PROJECTE FINAL DE CARRERA

MEMÒRIA

APRENENTATGE D'IDIOMES

Alumnes: Capell Brufau, Eduard i Lorca Sans, Salvador

Consultor: Roset Mayals, Roman

CONTINGUT

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 Leitner[1].

[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:

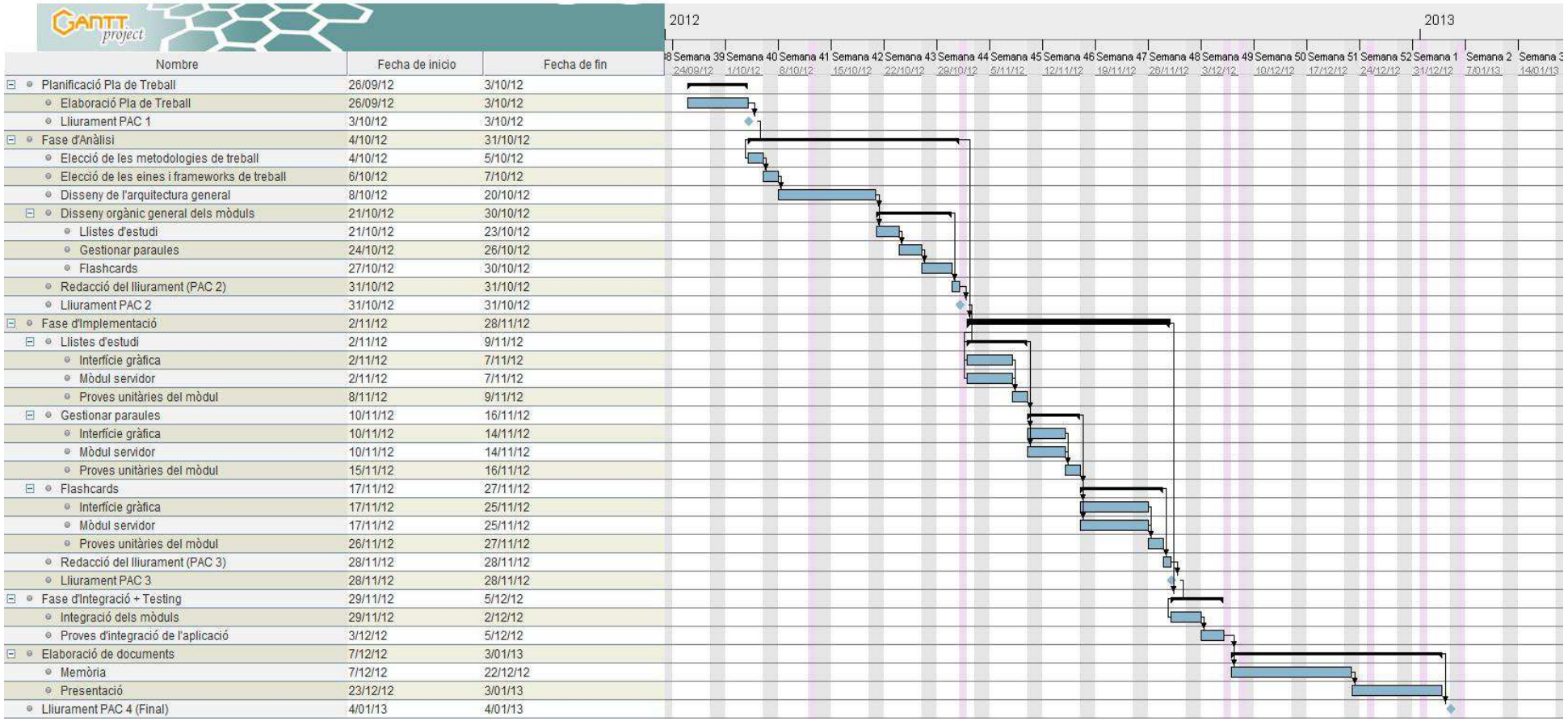
- Lliurar una eina d'aprenentatge que s'adapti a les necessitats dels estudiants que s'inicien en l'aprenentatge de la llengua japonesa.
- Assimilar tots els conceptes i nocions nous que aniran sortint durant el cicle de vida del programari, i més en concret la metodologia que ens portarà a aconseguir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



