

Projecte Final de Carrera

Memòria

Aprenentatge d’idiomes

Alumnes: Capell Brufau, Eduard i Lorca Sans, Salvador

Consultor: Roset Mayals, Roman

Contingut

[Capítol 1: Introducció 3](#_Toc338015765)

[1.1. Idea 3](#_Toc338015766)

[1.2. Possible funcionament 3](#_Toc338015767)

[1.3. Objectius 4](#_Toc338015768)

[1.3.1. Objectiu general 4](#_Toc338015769)

[1.3.2. Objectius específics 4](#_Toc338015770)

[1.4. Planificació amb fites i temporalització 4](#_Toc338015771)

[Capítol 2: Estat de l’art 6](#_Toc338015772)

[Capítol 3: Estudi de mercat 9](#_Toc338015773)

[3.1. Diccionaris 9](#_Toc338015774)

[3.2. Flashcards 9](#_Toc338015775)

[3.3. Àudio 9](#_Toc338015776)

[3.4. Pràctica de Traços 10](#_Toc338015777)

[Capítol 4: Anàlisi i disseny 11](#_Toc338015778)

[4.1. Tecnologia 11](#_Toc338015779)

[4.2. Arquitectura del sistema 12](#_Toc338015780)

[4.3. Disseny preliminar de les pantalles 13](#_Toc338015781)

# Capítol 1: Introducció

## 1.1. Idea

Partim d’una idea de projecte que es basa en la creació d’un sistema d’aprenentatge de l’idioma japonès mitjançant dues utilitats:

* Llistes d’estudi
* Targetes didàctiques

La primera utilitat es basa en una plataforma per a definir llistes d’estudi. Una llista d’estudi és un conjunt de fitxes o elements que contenen una paraula en japonès (amb símbols *kanji*) i que estan organitzades de forma que un estudiant pot anar aprenent els símbols i la seva pronunciació navegant per les llistes d’estudi. Per exemple, podem definir una llista anomenada *colors*, i que les fitxes t’ensenyin com s’escriu i es pronuncia el *color* en qüestió.

La segona utilitat té com a objectiu la consolidació de l’aprenentatge anterior. Les targetes didàctiques (en anglès *flashcards*) són un conjunt de targetes que contenen informació, com a paraules i nombres, en un o tots dos costats usades per adquirir diversos coneixements a través de la relectura del conjunt de targetes. En un costat de la targeta apareixeria el símbol d’un dels elements en la llengua que volem aprendre (en aquest cas la japonesa) i en l'altre la pronunciació en japonès i el seu significat en català. Les *flashcards* s'usen àmpliament com un exercici d'aprenentatge per ajudar a la memorització per mitjà de la repetició espaiada.

## 1.2. Possible funcionament

L’aplicació en mòbils permet la gestió de les llistes d’estudi i les paraules, més la visualització de les *flashcards*. Les característiques principals són les següents:

* Personalització de les llistes: Possibilitat de crear noves llistes d’estudi i d’afegir/modificar/esborrar elements a aquestes llistes. Per exemple: Dies de la setmana, mesos de l’any, colors, números, coses d'un hotel, aliments, etc.
* Gestió d’un diccionari de paraules: Creació, modificació i esborrat de paraules que proveiran les llistes d’estudi.
* Visualització de les *flashcards*: Desplaçament entre fitxes amb opció per escoltar com es pronuncia la paraula que conté i presentació de les fitxes usant la metodologia del sistema Leitner1.

1 El sistema Leitner és un sistema de preguntes i respostes basat en fitxes inventat per Sebastian Leitner.

## 1.3. Objectius

### 1.3.1. Objectiu general

El nostre objectiu principal és el disseny i implementació d’una aplicació web adaptada als dispositius mòbils (*tablets*, *smartphones*) que disposin de navegador.

### 1.3.2. Objectius específics

Ens agradaria complir els següents objectius específics:

* Lliurar una eina d’aprenentatge que s’adapti a les necessitats dels estudiants que s’inicien en l’aprenentatge de la llengua japonesa.
* Assimilar tots els conceptes i nocions nous que aniran sortint durant el cicle de vida del programari, i més en concret la metodologia que ens portarà a aconseguir l’èxit.

