos de este protocolo (GET, POST, PUT, DELETE ...) para consultar y modificar los distintos recursos. Con frecuencia a este tipo de sistemas se les llama RESTful. Permiten mucha libertad a la hora de desarrollarlo, y como para consumir este tipo de servicio Web solo hace falta hacer peticiones HTTP es ampliamente soportado. Además permite ofrecer los datos en JSON. Por esto decidimos usar esta tecnología para nuestro proyecto.

3.3. Propuestas y objetivos

Gracias a la investigación de todo lo expuesto anteriormente, fijamos nuestras ideas. Nos decidimos a hacer un proyecto colaborativo para aprender lenguajes de programación, similar a Duolingo. Como iba a estar centrado en código de programación, decidimos llamarlo DuoCode, pues aprenderás lenguajes de programación nuevos a partir de otros que ya sabes.

Las principales características que queríamos que tuviese disponibles el usuario son:

- Poder seleccionar el idioma de programación que sabes y el que quieres aprender. El sistema te mostrará código en el lenguaje que dominas y te pedirá rellenarlo en el que no.
- Habrá una clara organización con código agrupado por temas.
- El usuario tendrá una puntuación que irá aumentando según avance. Además, los ejercicios estarán agrupados y tendrá una serie de vidas para completarlos. Esto favorecerá que el usuario se fidelice con la aplicación.
- Cuando el usuario falle un ejercicio pero no esté de acuerdo, podrá proponerlo como candidato a válido. Con la ayuda de la comunidad y de moderadores, estos se podrán incorporar como soluciones en un lenguaje de programación determinado.
- Se podrá guardar ejercicios favoritos para poder consultarlos de nuevo.

- Habrá una interacción con las redes sociales. Se podrá hacer login con alguna de ellas y compartir tus resultados en esta.

Basándonos en las conclusiones extraídas del apartado anterior, nuestro objetivos para el proyecto son:

- Desarrollar un sistema Web RESTful para el aprendizaje de la programación. Para ello nos planteamos los siguientes pasos:
 - Especificar los distintos recursos REST, y que información recibe y produce para cada uno de los métodos HTTP (GET, POST, PUT Y DELETE).
 - Diseñar una base de datos relacional para soportar el servicio, especificando las distintas tablas y sus relaciones.
 - Implementar y probar el servicio REST.
- Desarrollar una (o dos) interfaces gráficas que usen el servicio que acabamos de describir:
 - Diseñar la interfaz usando HTML/CSS, usando datos de prueba.
 - Conectar la interfaz con el servicio Web mediante llamadas asíncronas y JavaScript, de manera que use los datos reales proporcionados por el servicio REST. Al uso de esta técnica se le suele llamar AJAX (acrónimo en inglés de Asynchronous JavaScript And XML).
 - En caso de tener tiempo diseñar la interfaz de la aplicación para Android y realizar su implementación, o adaptar la Web para su visualización en dispositivos móviles.

4. Requisitos y base de datos

En esta sección hablaremos del método que utilizamos para abarcar la fase de desarrollo de los requisitos y de la creación de la base de datos.

4.1. Requisitos

Esta es la primera fase que abarcamos al empezar con el proyecto. Después de discutir las ideas sobre cómo queríamos que fuese el servicio web, empezamos a redactar los requisitos.

La redacción de los requisitos nos ha facilitado mucho la tarea de la implementación ya que así no teníamos que improvisar mientras escribíamos código y cuando lo necesitábamos recurríamos a ellos. Hemos hecho pequeñas modificaciones a medida que nos encontrábamos con algo distinto a lo que nos habíamos imaginado o que no habíamos tenido en cuenta antes de empezar a implementar.

Para hacerlos, los organizamos según los recursos que utilizaríamos (por ejemplo: temas, lecciones, ejercicios, usuarios...) y cada uno de ellos tiene un máximo de cuatro partes (GET, POST, PUT y DELETE) debido a que utilizamos una API REST y trabajamos con estos métodos de petición. Indicamos en todos qué se tiene que recibir por Payload y por Header (opcionalmente) y cuál será la respuesta que recibiremos.

La especificación de los requisitos se encuentra en el Apéndice A

4.2. Base de datos

A la vez que íbamos definiendo los requisitos, fuimos dándonos cuenta de las tablas y atributos que necesitaríamos en la base de datos y formamos el modelo entidad-relación (Figura 1) y el modelo relacional (Figura 2).