Nombre	Fecha de inicio	Fecha de fin
Planificació Pla de Treball	26/09/12	3/10/12
Elaboració Pla de Treball	26/09/12	3/10/12
Lliurament PAC 1	3/10/12	3/10/12
Fase d'Anàlisi	4/10/12	31/10/12
Elecció de les metodologies de treball	4/10/12	5/10/12
Elecció de les eines i frameworks de treball	6/10/12	7/10/12
Disseny de l'arquitectura general	8/10/12	20/10/12
Disseny orgànic general dels mòduls	21/10/12	30/10/12
Llistes d'estudi	21/10/12	23/10/12
Gestionar paraules	24/10/12	26/10/12
Flashcards	27/10/12	30/10/12
Redacció del lliurament (PAC 2)	31/10/12	31/10/12
Lliurament PAC 2	31/10/12	31/10/12
Fase d'Implementació	2/11/12	28/11/12
Llistes d'estudi	2/11/12	9/11/12
Interfície gràfica	2/11/12	7/11/12
Mòdul servidor	2/11/12	7/11/12
Proves unitàries del mòdul	8/11/12	9/11/12
Gestionar paraules	10/11/12	16/11/12
Interfície gràfica	10/11/12	14/11/12
Mòdul servidor	10/11/12	14/11/12
Proves unitàries del mòdul	15/11/12	16/11/12
Flashcards	17/11/12	27/11/12
Interfície gràfica	17/11/12	25/11/12
Mòdul servidor	17/11/12	25/11/12
Proves unitàries del mòdul	26/11/12	27/11/12
Redacció del lliurament (PAC 3)	28/11/12	28/11/12
Lliurament PAC 3	28/11/12	28/11/12
Fase d'Integració + Testing	29/11/12	5/12/12
Integració dels mòduls	29/11/12	2/12/12
Proves d'integració de l'aplicació	3/12/12	5/12/12
Elaboració de documents	7/12/12	3/01/13
Memòria	7/12/12	22/12/12
Presentació	23/12/12	3/01/13
Lliurament PAC 4 (Final)	4/01/13	4/01/13

Il·lustració 1. *Diagrama Gantt de planificació*

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 un ordinador o un tablet Android, 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.

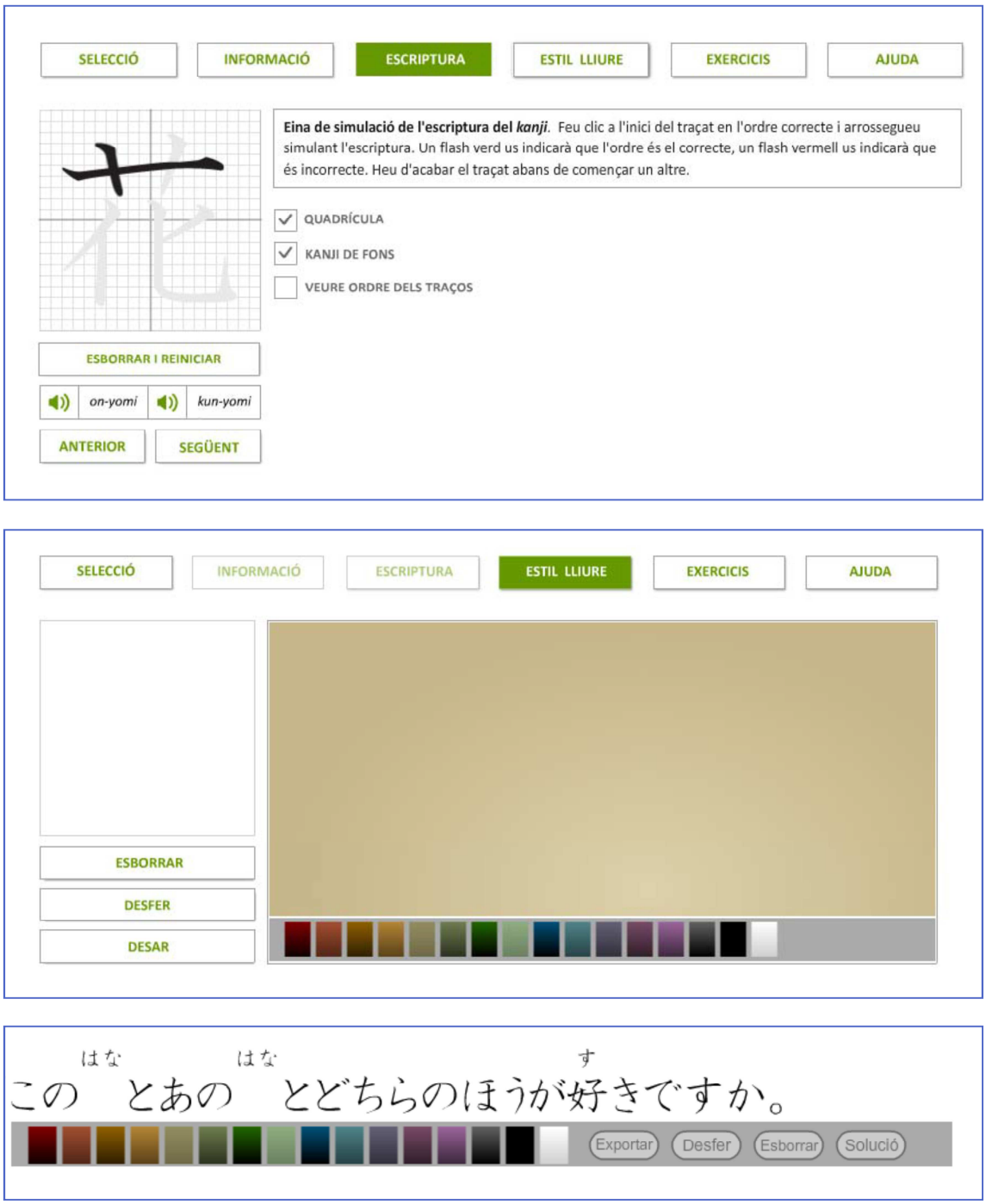
Il·lustració 2. *Selecció d'imatges kanji en el QVE*

