

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

1.1. Idea 3

1.2. Possible funcionament 3

1.3. Objectius 4

1.3.1. Objectiu general 4

1.3.2. Objectius específics 4

1.4. Planificació amb fites i temporalització 4

Capítol 2: Estat de l’art 6

Capítol 3: Estudi de mercat 9

3.1. Diccionaris 9

3.2. *Flashcards* 9

3.3. Àudio 10

3.4. Pràctica de Traços 11

Capítol 4: Anàlisi i disseny 12

4.1. Tecnologia 12

4.2. Arquitectura del sistema 14

4.3. Disseny preliminar de les pantalles 15

4.3.1. Llistes d’estudi 15

4.3.2. Diccionari 17

4.3.3. *Flashcards* 19

4.4. Disseny preliminar del mòdul del servidor 21

4.4.1. Avantatges de la implementació de les accions en servidor 21

4.4.2. Inconvenients de la implementació de les accions en servidor 22

4.4.3. El controlador 22

4.4.4. El Model 22

4.4.5. Els objectes del model 22

4.4.6. Emmagatzematge de dades 23

4.4.7. Comunicacions client – servidor 24

4.4.8. Peticions Client – Servidor 25

# Introducció

## 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.

## 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.

## Objectius

### 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.

### 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.

## Planificació amb fites i temporalització

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



1. Diagrama Gantt de planificació

# 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.



1. 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.



1. 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”.







1. 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.

# 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).

## 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.

A continuació podem veure alguns exemples d’aquest tipus d’aplicacions:

 **

1. Pantalles de consulta de definició (esquerra) i cerca d’una paraula (**Japanese for iOS**: <http://japaneseapp.com/>).

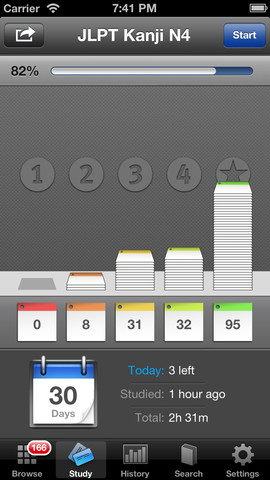
## *Flashcards*

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)

Exemples de pantalles d’aplicacions centrades en l’estudi amb flashcards:

1. Pantalles de pràctica amb flashcards. Captures corresponents a **Japanese Flip** (primera captura, <https://itunes.apple.com/us/app/japanese-flip/id289263209?mt=8>) i **Sticky Study Japanese** (segona i tercera captures, <https://itunes.apple.com/us/app/japanese-flip/id289263209?mt=8>).

## 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.

Podem veure’n alguns exemples:

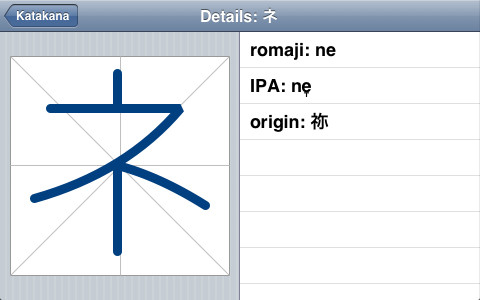
1. Pantalles de pràctica amb àudio. Inclouen tant la possibilitat d’escoltar com sona una paraula, com la possibilitat de gravar la pròpia veu i re-escoltar-la posteriorment. Les captures mostrades corresponen a l’aplicació **Learn Japanese Vocabulary – Gengo** (<https://itunes.apple.com/us/app/learn-japanese-vocabulary/id294770805?mt=8>).

## Pràctica de Traços

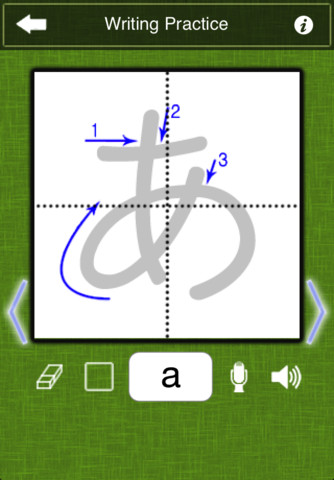
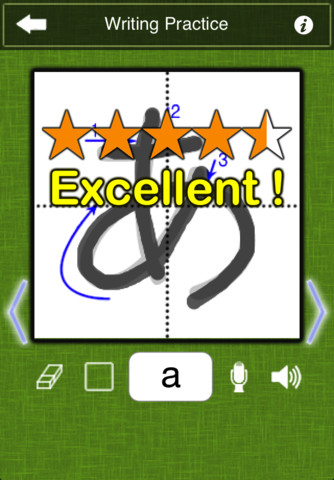
Una característica distintiva de l’idioma japonès és els traços que formen els caràcters. Donat que aquest tret constitueix un àmbit d’estudi en si mateix, és lògic que hi hagi moltes aplicacions que permeten la pràctica dels traços.