I de manera més precisa:

* Programar la part servidor (mitjançant una API) i la part client del dispositiu mòbil.
* Col·laborar amb un altre company de titulació i realitzar un projecte comú més ambiciós.

## 1.4. Planificació amb fites i temporalització

A continuació es mostra la planificació i temporització de les tasques que hem definit es en aquest diagrama *Gantt*:



Il·lustració 1 – Diagrama Gantt de planificació

# Capítol 2: Estat de l’art

Actualment existeix una aplicació que té com a finalitat l’aprenentatge i l’avaluació personal de l’escriptura japonesa. El Quadern Virtual d’Escriptura (QVE) és una aplicació que es pot fer servir amb ordinador + ratolí, tablet android o tauleta gràfica, i conté, per una banda, un assistent d’escriptura virtual que permet l'aprenentatge i pràctica del traç de caràcters, i per altra banda, una part teòrica sobre els caràcters. Aquesta part teòrica és la que menys profunditat presenta i de la que volem treure profit.



Il·lustració 2 – Selecció d’imatges kanji en el QVE

L’eina treballa amb 10 símbols *kanji* que representen 10 paraules en el nostre alfabet, i quan sel·lecciones un símbol et porta a la seva pantalla d’informació.



Il·lustració 3 – Pantalla d’informació d’un kanji en el QVE

En aquesta pantalla es mostren explicacions sobre el *kanji*, significats, àudios *on-yomi* i *kun-yomi,* i alguns exemples d’ús. També es pot observar una animació que indica l’ordre de traços i reproduïble mitjançant uns botons.

L’àmbit de l’aplicació és practicar l’escriptura dels símbols japonesos, i es fa amb les pantalles “Escriptura”, “Estil lliure” i “Exercicis”.







Il·lustració 4 – Pantalles de pràctica d’escriptura

Aquesta eina de simulació de l’escriptura del *kanji* permet a l’estudiant fer un traçat en l’ordre correcte amb el dit, llàpis tàctil o ratolí (si està usant un ordinador). El sistema indica a l’usuari si l’ordre que segueix és correcte o no mitjançant un flash verd o vermell respectivament.

# Capítol 3: Estudi de mercat

Tot seguit analitzarem les aplicacions existents actualment per a dispositius mòbils que tenen com a objectiu l’aprenentatge del japonès.

Dividim les aplicacions existents en les següents categories: diccionaris, flashcards, àudio, pràctica dels traços. Algunes de les aplicacions encaixen en més d’una de les categories proposades (per exemple, hi ha aplicacions que fan la funció de diccionari i tenen un mòdul d’àudio per saber la pronunciació d’una paraula).

## 3.1. Diccionaris

És la categoria més freqüent, probablement per la facilitat d’implementació. La característica bàsica és la possibilitat de cerca de paraules, amb el seu significat.

Característiques més interessants de les aplicacions en aquesta categoria:

* Cerca en japonès o en anglès.
* Cerca per radicals.
* Cerca per nombre de traços.
* Possibilitat d’introducció de caràcters escrits “a mà”, la qual cosa implica que hi ha implementat un sistema de reconeixement dels traços dels caràcters.
* Creació de llistes de paraules preferides.

<Aquí hi posaré alguna captura de pantalla>

## 3.2. Flashcards

Aquesta és una categoria molt freqüent en les aplicacions per l’estudi d’idiomes. Consisteix en una simulació de targetes didàctiques. A una banda de la targeta hi ha una pregunta, i a l’altra banda hi ha la resposta. Les plataformes mòbils són un entorn ideal per l’ús d’aquesta metodologia, per la qual cosa han triomfat molt.

Aquestes aplicacions, a més de replicar la funcionalitat de *flashcards*, també permeten d’aportar valor addicional, de la següent forma:

* Estadístiques
* Repetició de les preguntes on l’estudiant ha fallat més
* Possibilitat d’afegir targetes personalitzades
* Interacció amb d’altres usuaris o aplicacions (per exemple, correu electrònic o missatgeria instantània)

<Aquí hi posaré alguna captura de pantalla>

## 3.3. Àudio

Aquestes aplicacions tenen una funcionalitat molt simple: donat un caràcter o una paraula, permeten la reproducció de la pronunciació de la mateixa.