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.

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.

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

Il·lustració 5. *Pantalles de consulta de definició (esquerra) i cerca d'una paraula (**Japanese for iOS**: http://japaneseapp.com/).*

3.2 *FLASHCARDS*

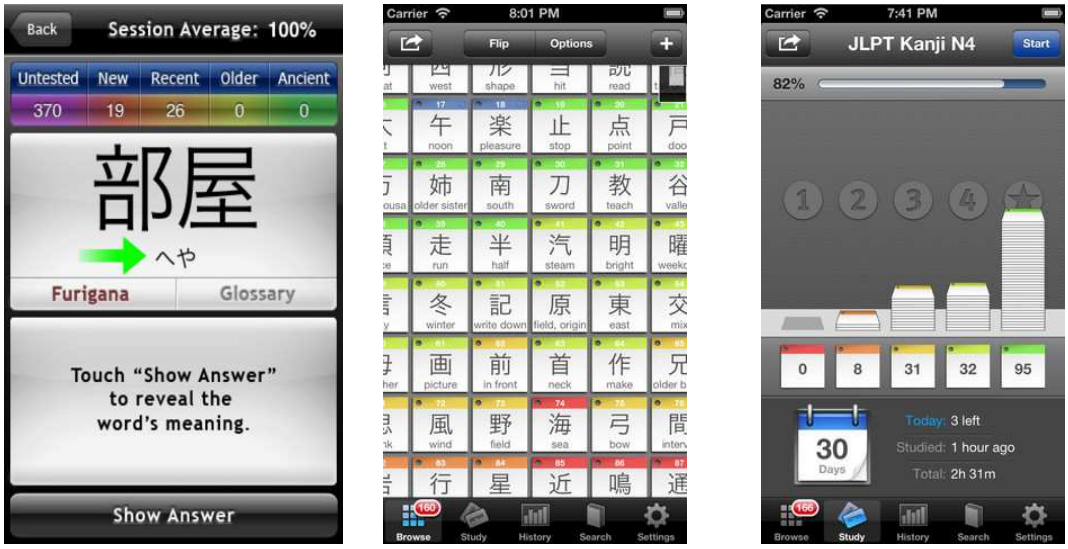
Aquesta és una categoria molt freqüent en les aplicacions per l'estudi d'idiomes. Consisteix en una simulació de targetes didàctiques. A una banda de la targeta hi ha una

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)

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



Il·lustració 6. *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).*

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.