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.

Exemples en aquest camp:

1. Pantalles Per la pràctica amb els traços del llenguatge japonès. En aquest cas es l’aplicació es limita a ensenyar l’ordre i manera d’execució dels traços. Captures extretes de l’aplicació **Kana Writing** (<https://itunes.apple.com/us/app/kana-writing/id451464932?mt=8>).

1. Pantalles Per la pràctica amb els traços del llenguatge japonès. Aquesta aplicació va una mica més enllà, i és més interactiva, de manera que avalua la pràctica de l’usuari, posant nota a l’execució dels traços d’un caràcter. Les captures s’han extret de l’aplicació **Kana Strokes** (<https://itunes.apple.com/us/app/kana-strokes-japanese-hiragana/id318485239?mt=8>).

# Anàlisi i disseny

## 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.



1. Framework escollit, el Sencha Touch 2

Per a la part client hem escollit l’entorn de treball **Sencha Architect 2** i utilitzarem els nous estàndards HTML5 i CSS3, que seran utilitzats sobre un servidor web.



1. 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.



1. 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>



1. Sistema de control de versions del nostre projecte

## 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:



1. Model Vista Controlador (MVC) del framework Sencha Touch

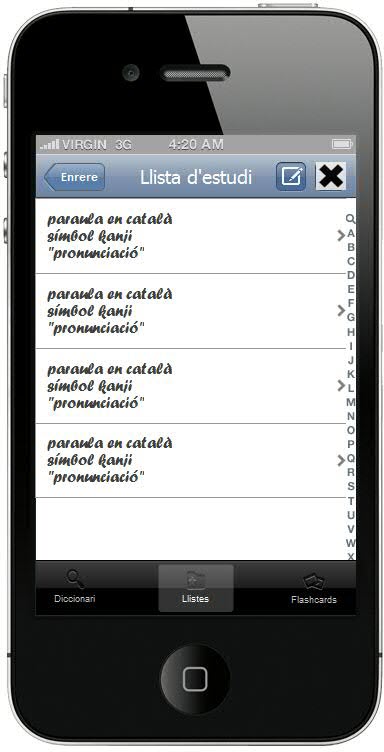
* Una capa **Model** que s’encarrega de la representació especifica de la informació amb la que el sistema opera. Es composa per la lògica de negoci i pel sistema de gestió de base de dades.
* Una capa **Vista** que s’encarrega de la representació del model en un format adequat per interactuar, i que 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.

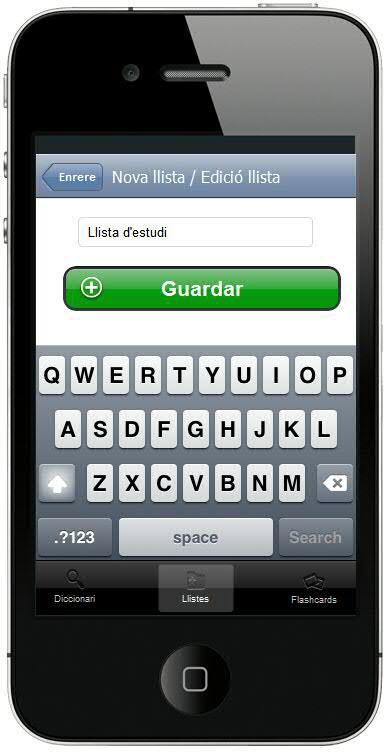
## Disseny preliminar de les pantalles

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 però amb un flux de navegació.