Algunes aplicacions van més enllà i també permeten que l’usuari gravi la seva versió de la paraula, per poder-la comparar amb la pronunciació correcta.

<Aquí hi posaré captura de pantalla>

## 3.4. Pràctica de Traços

Una característica distintiva de l’idioma japonès és els traços que formen els caràcters. Donat que aquest tret constitueix un àmbit d’estudi en si mateix, és lògic que hi hagi moltes aplicacions que permeten la pràctica dels traços.

La mecànica és simple: normalment les aplicacions mostren l’ordre dels traços en una animació, i llavors és l’estudiant el que ha de reproduir els traços. Un cop acabada la introducció, algunes aplicacions reconeixen aquests traços i avaluen el rendiment de l’usuari, mentre que d’altres aplicacions es limiten a deixar en mans de l’estudiant la comprovació de la correcció dels traços introduïts.

<Aquí hi posaré captura de pantalla>

# Capítol 4: Anàlisi i disseny

## 4.1. Tecnologia

Aquest projecte l’hem realitzat amb el *framework* **Sencha Touch 2**. És un gran marc de treball que permet al desenvolupador construir aplicacions que funcionin en iOS, Android, BlackBerry, Kindle Fire, etc.



Il·lustració 6 – Framework escollit , el Sencha Touch 2

Per a la part client hem escollit l’entorn de treball **Sencha Architect 2**, el qual permet generar fitxers HTML5 i CSS3 que seran utilitzats sobre un servidor web.



Il·lustració 6 – Llenguatges usats per a l’elaboració del projecte

La part del servidor l’hem fet amb **Node.js**. Aquesta és una plataforma creada amb el motor JavaScript de Chrome, per desenvolupar aplicacions ràpides. Node.js es basa en un model que respon a esdeveniments, sense colls d'ampolla d'Entrada/Sortida que ho fan lleuger i eficient, perfecte per a aplicacions en temps real amb molta càrrega de dades i que corren a través de múltiples dispositius.



Il·lustració 7 – Servidor escollit, pur JavaScript

Per últim, a l’hora de centralitzar el treball fet hem decidit tirar d’un repositori centralitzat públic per guardar el nostre codi *open source*. És un dels sistemes de controls de versions més usats i ofereix hospedatge i altres serveis socials. El seu nom és **GitHub**, i el nostre projecte es pot trobar al següent enllaç:

<https://github.com/salvinha/UOCPFC_Eduard_Salva.git>



Il·lustració 8 – Sistema de control de versions del nostre projecte

## 4.2. Arquitectura del sistema

L’arquitectura de l’aplicació web seguirà el model vista controlador (MVC) . Aquest patró és aplicat per l’arquitectura del *framework* Sencha Touch.

Bàsicament, aquest model es comporta de la següent forma:



Il·lustració 9 – Model Vista Controlador (MVC) del framework Sencha Touch

* Una capa **Model** que s’encarrega de la representació especifica de la informació amb la que el sistema opera. Es composa per la lògica de negoci i pel sistema de gestió de base de dades.
* Una capa **Vista** que s’encarrega de la representació del model en un format adequat per interactuar, i que es coneix com interfície d’usuari.
* Una capa **Controlador** que s’encarrega d’accedir al model per tal de consultar les dades que calguin representar a la vista. És a dir, el controlador és la capa que s’encarrega de la comunicació entre la vista i el model.

El patró MVC és molt utilitzat en entorns web on l’usuari interactua amb la pàgina, plana o vista; el controlador rep la notificació de l’acció sol·licitada per l’usuari, accedeix al model si convé i executa l’acció. Després la vista s’encarrega de rebre les dades del model per visualitzar el resultat a l’usuari.

## 4.3. Disseny preliminar de les pantalles

Hem realitzat una primera versió de les pantalles amb l’ajuda de l’eina *Prototyper Free* la qual ens ha ajudat a fer una maqueta digital sense funcionalitat.



Il·lustració 10 – Pantalla d’inici de la maqueta

## disseny preliminar del mòdul del servidor

Seguint amb el model MVC exposat anteriorment, ara ens centrarem en el disseny de la part corresponent al servidor.