Podem veure'n alguns exemples:

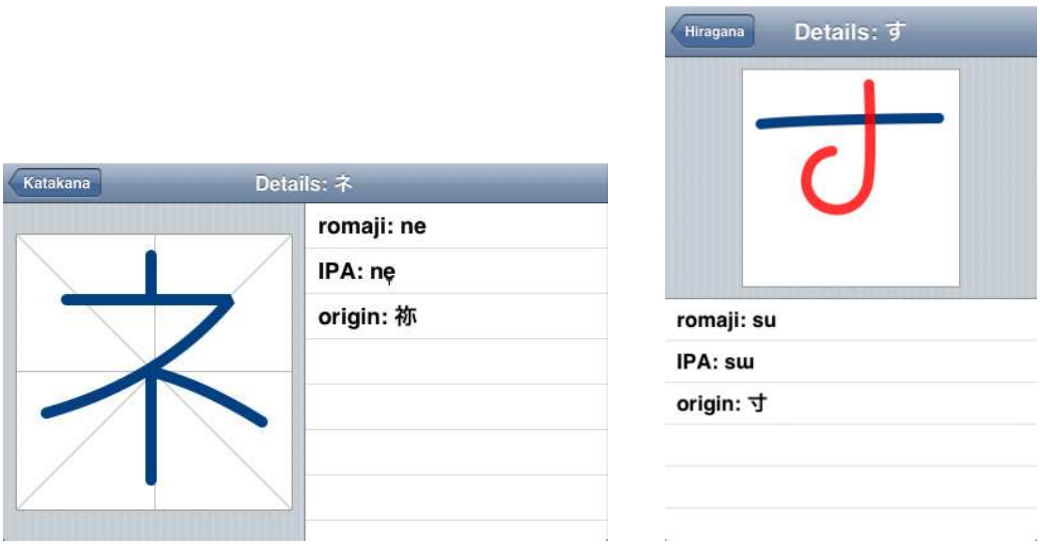
Il·lustració 7. *Pantalles de pràctica amb àudio. Inclouen tant la possibilitat d'escoltar com sona una paraula, com la possibilitat de gravar la pròpia veu i re-escoltar-la posteriorment. Les captures mostrades corresponen a l'aplicació* ***Learn Japanese Vocabulary – Gengo*** *(https://itunes.apple.com/us/app/learn-japanese-vocabulary/id294770805?mt=8).*

3.4 PRÀCTICA DE TRAÇOS

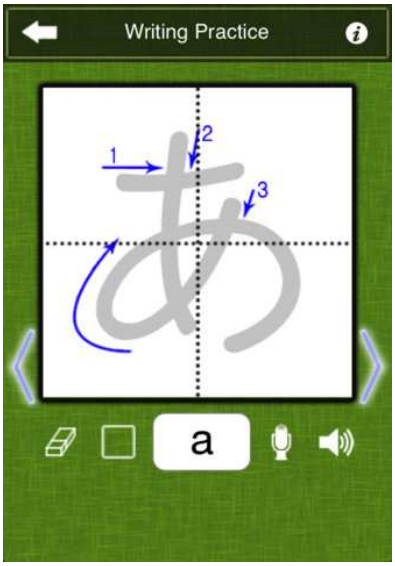
Una característica distintiva de l'idioma japonès és els traços que formen els caràcters. Donat que aquest tret constitueix un àmbit d'estudi en si mateix, és lògic que hi hagi moltes aplicacions que permeten la pràctica dels traços.

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.

Exemples en aquest camp:



Il·lustració 8. *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).*



Il·lustració 9. *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).*

4 METODOLOGIA DE TREBALL

Per fer aquest treball ens hem basat en una metodologia que comparteix origen amb el llenguatge dels *kanjis*, el japonès. Es diu *Kanban*[2], i és un sistema de producció que dispara treball només quan existeix capacitat per processar-ho. El "disparador de treball" és representat per targetes *Kanban* de les quals es disposa d'una quantitat limitada. Cada targeta *Kanban* acompanya a un ítem durant tot el procés de producció, fins que aquest, és empès fora del sistema, alliberant una targeta. Un nou ítem de treball, solament podrà ser ingressat/acceptat si es disposa d'una targeta *Kanban* lliure.

No obstant això, en la pràctica, *Kanban* no es limita a una etiqueta (targeta). Aquesta targeta no serviria de molt si no s'apliqués d'acord a certs principis i regles. Amb tan sol tres simples regles, *Kanban* demostra ser una de les metodologies adaptatives que menys resistència al canvi presenta. Aquestes són les tres regles que hem seguit durant el desenvolupament del projecte final de carrera:

1. Visualització en tot moment dels processos que feim els dos companys.
2. Limitar el treball en curs, és a dir, acordar prèviament el nombre d'ítems que es poden abordar en cada moment.
3. Optimitzar el flux de treball.