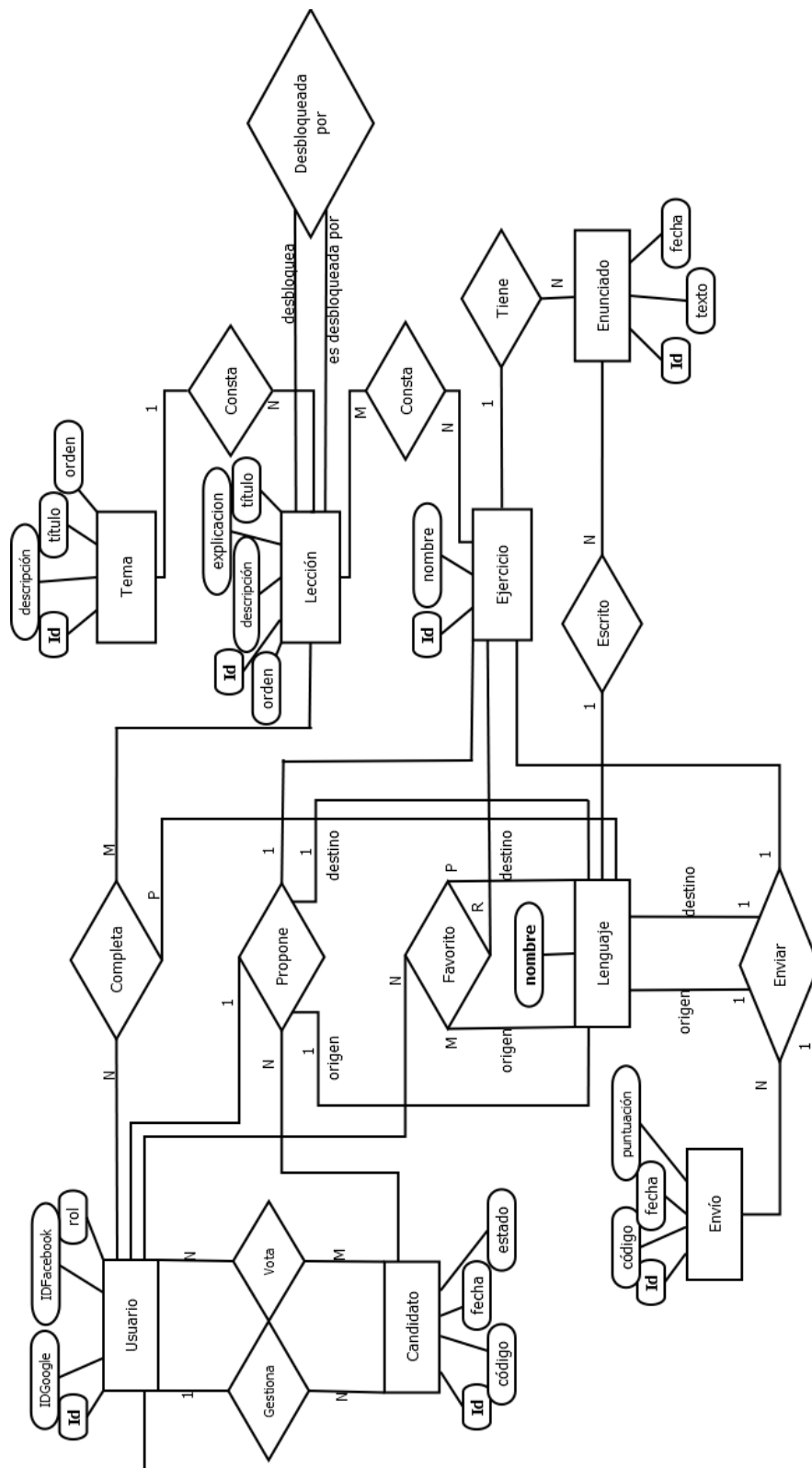


Figura 1: Modelo entidad-relacion

Estructura de las tablas

- Usuario: Tabla que contiene los datos de los usuarios. No hace falta guardar el nombre u otros datos personales ya que los obtenemos de la aplicación con la que haya iniciado sesión.

Nombre	Descripción
ID	ID que le asigna la aplicación al usuario.
IDGoogle	ID que le asigna Google. Es opcional ya que el usuario puede iniciar con Facebook.
IDFacebook	ID que le asigna Facebook. Es opcional ya que el usuario puede iniciar con Google.

- UsuarioVotaCandidato: Tabla que contiene todos los votos de los usuarios a los candidatos.

Nombre	Descripción
IDUsuario	ID del usuario que vota.
IDCandidato	ID del candidato al que se quiere votar.
Voto	Puede tomar los valores 0 o 1 dependiendo de si el voto es positivo (1) o negativo (0).

- Candidato: Tabla que contiene la información de los candidatos enviados por los usuarios de la aplicación.

Nombre	Descripción
ID	Identificador único para el candidato.
Código	Texto propuesto como posible solución del ejercicio.
Fecha	Fecha en la que se realiza la propuesta.
Estado	Puede ser No Gestionado (0), Aceptado (1) o Rechazado (-1).
GestionadoPor	ID del usuario administrador que finalmente acepta o rechaza el candidato. Inicialmente se encuentra vacío.
IDUsuario	ID del usuario que propone el candidato.
IDEjercicio	ID del ejercicio que resuelve dicho candidato.
LenguajeOrigen	Lenguaje en el que se encuentra el enunciado del ejercicio.
LenguajeDestino	Lenguaje en el que está la posible solución.

- Tema: Tabla que contiene la información de los temas, primera división para organizar los ejercicios según a qué van dedicados. Los temas se conforman de una colección de lecciones.

Nombre	Descripción
ID	ID propio del tema
Orden	Orden en el que queremos que aparezca el tema.
Título	Nombre que distingue a los temas.
Descripción	Texto breve que describe el tema.

- Lección: Tabla que contiene la información de las lecciones, segunda división para la organización. Las lecciones se conforman de una serie de ejercicios y están incluidas en temas.