La idea és fer disponibles un conjunt de serveis, que seran els que invocaran les accions del client. D’aquesta manera, sempre que es vulgui realitzar alguna operació des de les pantalles, aquesta operació serà gestionada pel servidor. Aquesta arquitectura té avantatges i desavantatges, que mostrem a continuació.

### Avantatges de la implementació de les accions EN servidor

* Centralitat: totes les respostes a qualsevol acció estan sota el nostre control. Si hi ha un error, o bé si hi ha alguna millora a fer, n’hi ha prou d’actualitzar el servidor per tal de fer disponible la millora a tots els clients, sense que faci falta que els clients s’actualitzin.
* Lleugeresa: menys pes de l’aplicació client. Si volguéssim fer disponible tota la funcionalitat en els clients, això implicaria que aquests s’haurien de baixar totes les dades necessàries per fer funcionar el sistema. Això té rellevància tant en el consum de dades de l’usuari, com en l’ocupació d’espai en la memòria del dispositiu de l’usuari.
* Potència: les accions a realitzar les fa el servidor, i no pas el client. Això implica que la potència de càlcul resideix en el servidor, i el client es despreocupa de tota aquesta lògica. És un factor molt important, perquè si posem més funcionalitat al dispositiu mòbil això implica que pot suposar un consum de bateria més elevat i l’experiència d’usuari se’n pot ressentir.
* Manteniment: el disseny client – servidor ens obliga a mantenir un servei en el llarg termini, perquè els usuaris confiaran que l’aplicació els ha de funcionar durant un llarg període de temps. Seran necessaris mecanismes de monitorització i de backup per tal d’assegurar la màxima disponibilitat possible.

### Inconvenients de la implementació de les accions EN servidor

* Connectivitat: el fet que les accions s’hagin de realitzar sempre contra el servidor obliga a estar connectats permanentment. Hi ha opcions intermèdies, com són la de precarregar (memòria cau) certa informació que preveiem que serà utilitzada, però no es pot comptar amb fer servir l’aplicació de manera completament offline.
* Dependència: si les accions són al servidor, el programador segueix controlant l’aplicació encara que l’usuari se l’hagi baixat. Això implica que si el servidor cau o hi ha problemes en el mòdul del servidor, el client és inoperatiu, malgrat que tingui connectivitat i tingui la versió més recent de l’aplicació.

### El controlador

El Controlador serà un únic punt d’accés a qualsevol funcionalitat del servidor. Totes les peticions arribaran al Controlador, que les gestionarà de la manera convenient. Les responsabilitats del Controlador són les següents:

* Rebre totes les peticions.
* Donada una petició, esbrinar a qui correspon la tasca de dur-la a terme, i delegar-la-hi.
* Recollir els resultats de la petició.
* Retornar els resultats de la petició al client.

<Aquí hi he d’afegir un diagrama de funcionament del Controlador>

### El Model

El Model, en el paradigma MVC, és el conjunt de components que representen la realitat que estem modelant. En el cas concret de la nostra aplicació això suposarà:

* Codi *nodejs* amb les classes que representen els objectes que prenen part en el sistema.
* Base de dades relacional, amb la informació desada que l’aplicació necessita per a funcionar.

### Els objectes del model

Per representar la realitat que ens ocupa, necessitem un conjunt d’objectes, que veiem a continuació:

<Aquí hi he d’afegir un diagrama de classes UML on hi apareguin les classes que necessitem>

Descripció breu de cada classe:

**Paraula**: una paraula.

**Idioma**: idioma en què està una paraula.

**Pronunciació**: representació àudio de la pronunciació de la paraula.

**Llista**: llista de paraules.

**Usuari**: un usuari del sistema.

**Flashcard**: una flashcards de les que utilitza l’usuari per practicar.

### La base de dades

Per tal de desar tota la informació que necessitarà l’aplicació, utilitzarem una base de dades relacional. Es proposa el servidor MySQL, versió 5.5.