Il·lustració 10. *Exemple de panell Kanban*

[2] *Kanban és un terme que pot traduir-se com a etiqueta o tiquet d'instrucció.*

5 ANÀLISI I DISSENY

5.1 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 els sistemes operatius iOS o Android, i en dispositius tan variats com BlackBerry, Kindle Fire, iPad, Nexus 7, etc.

Il·lustració 11. *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.

Il·lustració 12. *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 13. *Servidor escollit, pur JavaScript*

La base de dades que utilitzarem per guardar certs objectes serà MySQL (versió 5.5 o superior).

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:

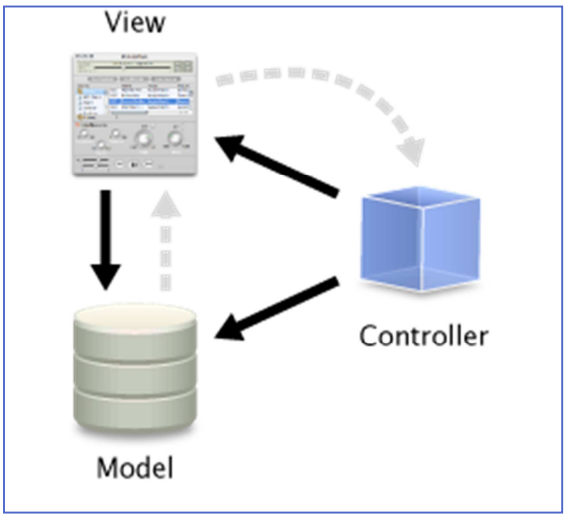
https://github.com/salvinha/UOCPFC_Eduard_Salva.git

Il·lustració 14. *Sistema de control de versions del nostre projecte*

5.2 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 15. *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.

5.3 DISSENY PRELIMINAR DEL MÒDUL CLIENT

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.

5.3.1 LES PANTALLES DEL CLIENT

5.3.1.1 LLISTES D'ESTUDI

Il·lustració 16. *Pantalla d'inici de l'aplicació, llistes d'estudi, i pantalla detall de la llista*

Il·lustració 17. *Pantalles d'edició i d'esborrat d'una llista*

Il·lustració 18. *Pantalla per veure el detall del concepte d'una llista*

5.3.1.2 DICCIONARI

Il·lustració 19. *Pantalles llistat i detall del diccionari*

Il·lustració 20. *Pantalles d'edició i de diàleg d'un concepte*

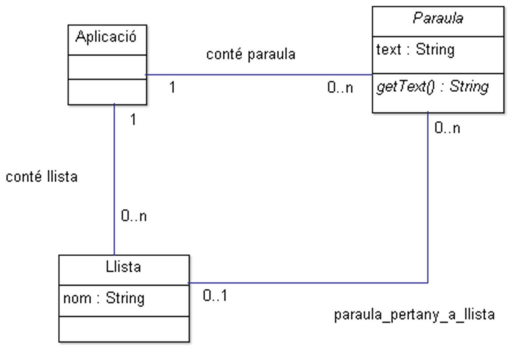
5.3.1.3 *FLASHCARDS*

Il·lustració 21. *Pantalla prèvia a l'inici de l'exercici per a escollir una llista*

Il·lustració 22. *Pantalles de l'exercici, anvers i revers de la targeta amb el concepte escrit*

5.3.2 ELS OBJECTES DEL MODEL

En el següent diagrama hi podem veure els objectes bàsics del model que es faran servir per la implementació del client:



Il·lustració 23. *Diagrama de classes del model del client*

Descripció breu de cada classe:

- **Aplicació**: és una entitat que engloba tota la resta d'objectes que s'utilitzaran en la aplicació.
- **Paraula**: és una classe abstracta que denota el que coneixem com a *concepte_paraula*, és a dir: text en català i japonès, so en català i japonès.
- **Llista**: és un conjunt de paraules agrupades sota un mateix nom.

Explicació de les relacions que veiem al diagrama:

- **conté_llista**: és una relació **1 – n**, indicant que l'aplicació pot contenir qualsevol nombre de llistes, incloent-hi cap llista.
- **conté_paraula**: és una relació **1 – n**, indicant que l'aplicació pot contenir qualsevol nombre de paraules, incloent-hi cap paraula.
- **paraula_pertany_a_llista**: és una relació **n – 0,1**, indicant que una llista pot contenir qualsevol nombre de paraules (incloent-hi 0), i una paraula pot estar en una o cap llista.