### Llistes d’estudi



1. Pantalla d’inici de l’aplicació, llistes d’estudi, i pantalla detall de la llista

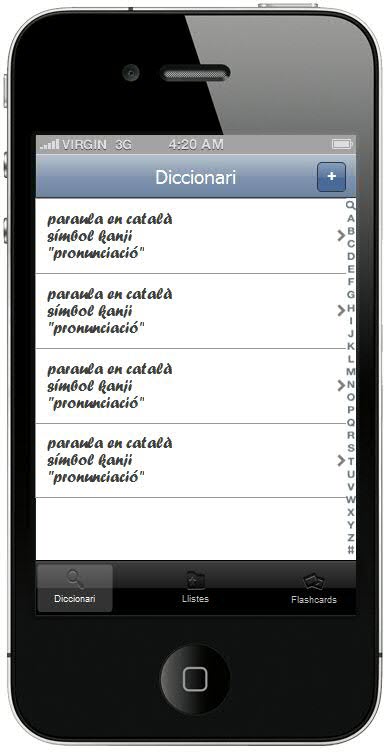


1. Pantalles d’edició i d’esborrat d’una llista

**

1. Pantalla per veure el detall del concepte d’una llista

### Diccionari



1. Pantalles llistat i detall del diccionari

**

1. Pantalles d’edició i de diàleg d’un concepte

### Flashcards

**

1. Pantalla prèvia a l’inici de l’exercici per a escollir una llista

**

1. Pantalles de l’exercici, anvers i revers de la targeta amb el concepte escrit

## 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 certes 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. Això vol dir que certes accions només es podran fer quan el dispositiu estigui connectat a la xarxa.
* 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.

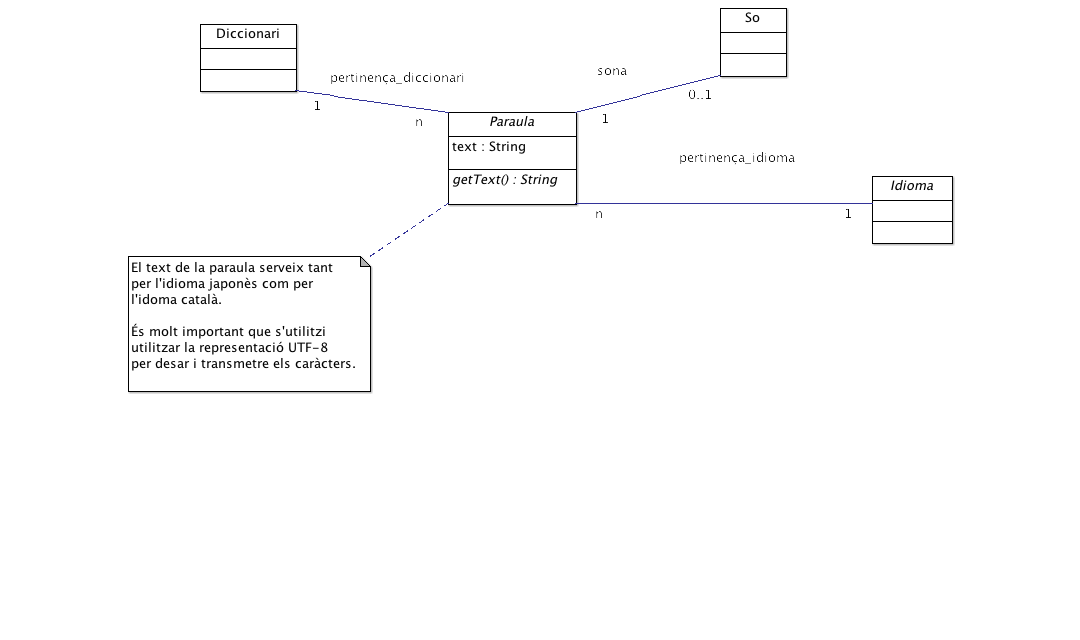
### El Model

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:



1. Diagrama de classes del model del servidor. Diagrama fet amb l’eina ArgoUML (<http://argouml.tigris.org/>).

Descripció breu de cada classe:

* **Paraula**: una paraula, en l’idioma que sigui.
  + S’ha representat com una classe abstracta, de la qual heretaran les paraules concretes en els idiomes que sigui.
  + Conté el mètode abstracte *getText():String*. Aquest mètode retornarà el text de la paraula (variable *text*). Com s’indica al requadre del comentari del diagrama, el text es pot guardar en aquesta variable, tenint sempre en compte d’utilitzar la representació UTF-8.
* **Idioma**: idioma en què està una paraula.
  + S’ha representat com una classe abstracta, de manera que posteriorment s’implementin els idiomes concrets que es necessitin, en el nostre cas Català i Japonès.
* **So**: una representació àudio d’una paraula, en l’idioma que sigui.
  + S’utilitza aquesta classe com una representació a alt nivell. Posteriorment es decidirà per quina implementació s’opta per representar l’àudio, tenint en compte que s’ha d’emmagatzemar i transmetre per la xarxa de manera òptima.
* **Diccionari**: el conjunt de paraules del sistema.