Nombre	Descripción
ID	Identificador único para la lección.
Orden	Orden en el que queremos que aparezca la lección.
Título	Nombre que distingue a las lecciones.
Descripción	Texto breve que describe la lección.
Explicación	Texto largo que sirve como introducción a la lección.
IDTema	ID del tema al que pertenece esta lección.

- UsuarioCompletaLeccion: Tabla que contiene la relación de las lecciones con los usuarios que las han completado.

Nombre	Descripción
IDUsuario	ID del usuario que ha completado la lección.
IDLección	Lección que ha sido completada.
Lenguaje	Lenguaje en el que se ha completado dicha lección, ya que una lección puede ser completada por el mismo usuario en distintos lenguajes.

- DesbloqueadaPor: Tabla que contiene la dependencia entre lecciones. Hay lecciones a las que solo se puede acceder si se han superado otras anteriormente.

Nombre	Descripción
IDLección	ID de la lección a la que queremos asignar dependencia.
DesbloqueadaPor	Lección que debe ser superada para poder acceder a la mencionada anteriormente.
Lenguaje	Lenguaje en el que se ha completado dicha lección, ya que una lección puede ser completada por el mismo usuario en distintos lenguajes.

- Ejercicio: Tabla que contiene la información de los ejercicios existentes.

Nombre	Descripción
ID	Identificador asignado por la aplicación para cada ejercicio.
Nombre	Nombre con el que se diferenciará de los demás.

- LeccionConstaEjercicio: Tabla que contiene la relación donde se asignan los ejercicios a las correspondientes lecciones.

Nombre	Descripción
IDLección	ID de la lección a la que se añade un ejercicio.
IDEjercicio	ID del ejercicio añadido a la lección.

- Lenguaje: Tabla que contiene los lenguajes que podrán aprender los usuarios.

Nombre	Descripción
Nombre	Nombre del lenguaje.

- Enunciado: Tabla que contiene la información de los enunciados. Un enunciado es un código con un correspondiente lenguaje que forma parte de un ejercicio. Un ejercicio puede tener varios enunciados en varios idiomas.

Nombre	Descripción
ID	ID que identifica al enunciado.
Texto	Código, en lenguaje de origen, que tendrá que ser <i>traducido</i> .
Fecha	Fecha en la que fue añadido el enunciado.
IDEjercicio	ID del ejercicio al que corresponde este enunciado.
Lenguaje	Lenguaje en el que está escrito el código de dicho enunciado.

- Favorito: Tabla que contiene la relación entre los usuarios y los ejercicios favoritos de estos.

Nombre	Descripción
IDUsuario	ID del usuario que añade un ejercicio a favorito.
IDEjercicio	ID del ejercicio que ha sido marcado como favorito. <i>traducido</i>
LenguajeOrigen	Lenguaje del enunciado que ha sido marcado como favorito.
LenguajeDestino	Lenguaje que el usuario quería aprender al marcar este ejercicio como favorito.

- Envío: Tabla que contiene la relación entre los usuarios y todos los ejercicios que han realizado en la aplicación.

Nombre	Descripción
ID	ID correspondiente a cada envío.
Código	Código respuesta del usuario. <i>traducido</i>
Fecha	Fecha en la que se ha resuelto el ejercicio.
Puntuación	Puntuación obtenida.
IDUsuario	Usuario que ha realizado el envío.
IDEjercicio	Ejercicio resuelto.
LenguajeOrigen	Lenguaje inicial del enunciado.
LenguajeDestino	Lenguaje en el que el usuario ha enviado el ejercicio.

5. Manual de instalación

En este apartado se detallarán los pasos a seguir para tener instalado **DuoCode** en el sistema, con el fin de poder trabajar directamente con los ficheros fuentes y extender el proyecto. El software necesario para trabajar con el proyecto se especifica a continuación.

■ Requisitos generales

- Java 7
- Git
- Navegador Web - Google Chrome

■ Requisitos para el Servicio Web

- Xampp, Wampp o similar
- Tomcat 8.0
- NetBeans 8.0

■ Requisitos para el Front-End

- Soporte SSL en Xampp y Tomcat

■ Requisitos para la Aplicación móvil

- Node.Js 0.10.X
- Phonegap 5.0.0
- Eclipse Luna con plugin para Android

5.1. Instalación paso a paso

■ Java

En primer lugar necesitaremos tener instalada una versión de Java en nuestro sistema. La forma más sencilla es acceder a la web <http://www.java.com/es/> y descargar la versión adecuada para nuestro sistema operativo.

Si lo prefieres puedes usar la línea de comandos:

```
$ sudo apt-get install openjdk-7-jdk openjdk-7-jre
```

■ Git

Para poder descargar todos los ficheros fuentes del proyecto es necesario tener un cliente Git instalado y configurado.

Una opción es acceder a la web <http://git-scm.com/downloads/guis> y elegir el cliente que queramos para nuestro sistema operativo.