Informació que es desarà a la base de dades:

* **Usuaris**:
  + Identificador de cada usuari. Pot ser un identificador explícit (Email / contrasenya) o un simple codi associat a l’ID del terminal des d’on es connecta l’usuari. Una altra opció és utilitzar un identificador explícit triat per l’usuari (Email) però sense obligar-lo a autenticar-se amb una contrasenya.
  + Peticions al sistema: cada cop que un usuari fa una petició al sistema, s’enregistrarà aquesta petició.
* **Paraules**:
  + Desarem la paraula en l’idioma japonès i també la seva traducció en català.
  + La paraula en japonès es desarà en l’alfabet japonès, amb els seus *kanji* corresponents. La versió que proposem de MySQL suporta aquesta funcionalitat.
  + Es permetrà desar una representació àudio de la manera de pronunciar la paraula en cada un dels idiomes.
  + Els usuaris podran afegir paraules pròpies, de manera que aquesta taula tindrà unes paraules predeterminades, i unes altres que cada usuari afegirà.
* **Llistes**:
  + Són llistes de paraules, agrupades per àmbits (menjars, colors, llocs geogràfics, etc.).
  + Inicialment hi ha un conjunt de llistes predeterminades, però els usuaris n’hi poden afegir de pròpies, i vincular-hi les paraules que vulguin.
* **Flashcards**:
  + <Pendent Email>

<Aquí hi posaré el diagrama E/R, semblant al diagrama UML>

### comunicacions client – servidor

Cada comunicació entre client i servidor es farà mitjançant peticions JSON[[1]](#footnote-1). Totes les peticions tenen la següent estructura:

* Nom de la petició
* Paràmetres (alguns obligatoris, d’altres d’opcionals)
* Resposta del servidor, que prèviament haurà realitzat les accions oportunes, si és necessari.

Les peticions de què disposarà la nostra aplicació seran les següents:

* **registre\_usuari**, per fer l’alta d’usuaris nous.
* **autenticacio\_usuari**, per accedir al sistema.
* **llistat\_diccionari**, per fer un llistat de les paraules del diccionari.
* **llistes\_personalitzades**, per veure les llistes personalitzades que té l’usuari.
* **nova\_llista\_personalitzada**, per crear una nova llista associada a l’usuari actiu.
* **edicio\_llista\_personalitzada**, per canviar el nom d’una llista personalitzada.
* **baixa\_llista\_personalitzada**, per esborrar una llista personalitzada.
* **consulta\_llista\_personalitzada**, per descarregar les paraules d’una llista personalitzada de l’usuari.
* **flashcard**, per descarregar-se la següent flashcard.