Explicació de les relacions que veiem al diagrama:

* Relació **Idioma – Paraula**: és una relació **1 – n**. Una paraula està sempre en un idioma; un idioma pot tenir qualsevol nombre, positiu o 0, de paraules.
* Relació **Diccionari – Paraula**: és una relació **1 – n**. Una paraula és a un diccionari; un diccionari pot contenir qualsevol nombre, positiu o 0, de paraules.
* Relació **So – Paraula**: és una relació **1 – 0..1**. Una paraula pot, o no, tenir un àudio associat; un àudio correspon sempre a una paraula.

### Emmagatzematge de dades

Pel que fa a l’emmagatzematge de les dades, hem de tenir en compte que hem de guardar dos tipus d’informacions:

* D’una banda, la informació textual de la paraula (representació de text, pertinença al diccionari, idioma).
* De l’altra, la representació en format àudio de la paraula.

La nostra proposta consisteix a utilitzar dos mecanismes diferents per tal d’emmagatzemar aquesta informació:

* Es proposa guardar la informació textual en una base de dades relacional (es proposa el servidor MySQL 5.5 o superior, però es pot avaluar la conveniència d’utilitzar un servidor més lleuger, com ara SQLite).
* Pel que fa a la informació d’àudio, es desarà en el sistema de fitxers. Això proporciona més flexibilitat pel que fa a l’emmagatzematge. Desar fitxers d’àudio a la base de dades pot ser problemàtic, sobre tot en el cas en què vulguem fer migracions a d’altres sistemes, o bé còpies de seguretat. A la base de dades hi guardarem, associada a cada paraula, el nom del fitxer d’àudio i la ruta del sistema de fitxers on s’ha desat l’àudio de la paraula.

Informació que es desarà a la base de dades:

* **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.
  + Els usuaris podran afegir paraules pròpies, de manera que la base de dades tindrà unes paraules predeterminades, i unes altres que cada usuari afegirà.
  + Desarem un enllaç a la representació àudio per saber on es troba ubicada en el sistema de fitxers del servidor.

No s’inclou cap diagrama Entitat / Relació, perquè l’entitat **Paraula** serà l’única que es desarà a la base de dades.

### 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.

D’ara en endavant, parlarem d’un nou terme, **Concepte\_Paraula**. Aquest terme es refereix a tota la informació que envolta una paraula:

* Text en català de la paraula.
* So en català de la paraula.
* Text en japonès de la paraula.
* So en japonès de la paraula.

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

* **crear\_concepte\_paraula**, per donar d’alta una nova paraula al sistema
* **editar\_camp\_concepte\_paraula**, per canviar de valor algun dels camps d’un concepte\_paraula.
* **get\_concepte\_paraula**, per obtenir un concepte\_paraula concret.
* **search\_concepte\_paraula**, per buscar en la llista d’elements concepte\_paraula.
* **get\_sound**, per obtenir el so associat a una paraula