Si lo prefieres puedes usar la línea de comandos:

```
$ sudo apt-get install git
```

Una vez instalado git podremos clonar el proyecto DuoCode y obtener una copia en nuestro sistema. Si trabajas con un cliente gráfico simplemente tendrás que pinchar en el botón 'clone' y escribir la dirección del repositorio:

```
https://github.com/jucallej/DuoCode.git
```

También puedes clonar el proyecto directamente desde la línea de comandos:

```
$ git clone https://github.com/jucallej/DuoCode.git
```

■ Google Chrome

Cualquier navegador sería válido pero Chrome cuenta con un plugin (Advance Rest Client) para hacer pruebas con el servicio web Rest bastante intuitivo. Para instalar el navegador accedemos a la web de Chrome <https://www.google.es/chrome/> y pinchamos en el botón de descarga.

Si lo prefieres también puedes usar la línea de comandos:

```
$ sudo apt-get install google-chrome-stable
```

■ XAMPP - WAMP - MAMP o similar

Será necesario tener en local instalado un servidor como cualquiera de los nombrados anteriormente. Nos proporcionan una base de datos MySQL y un server Apache. Para instalarlo solo hay que acceder a la web "<https://www.apachefriends.org/index.html>" (en el caso de querer instalar XAMPP) y descargar la versión para nuestro sistema operativo.

Una vez instalado y funcionando accedemos a <http://localhost/phpmyadmin> para importar la base de datos. Creamos una Base de datos nueva con el nombre 'Duo-code', la seleccionamos e importamos el archivo 'Duocode.sql' para que nos cree las tablas y cargue la información del proyecto.

En la carpeta 'htdocs' que se encuentra dentro de la carpeta del servidor XAMPP es donde tendremos que poner la parte del front-end. Copiamos la carpeta 'duocode' que contiene el index.html y todos los scripts y la pegamos en 'htdocs'. Podemos probar el funcionamiento accediendo a la dirección <http://localhost/duocode>

■ Tomcat

Para el servicio web REST necesitaremos tener instalado Tomcat 8.0 o superior.

Accedemos a la web <http://tomcat.apache.org/download-80.cgi> (para la versión 8.0), descargamos la última versión para nuestro sistema operativo y lo descomprimos.

■ NetBeans

El IDE que se ha usado para desarrollar el proyecto es NetBeans 8.0 o superior por su facilidad para integrar el control de versiones Git y los servidores, se puede descargar desde la web <https://netbeans.org/downloads/> la versión para nuestro sistema operativo.

Una vez instalado importamos el proyecto descargado desde el Git y seleccionamos el servidor con el que queremos que funcione, en nuestro caso será el Tomcat descargado anteriormente.

Las librerías necesarias se importarán de manera automática una vez que hayamos cargado el proyecto en NetBeans.

■ Certificados SSL

Para que tanto el front-end como el servicio web funcionen con https es necesario activar SSL en los servidores Apache del XAMPP y el Tomcat descargado.

Tomcat lo podemos configurar gracias al archivo server.xml encontrado en /apache-tomcat-8.0.21/conf/server.xml Lo abrimos y dentro del elemento:

```
< Service name = 'Catalina' >
```

pegamos estas líneas de código (cambiando la ruta del proyecto):

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreType="PKCS12"
    keystoreFile="{Ruta al repositorio}\DuoCode\duocode\certificados\mycert.p12" keyS
```

Para configurar el server Apache que nos proporciona XAMPP primero tendremos que poner los archivos "mars-server.crt, mars-server.key, my-ca.crt.^{en} la siguiente ruta:

'/opt/lampp/etc/ssl.crt/' los ficheros con la extensión .crt

'/opt/lampp/etc/ssl.key/' el fichero con la extensión .key

Una vez que tenemos los certificados y la clave en las rutas adecuadas editamos el fichero httpd-ssl.conf. Dependiendo de la versión puede tener un aspecto u otro, en nuestro caso hemos definido un nuevo VirtualHost con la siguiente información. En el caso de trabajar bajo un sistema Linux se puede copiar y pegar las siguientes líneas en el archivo:

```
<VirtualHost _default_:443>

DocumentRoot "/opt/lampp/htdocs"
ServerName localhost:443
ServerAdmin you@example.com
ErrorLog "/opt/lampp/logs/error_log"
TransferLog "/opt/lampp/logs/access_log"

SSLEngine on

SSLCertificateFile "/opt/lampp/etc/ssl.crt/mars-server.crt"
SSLCertificateKeyFile "/opt/lampp/etc/ssl.key/mars-server.key"
SSLCertificateChainFile "/opt/lampp/etc/ssl.crt/my-ca.crt"

SSLCACertificateFile "/opt/lampp/etc/ssl.crt/my-ca.crt"

<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory "/opt/lampp/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

BrowserMatch "MSIE [2-5]" \
```

```

        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0

CustomLog "/opt/lampp/logs/ssl_request_log" \
        "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

<Directory "/opt/lampp/htdocs">
    Options Indexes
    AllowOverride None
    Allow from from all
    Order allow,deny
</Directory>

</VirtualHost>

```

Si hemos seguido correctamente las instrucciones ya nos debería permitir acceder a la dirección <https://localhost/duocode> aunque nos saldrá un mensaje de que el certificado no está verificado (es un certificado que hemos creado nosotros) así que lo añadimos como excepción y ya tendríamos DuoCode instalado.

■ Node.js

Para poder trabajar con PhoneGap - Cordova será necesario tener instalado Node.js en nuestro equipo. Podemos hacerlo descargándolo desde la web '<https://nodejs.org/>' o directamente desde un terminal:

```

$ sudo add-apt-repository ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install nodejs

```

Es necesario que la versión sea 0.8+, podemos comprobarlo tecleando desde el terminal:

```
$ node -v
```

■ PhoneGap

PhoneGap nos servirá para crear una app móvil desde el HTML5, CSS y JS de nuestro proyecto.

Para instalarlo podemos descargarlo desde la web '<http://phonegap.com>' o directamente desde un terminal:

```
$ npm install -g phonegap
```

■ Eclipse y Android SDK

Gracias al plugin de Android podremos usar Eclipse como IDE para probar la app móvil de DuoCode.

Instalaremos la última versión de Eclipse descargándolo desde la web '<http://eclipse.org>'

Una vez descargado lo abrimos y vamos a añadir el plugin para Android. Pinchamos en **help - Install New Software** y añadimos la siguiente URL '<https://dl-ssl.google.com/android/eclipse/>' y le damos a **next** hasta el final.

Reiniciamos Eclipse y tendremos que especificar la dirección del **Android SDK** que acabamos de descargar para que se actualice y tener Eclipse listo.

Podemos importar la carpeta del proyecto que encontramos en DuoCode y probarlo con el emulador.

Referencias

- [1] Acuerdo del consejo de universidades, (b.o.e. 4 de agosto de 2009). <http://www.boe.es/boe/dias/2009/08/04/pdfs/BOE-A-2009-12977.pdf>.
- [2] Bussu. <https://www.busuu.com/es/>.
- [3] Duolingo. <https://es.duolingo.com/>.
- [4] Moodle. <https://moodle.org/>.
- [5] Plan de estudios del grado de ingeniería informática de la universidad complutense. <https://informatica.ucm.es/estudios/2014-15/grado-ingenieriainformatica-planestudios>.
- [6] Stack overflow. <http://stackoverflow.com/>.
- [7] Wikipedia. <https://es.wikipedia.org>.
- [8] A. R. (coordinador). Estudio de viabilidad de un entorno de aprendizaje colaborativo de lenguajes de programación. Informe del PIMCD 97/2014, 2015.

A. Especificación de requisitos

En esta sección se exponen todos los requisitos existentes de nuestra aplicación, explicando la función de cada uno de ellos, sus parámetros de entrada y su respuesta.

A.1. localhost/duocode/rest/temas

En este apartado se explica cómo hacer un GET para obtener todos los temas y un POST para crear un nuevo tema.

REQ01 – GET: Devuelve la lista de todos los temas existentes en modo URL. No hace falta enviarle ninguna información.

- Response:

```
{'temas': ['localhost/duocode/rest/temas/1',  
'localhost/duocode/rest/temas/2']}
```

REQ02 – POST: Crea un tema nuevo. Por una parte le pasamos por Payload la información del tema que queremos crear (título, descripción y orden en el que queremos que se muestre):

- Payload:

```
{'titulo': 'Bucles', 'descripcion': 'En este tema veremos bucles while',  
'orden': '2'}
```

Por otra parte le enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” e “id”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente, en este caso el id será “-1”, ya que no ha sido asignado ninguno; si es negativa todo ha ido correctamente y el id será el asignado a este nuevo recurso. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no', 'id': '4'}  
{'error': 'si', 'id': '-1'}
```

A.2. localhost/duocode/rest/temas/idTema

En este apartado se explica cómo hacer un GET para obtener un tema en específico, un PUT para modificarlo y un DELETE para eliminarlo. En estos requisitos obtenemos el ID del tema mediante la URL.

REQ03 – GET: Devuelve los datos y las lecciones que tiene el tema.

- Response:

```
{'titulo': 'Bucles', 'descripcion': 'En este tema veremos bucles while',  
'fechaCreacion': '17/11/2014', 'lecciones':  
['localhost/duocode/rest/lecciones/1', 'localhost/duocode/rest/lecciones/2'],  
'orden': '2'}
```

REQ04 – PUT: Modifica el tema en su totalidad.

- Payload:

```
{'titulo': 'tituloTema', 'descripcion': 'descripcionTema',  
'orden': 'int con el orden en el que lo quieres mostrar'}
```

Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

REQ05 – DELETE: Borra el tema completamente. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”, en caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```


A.3. localhost/duocode/rest/lecciones/

En este apartado se explica cómo hacer un GET para obtener todas las lecciones y un POST para crear una nueva lección.

REQ06 – GET: Devuelve la lista de todas las lecciones existentes en modo URL. No hace falta enviarle ninguna información.