### Peticions

|  |  |  |
| --- | --- | --- |
| **Nom** | **registre\_usuari** | |
| **Descripció** | Acció que s’executa quan un usuari es dóna d’alta al sistema. Un usuari ens donarà el seu correu, una contrasenya, i el nom amb el qual vol ser conegut pel sistema. Totes aquestes informacions són obligatòries.  Finalment és possible que no es faci el registre amb tantes dades, n’hi haurà prou amb algun identificador del terminal des del qual l’usuari està interactuant amb el programa. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **Email** | | Adreça de correu de l’usuari que es registra |
| **Contrasenya** | | Contrasenya triada per l’usuari |
| **Confirmació de la contrasenya** | | Confirmació de la contrasenya triada per l’usuari |
| **Nom** | | Nom amb què l’usuari vol ser conegut pel sistema |
| Precondicions | | |
| * Tots els camps s’han emplenat correctament * L’adreça de correu és vàlida * Les dues contrasenyes són coincidents | | |
| **Postcondicions** | | |
| * L’usuari ha estat donat d’alta al sistema. * Les dades de l’usuari s’han emmagatzemat a la base de dades. La contrasenya es desarà de forma encriptada. * Es retorna un codi de confirmació al dispositiu que ha fet la petició (OK-0). * S’envia un correu electrònic de benvinguda a l’adreça especificada. (correu de verificació?). | | |
| **Excepcions** | | |
| * Si no tots els camps han estat emplenats, es retornarà un codi d’error al client (KO-1). * Si l’adreça de correu no és vàlida, es retornarà un codi d’error al client (KO-2). * Si les contrasenyes no són coincidents, es retornarà un codi d’error al client (KO-3). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **autenticacio\_usuari** | |
| **Descripció** | Acció que s’executa quan un usuari prèviament registrat entra al sistema.  Si finalment no es fa registre complert d’usuaris, aquest mòdul només consistirà a comprovar si l’ID del terminal existeix a la base de dades; si no hi és, s’hi afegirà, si ja hi és, no caldrà fer res més en aquesta acció. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **Email** | | Adreça de correu de l’usuari |
| **Contrasenya** | | Contrasenya de l’usuari |
| Precondicions | | |
| * Tots els camps s’han emplenat correctament | | |
| **Postcondicions** | | |
| * El sistema desarà a la base de dades l’acció de login, hagi tingut èxit o no. * El sistema haurà respost amb un codi que indicarà l’èxit de l’acció. (OK-0). * La contrasenya no s’enviarà en clar, sinó de forma encriptada. | | |
| **Excepcions** | | |
| * Si no tots els camps han estat emplenats, es retornarà un codi d’error (KO-1) * Si l’adreça de correu no és vàlida, es retornarà un codi d’error (KO-2). * Si la combinació de correu electrònic i contrasenya no és correcta, es retornarà un codi d’error (KO-3). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **llistat\_diccionari** | |
| **Descripció** | El client fa una consulta, indicant quina és la darrera paraula que té carregada (es proposa aquesta manera per tal de no sobrecarregar el sistema amb trànsit excessiu de dades o amb llistes excessives pel terminal).  El servidor retornarà un màxim (que especificarem posteriorment) de N paraules.  Es farà una cerca a la base de dades de les N paraules següents a la que ens està indicant el client (paraula\_base).  Per cada paraula es retorna el text en català i en japonès. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **paraula\_base** | | Paraula en català a partir de la qual es farà la cerca al diccionari de la base de dades del servidor. |
| **Precondicions** | | |
| * paraula\_base existeix al diccionari de la base de dades | | |
| **Postcondicions** | | |
| * El sistema retornarà les N paraules següents trobades al diccionari. | | |
| **Excepcions** | | |
| * Si no hi ha més paraules després de la que ens indica el client, es retornarà un codi d’error (KO-1). * Si paraula\_base no és a la base de dades, es retornarà un codi d’error (KO-2). * Si no hi ha suficients paraules (nombre de resultats < N), es retornaran només els resultats obtinguts, sense cap missatge d’error addicional. | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **llistes\_personalitzades** | |
| **Descripció** | El client fa una consulta per tal de saber quines són les llistes personalitzades de l’usuari actiu.  L’aplicació busca totes les llistes que té vinculades aquest usuari, i les retorna. La informació que es retorna per cada llista consisteix en el nom de la llista i el nombre de paraules associades a la llista.  És possible que una llista tingui 0 paraules associades. En aquest cas, la llista es retornarà igualment. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **identificador\_usuari** | | El codi que permet identificar de manera única cada dispositiu. |
| **Precondicions** | | |
| * L’usuari existeix al sistema. | | |
| **Postcondicions** | | |
| * El sistema retornarà totes les llistes personalitzades que tingui l’usuari, incloent-hi el nom de la llista i el nombre de paraules associades. | | |
| **Excepcions** | | |
| * Si el codi d’usuari especificat al paràmetre no existeix, es retornarà un codi d’error (KO-1). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **nova\_llista\_personalitzada** | |
| **Descripció** | El client fa una petició indicant qui és l’usuari actiu, i quin és el nom de la nova llista personalitzada. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **identificador\_usuari** | | El codi que permet identificar de manera única cada dispositiu. |
| **nom\_llista** | | El nom que l’usuari li vol donar a la nova llista personalitzada. |
| **Precondicions** | | |
| * L’usuari existeix al sistema. * No hi ha una llista amb el mateix nom al sistema. | | |
| **Postcondicions** | | |
| * El sistema haurà afegit la llista personalitzada i l’haurà vinculat a l’usuari actiu. * El sistema estarà preparat per començar a vincular paraules a la llista acabada de crear. | | |
| **Excepcions** | | |
| * Si el codi d’usuari especificat al paràmetre no existeix, es retornarà un codi d’error (KO-1). * Si el nom de la llista personalitzada és massa llarg, es retornarà un codi d’error (KO-2). * Si el nom de la llista personalitzada ja existeix, es retornarà un codi d’error (KO-3). | | |

1. *JavaScript Object Notation*, sistema lleuger d’intercanvi de dades. Nodejs està implementat en V8 (<http://code.google.com/p/v8/>), que proporciona de forma nativa accés als objectes JSON. [↑](#footnote-ref-1)