### Peticions Client – Servidor

|  |  |  |
| --- | --- | --- |
| **Nom** | **crear\_concepte\_paraula** | |
| **Descripció** | Acció que s’executa quan un usuari vol donar d’alta una nova paraula al sistema. L’usuari introduirà el text en català i en japonès, i també gravarà (si el dispositiu ho permet) l’àudio associat a cada un dels idiomes.  Tota aquesta informació s’enviarà al servidor, que la desarà si és possible (per exemple, si la paraula ja existeix en català o bé en japonès, no es podrà desar, i es retornarà un codi d’error). | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **text\_catala** | | Text de la paraula en català |
| **text\_japones** | | Text de la paraula en japonès (els *kanji* corresponents) |
| so\_catala | | Àudio de la paraula en català. Opcional. |
| so\_japones | | Àudio de la paraula en japonès. Opcional. |
| **Precondicions** | | |
| * L’usuari ha entrat tots els camps obligatoris. * La paraula no existeix prèviament al diccionari del servidor. | | |
| **Postcondicions** | | |
| * La paraula ha estat donada d’alta al sistema, en els dos idiomes. Si l’usuari també ha enviat la representació en àudio en qualsevol dels dos idiomes (o en els dos) es desarà en forma de fitxer al sistema de fitxers, i la ruta es guardarà juntament amb la paraula a la base de dades. * El servidor retorna un codi de concepte\_paraula (ID) amb el qual s’identificarà a partir d’ara l’entitat creada. | | |
| **Excepcions** | | |
| * Si no tots els camps han estat emplenats, es retornarà un codi d’error al client (KO-1). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **editar\_camp\_concepte\_paraula** | |
| **Descripció** | Acció que s’executa quan un usuari vol canviar el valor d’algun dels camps d’un concepte paraula.  Els camps que es poden canviar amb aquesta acció són: text de la paraula (en català o en japonès); so de la paraula (en català o en japonès). | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **id\_concepte\_paraula** | | El codi del concepte\_paraula que es vol modificar. |
| text\_catala | | Text de la paraula en català |
| text\_japones | | Text de la paraula en japonès (els *kanji* corresponents) |
| so\_catala | | Àudio de la paraula en català. Opcional. |
| so\_japones | | Àudio de la paraula en japonès. Opcional. |
| **Precondicions** | | |
| * La paraula amb l’ID especificat ja existeix al sistema. * Almenys un dels següents paràmetres és diferent de null:   + text\_catala   + text\_japones   + so\_catala   + so\_japones | | |
| **Postcondicions** | | |
| * La paraula ha estat modificada en qualsevol dels seus camps. Si no hi havia cap valor diferent dels ja existents, no es modificarà res. | | |
| **Excepcions** | | |
| * Si no hi ha almenys un dels paràmetres necessaris (textos o sons), es retornarà un codi d’error (KO-1). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **get\_concepte\_paraula** | |
| **Descripció** | Acció que s’executa quan un usuari vol recuperar un concepte\_paraula a partir de l’ID.  Serveix per obtenir tota la informació associada a un concepte\_paraula, és a dir: textos (català i japonès) i sons (català i japonès). | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **id\_concepte\_paraula** | | El codi del concepte\_paraula que es vol modificar. |
| **Precondicions** | | |
| * La paraula amb l’ID especificat ja existeix al sistema. | | |
| **Postcondicions** | | |
| * El concepte\_paraula especificat per l’ID del paràmetre és retornat al client. * No s’ha modificat cap informació al servidor. | | |
| **Excepcions** | | |
| * Si no hi ha el paràmetre necessari (id\_concepte\_paraula), es retornarà un codi d’error (KO-1). * Si no hi ha cap paraula amb l’ID indicat, es retornarà un codi d’error (KO-2). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **search\_concepte\_paraula** | |
| **Descripció** | Acció que s’executa quan un usuari vol cercar paraules. El sistema cerca un text a la base de dades, en un idioma especificat, i retornarà (si és possible) una paraula amb el text que s’està cercant. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **text\_paraula** | | El text de la paraula, en l’idioma concret que indiqui l’usuari, que es vol cercar. |
| **idioma** | | El codi de l’idioma en el qual està escrita la paraula que s’està cercant. |
| **Precondicions** | | |
| * S’han omplert tots els camps necessaris en la petició. * Hi ha una paraula a la base de dades, en l’idioma indicat, que té un text coincident amb el text de cerca introduït per l’usuari. No es tindran en compte majúscules/minúscules. Sí que es tindran en compte els accents, en el cas de l’idioma català. | | |
| **Postcondicions** | | |
| * Es retorna un objecte concepte\_paraula, quan el text d’aquell concepte en l’idioma indicat per l’usuari coincideixi. * No es fa cap modificació a la base de dades. | | |
| **Excepcions** | | |
| * Si no hi ha els paràmetres necessaris, es retornarà un codi d’error (KO-1). * Si no es troba cap paraula coincident, es retornarà un codi d’error (KO-2). | | |

|  |  |  |
| --- | --- | --- |
| **Nom** | **get\_sound** | |
| **Descripció** | Acció que s’executa quan un usuari vol recuperar una representació àudio d’una paraula, en un idioma especificat.  En aquest punt no s’especifica cap format de representació, sinó que ens limitem a indicar la funcionalitat a alt nivell. Deixem per a la fase d’implementació l’elecció del format de les dades àudio. | |
| **Paràmetres (els obligatoris estan en negreta)** | | |
| **id\_concepte\_paraula** | | El codi del concepte\_paraula pel qual l’usuari vol descarregar la representació àudio. |
| **idioma** | | El codi de l’idioma pel so que s’intenta descarregar. |
| **Precondicions** | | |
| * La paraula amb l’ID especificat ja existeix al sistema. | | |
| **Postcondicions** | | |
| * Una representació àudio de la paraula, en l’idioma indicat per l’usuari, és retornat al client. * No s’ha fet cap modificació al servidor (ni a la base de dades, ni al sistema de fitxers). | | |
| **Excepcions** | | |
| * Si no hi ha els paràmetres necessaris, es retornarà un codi d’error (KO-1). * Si no hi ha cap paraula amb l’ID indicat, es retornarà un codi d’error (KO-2). * Si no hi ha representació àudio per la paraula sol·licitada, 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)