- Response:

```
{‘lecciones’: [‘localhost/duocode/rest/lecciones/1’,  
‘localhost/duocode/rest/lecciones/2’]}
```

REQ07 – POST: Crea una nueva lección. Por una parte le pasamos por Payload la información de la lección que queremos crear (título, descripción, explicación que aparecerá al inicio de ésta, orden en el que queremos que se muestre, ID del tema al que pertenecerá, un array de ejercicios que compondrán la lección y otro array de lecciones de las que depende para ser desbloqueada):

- Payload:

```
{‘titulo’: ‘titulo lección’, ‘descripcion’: ‘descripción lección’,  
‘explicacion’: ‘explicación detallada’, ‘orden’: ‘4’, ‘idTema’: ‘1’,  
‘idEjercicios’: [‘8’, ‘14’], ‘leccionesDesbloqueadoras’: [‘1’, ‘2’]}
```

Por otra parte le enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{‘idUserario’: ‘2’, ‘token’: ‘token’, ‘network’: ‘network’}
```

La respuesta que obtenemos está conformada por “error” e “id”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente, en este caso el id será “-1” ya que no ha sido asignado ninguno; si es negativa todo ha ido correctamente y el id será el asignado a esta nueva lección. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{‘error’: ‘no’, ‘id’: ‘3’}  
{‘error’: ‘si’, ‘id’: ‘-1’}
```

A.4. localhost/duocode/rest/lecciones/idLeccion

En este apartado se explica cómo hacer un GET para obtener una lección en específica, un PUT para modificarla y un DELETE para eliminarla. En estos requisitos obtenemos el ID del candidato mediante la URL.

REQ08 – GET: Devuelve los datos de la lección.

- Response:

```
{'titulo': 'título de la lección (ej. Bucles fácil)',
'descripcion': 'descripción de la lección (ej. bucles para practicar)',
'explicacion': 'explicación detallada', 'fechaCreacion': '17/11/2014',
'ejercicios': ['localhost/duocode/rest/ejercicios/2',
'localhost/duocode/ejercicios/3'],
'leccionesDesbloqueadoras': ['1', '2'], 'orden': '3'}
```

REQ09 – PUT: Modifica una lección. Aparte de cambiar los datos de una lección, en este método también podemos marcar como completada la lección para un usuario en un determinado lenguaje, por lo que tenemos dos posibles Payloads:

- Payload para modificar lección:

```
{'leccion' : {'titulo': 'titulo lección', 'descripcion':
'descripción lección', 'explicacion': 'explicación detallada',
'idEjercicios': ['1', '2'], 'leccionesDesbloqueadoras': ['2', '3'],
'orden': '2', 'idTema': '1'}}
```

- Payload para marcar como lección completada para un usuario:

```
{'idUsuarioCompletaLeccion': '3', 'lenguajeCompletadoLeccion': 'Java',
'leccion': {'titulo': 'titulo lección',
'descripcion': 'descripción de la lección',
'explicacion': 'explicación detallada', 'idEjercicios': ['1', '2'],
'leccionesDesbloqueadoras': ['2', '3'], 'orden': '2', 'idTema': '1'}}
```

Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUsuario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

REQ10 – DELETE: Borra la lección completamente. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUsuario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

A.5. localhost/duocode/rest/ejercicios

En este apartado se explica cómo hacer un GET para obtener todos los ejercicios y un POST para crear un nuevo ejercicio.

REQ11 – GET: Devuelve la lista de todos los ejercicios existentes en modo URL. No hace falta enviarle ninguna información.

- Response:

```
{‘ejercicios’: [‘localhost/duocode/rest/temas/ejercicios/1’,  
‘localhost/duocode/rest/ejercicios/2’]}
```

REQ12 – POST: Crea un ejercicio nuevo. Por una parte le pasamos el nombre y los enunciados (un mismo ejercicio puede tener varios enunciados porque cada uno corresponde a distintos lenguajes de programación):

- Payload:

```
{‘nombre’: ‘nombreDelEjercicio’, ‘enunciados’: [‘1’, ‘2’]}
```

Por otra parte le enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{‘idUserario’: ‘2’, ‘token’: ‘token’, ‘network’: ‘network’}
```

La respuesta que obtenemos está conformada por “error” e “id”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente, en este caso el id será “-1” ya que no ha sido asignado ninguno; si es negativa todo ha ido correctamente y el id será el asignado a este nuevo recurso. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{‘error’: ‘no’, ‘id’: ‘4’}  
{‘error’: ‘si’, ‘id’: ‘-1’}
```

A.6. localhost/duocode/rest/ejercicios/idEjercicio

En este apartado se explica cómo hacer un GET para obtener un ejercicio en específico, un PUT para modificarlo y un DELETE para eliminarlo. En estos requisitos obtenemos el ID del ejercicio mediante la URL.

REQ13 – GET: Devuelve los datos del ejercicio, los enunciados que tiene y el nombre del lenguaje asociado a cada uno.

- Response:

```
{‘nombre’: ‘nombreDelEjercicio’, ‘fechaCreacion’: ‘17/11/2014’,  
‘enunciados’: [{‘enunciado’: ‘localhost/duocode/rest/enunciados/5’,  
‘nombreLenguaje’: ‘Java’}, {‘enunciado’:  
‘localhost/duocode/rest/enunciados/8’, ‘nombreLenguaje’: ‘C++’}]}
```

REQ14 – DELETE: Borra un ejercicio completamente. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserio': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

A.7. localhost/duocode/rest/enunciados

En este apartado se explica cómo hacer un GET para obtener todos los enunciados y un POST para crear un nuevo enunciado.

REQ15 – GET: Devuelve la lista de todos los enunciados existentes en modo URL. No hace falta enviarle ninguna información.

- Response:

```
{'enunciados': [{'enunciado': 'localhost/duocode/rest/enunciados/1',  
'nombreLenguaje': 'Java'}, {'enunciado': 'localhost/duocode/rest/enunciados/2',  
'nombreLenguaje': 'C++'}]}
```

REQ16 – POST: Crea un enunciado nuevo. Por una parte le pasamos el lenguaje, el código y el ID del ejercicio correspondiente:

- Payload:

```
{'nombreLenguaje': 'Java', 'codigo': 'código del enunciado',  
'idDelEjercicioQueResuelve': '1'}
```

Por otra parte le enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserio': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” e “id”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente, en este caso el id será “-1” ya que no ha sido asignado ninguno; si es negativa todo ha ido correctamente y el id será el asignado a este nuevo enunciado. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no', 'id': '4'}  
{'error': 'si', 'id': '-1'}
```

A.8. localhost/duocode/rest/enunciados/idEnunciado

En este apartado se explica cómo hacer un GET para obtener un enunciado en específico, un PUT para modificar un enunciado y un DELETE para eliminarlo. En estos requisitos obtenemos el ID del candidato mediante la URL.

REQ17 – GET: Devuelve los datos del enunciado.

- Response:

```
{'fechaCreacion': '17/11/2014', 'codigo': 'código del enunciado a resolver',  
'nombreLenguaje': 'Java', 'idDelEjercicioQueResuelve': '1'}
```

REQ18 – PUT: Modifica un enunciado.

- Payload:

```
{'nombreLenguaje': 'Java', 'codigo': 'código del enunciado',  
'idDelEjercicioQueResuelve': '1'}
```

Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

REQ19 – DELETE: Borra un enunciado completamente. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

A.9. localhost/duocode/rest/lenguajes/

En este apartado se explica cómo hacer un GET para obtener todos los lenguajes y un POST para crear un nuevo lenguaje.

REQ20 – GET: Devuelve la lista de todos los lenguajes existentes. No hace falta enviarle ninguna información.

- Response:

```
{'lenguajes': [{'nombre': 'Java'}, {'nombre': 'C++}]}
```

REQ21 – POST: Crea un lenguaje nuevo. La única información necesaria es el nombre.

- Payload:

```
{'nombre': 'Python'}
```

Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” y “nombreConfirmacion”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente; si es negativa todo ha ido correctamente y el “nombreConfirmacion” será el asignado a este nuevo lenguaje. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no', 'nombreConfirmacion': 'Python'}
```

A.10. localhost/duocode/rest/candidatos/

En este apartado se explica cómo hacer un GET para obtener todos los candidatos y un POST para crear un nuevo candidato.

REQ22 – GET: Devuelve la lista de todos los candidatos existentes en modo URL. No hace falta enviarle ninguna información.

- Response:

```
{'candidatos': ['localhost/duocode/rest/candidatos/1',  
'localhost/duocode/rest/candidatos/2']}
```

REQ23 – POST: Crea un nuevo candidato asociado al usuario que solicita la petición. Por un lado le enviamos el código del candidato, el ID del ejercicio que resuelve, el lenguaje en el que está escrito el candidato y el lenguaje del enunciado:

- Payload:

```
{'codigo': 'codigoDelCandidato', 'idEjercicio': '4',
  'nombreLenguajeDestino': 'Java', 'nombreLenguajeOrigen': 'C++'}
```

Por otra parte le enviamos por Header el ID del usuario, el token y el network, que nos sirven para saber qué usuario es el que ha enviado el candidato:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” e “idCandidato”. Si la respuesta en el campo “error” es afirmativa significa que algo ha fallado y no ha podido terminar la operación correctamente, en este caso el id será “-1” ya que no ha sido asignado ninguno; si es negativa todo ha ido correctamente y el id será el asignado a este nuevo candidato.

- Response:

```
{'error': 'no', 'id': '4'}
```

A.11. localhost/duocode/rest/candidatos/idCandidato

En este apartado se explica cómo hacer un GET para obtener un candidato en específico, un PUT para modificarlo y un DELETE para eliminarlo. En estos requisitos obtenemos el ID del candidato mediante la URL.

REQ24 – GET: Devuelve los datos de un candidato, incluidos los votos que tiene:

- Response:

```
{'idEjercicio': 'localhost/duocode/rest/ejercicios/4',
  'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++',
  'codigo': 'código del candidato', 'idUserarioCreador': '2',
  'fechaCreacion': '17/11/2014', 'votos': [{'idUserarioVoto': '8',
  'voto': 'pos'}, {'idUserarioVoto': '5', 'voto': 'neg'}]}
```

REQ25 – PUT: Excepcionalmente no modifica todo el candidato, sino que sirve para que un usuario pueda votar, modificar el voto o eliminarlo (si se vuelve a votar positivo o negativo el voto se anula). Se envía el ID del usuario y el voto (1 si es positivo y 0 si es negativo).

- Payload:

```
{'votar': {'idUserario': '6', 'voto': '1'}}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente.

- Response:

```
{'error': 'no'}
```

REQ26 – DELETE: Borra un candidato completamente.

- Header:

```
{'idUserio': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

A.12. localhost/duocode/rest/usuarios/

En este apartado se explica cómo hacer un GET para obtener todos los usuarios.

REQ27 – GET: Devuelve todos los usuarios. A diferencia de otros, este GET solo lo puede hacer un administrador por lo que necesitamos nuevamente del Header.

- Header:

```
{'idUserio': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” y la lista de usuarios. Si “error” tiene una respuesta afirmativa, significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente y muestra la lista de usuarios mediante su URL. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no', 'usuarios': ['localhost/duocode/rest/usuario/1',  
'localhost/duocode/rest/usuario/2']}
```

A.13. localhost/duocode/rest/usuarios/idUsuario

En este apartado se explica cómo hacer un GET para obtener un usuario en específico y un DELETE para eliminarlo. En estos requisitos obtenemos el ID del candidato mediante la URL.

REQ28 – GET: Devuelve la información asociada a un usuario. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario que accede es el mismo del que se da la información:

- Header:

```
{'idUserio': '2', 'token': 'token', 'network': 'network'}
```

La respuesta es la información detallada de toda la sesión del usuario

- Response:


```
{'nick': 'nickDelUsuariro', 'leccionesCompletadas':
['localhost/duocode/rest/lecciones/1', 'localhost/duocode/rest/lecciones/2'],
'favoritos': [{'ejercicio': 'localhost/duocode/rest/ejercicios/5',
'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++'},
{'ejercicio': 'localhost/duocode/rest/ejercicios/7',
'nombreLenguajeOrigen': 'Python', 'nombreLenguajeDestino': 'C++'}],
'historialEjercicios': [{'idEnvio': '1',
'ejercicio': 'localhost/duocode/rest/ejercicios/4',
'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++',
'codigo': 'código enviado', 'fecha': '17/11/2014', 'puntuacion': '2'},
{'idEnvio': '2', 'ejercicio': 'localhost/duocode/rest/ejercicios/9',
'nombreLenguajeOrigen': 'Python', 'nombreLenguajeDestino': 'Perl',
'codigo': 'código enviado', 'fecha': '18/11/2014', 'puntuacion': '7'}],
'candidatosPropuestos': ['localhost/duocode/rest/candidatos/18',
'localhost/duocode/rest/candidatos/23']}
```

REQ29 – DELETE: Borra un usuario completamente. Enviamos por Header el ID del usuario, el token y el network, que nos sirven para comprobar que el usuario es administrador y tiene permisos para hacer esta operación:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error”. En caso afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente. El único caso de error posible es que el usuario no tenga los permisos necesarios.

- Response:

```
{'error': 'no'}
```

A.14. localhost/duocode/rest/envios

En este apartado se explica cómo hacer un GET para obtener todos los envíos y un POST para crear un nuevo envío.

REQ30 – GET: Devuelve todos los envíos para que un administrador pueda tener información de ellos. Como solo puede tener acceso a esto el administrador, es necesaria la siguiente información:

- Header:

```
{'idUserario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta se compone de usuarios con historiales de ejercicios.

- Response:

```
{'envios': [{'idUserario': '1', 'historialEjercicios': [{'idEnvio': '1',
'ejercicio': 'localhost/duocode/rest/ejercicios/5',
'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++',
'codigo': 'código enviado', 'fecha': '17/11/2014', 'puntuacion': '2'},
{'idEnvio': '2', 'ejercicio': 'localhost/duocode/rest/ejercicios/7',
```

```
{
  'nombreLenguajeOrigen': 'Python', 'nombreLenguajeDestino': 'Perl',
  'codigo': 'código enviado', 'fecha': '18/11/2014', 'puntuacion': '7'}],
{'idUsuario': '4', 'historialEjercicios': [{
  'idEnvio': '1',
  'ejercicio': 'localhost/duocode/rest/ejercicios/6',
  'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++',
  'codigo': 'código enviado', 'fecha': '17/11/2014', 'puntuacion': '2'},
{'idEnvio': '2', 'ejercicio': 'localhost/duocode/rest/ejercicios/5',
  'nombreLenguajeOrigen': 'Python', 'nombreLenguajeDestino': 'Perl',
  'codigo': 'código enviado', 'fecha': '18/11/2014', 'puntuacion': '7'}]]}]}
```

REQ31 – PUT: Corrige un ejercicio y también se comprueba si se ha completado la lección; en caso afirmativo se marca como completada en la base de datos. Por una parte enviamos la URL del ejercicio, el lenguaje del enunciado, el lenguaje de la solución y el código:

- Payload:

```
{
  'ejercicio': 'localhost/duocode/rest/ejercicios/idEjercicio1',
  'nombreLenguajeOrigen': 'Java', 'nombreLenguajeDestino': 'C++',
  'codigo': 'código enviado en el lenguaje de destino'}
```

Por otra parte enviamos el idUsuario, el token y el network para comprobar que el usuario que envía el ejercicio para corregir es el que tiene la sesión iniciada.

- Header:

```
{
  'idUsuario': '2', 'token': 'token', 'network': 'network'}
```

La respuesta que obtenemos está conformada por “error” y “puntuacion”. Si “error” tiene un valor afirmativo significa que algo ha fallado y no ha podido terminar la operación correctamente; en caso negativo todo ha ido correctamente y devuelve la puntuación que ha obtenido el ejercicio al ser corregido. El único caso de error posible es que el usuario no coincida.

- Response:

```
{
  'error': 'no', 'puntuacion': '2'}
```