5.3.3 PETICIONS CLIENT – CLIENT

Les peticions Client – Client són aquelles que no surten del dispositiu de l'usuari. En aquest cas no és necessari un motor tipus nodejs, sinó que n'hi haurà prou de programar un mòdul dins del Sencha que atengui aquestes peticions i retorni les respostes apropiades.

En tot cas, l'estructura segueix sent la mateixa que si fossin peticions Client – Servidor: hi ha una petició original, amb un nom i uns paràmetres, es fa la petició, es duu a terme la tasca corresponent, i es retornen uns resultats.

5.3.3.1 PETICIONS RELACIONADES AMB LES LLISTES

En el cas de les llistes, les peticions que conté l'aplicació són:

- **crear_llista**, per crear una nova llista.
- **get_llista**, per obtenir una llista, amb les corresponents paraules.
- **get_llistes**, per obtenir una llista amb totes les llistes creades per l'usuari.
- **editar_camp_llista**, per canviar alguna propietat de la llista.
- **afegir_concepte_llista**, per afegir paraules a una llista.
- **esborrar_llista**, per eliminar una llista existent.
- **esborrar_concepte_llista**, per treure una paraula d'una llista.

Nom	**crear_llista**
Descripció	Aquesta funció permet de crear una nova llista buida, amb el nom especificat per paràmetre.
Paràmetres (els obligatoris estan en negreta)	
nom_llista	El nom de la nova llista
Precondicions	
• No hi ha al sistema una llista amb el mateix nom.	
Postcondicions	
• S'ha creat una nova llista buida, amb el nom indicat. • Es retornarà el codi de la nova llista creada.	
Excepcions	
• Si no hi ha el paràmetre necessari, es retornarà un codi d'error (KO-1). • Si una llista amb el mateix nom ja existeix, es retornarà un codi d'error (KO-2).	

Nom	**get_llista**
Descripció	Aquesta funció retornarà una llista de paraules, a partir d'un ID de llista especificat per paràmetre.
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que vol obtenir l'usuari.
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema.	
Postcondicions	
• Es retorna un objecte de tipus Llista, que conté les paraules que hi pertanyen. • No s'ha modificat cap objecte (paraules o llistes).	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si no hi ha cap llista amb l'ID especificat, es retornarà un codi d'error (KO-2).	

Nom	**get_llistes**
Descripció	Aquesta funció retornarà el conjunt de llistes que hi ha al dispositiu de l'usuari.
Paràmetres (els obligatoris estan en negreta)	
No hi ha paràmetres	
Precondicions	
• Hi ha una o més llistes definides al sistema.	
Postcondicions	
• Es retorna un conjunt de llistes. • No s'ha modificat cap objecte (paraules o llistes).	
Excepcions	
• Si la crida conté paràmetres, es retornarà un codi d'error (KO-1). • Si no hi ha cap llista, es retornarà un codi d'error (KO-2).	

Nom	**editar_camp_llista**
Descripció	Aquesta funció permetrà, donat un camp i un nou valor, modificar alguna de les propietats de la llista. Tal com s'ha dissenyat l'aplicació, només hi ha un camp rellevant (nom de la llista), però es deixa la funció oberta per tal que pugui suportar més camps en el futur.
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que es vol modificar.
nom_camp	Camp que es vol modificar. Camps suportats: • Nom de la llista (*nom_llista*, de tipus text)
nou_valor	Nou valor que prendrà el camp que s'està modificant.
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema. • El camp que es vol modificar és vàlid (actualment, l'únic camp admès és *nom_llista*. • El valor anterior a la modificació és diferent del nou valor.	
Postcondicions	
• S'ha modificat el valor del camp de la llista amb l'ID especificat, amb el valor nou substituint l'antic. • Les paraules que pertanyien a la llista hi segueixen associades, sense que s'hagi de	

fer cap acció addicional. • Si tot ha anat correctament, es retornarà el codi OK.
Excepcions
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si el camp especificat no és vàlid, es retornarà un codi d'error (KO-2). • Si el nou valor és igual que el ja existent, es retornarà un codi d'error (KO-3).

Nom	**afegir_concepte_llista**
Descripció	Aquesta funció permetrà, donat un codi de concepte_paraula i una llista, associar la paraula i la llista en qüestió.
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que es vol modificar.
id_concepte_paraula	Camp que es vol modificar. Camps suportats: • Nom de la llista (*nom_llista*, de tipus text)
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema. • La paraula amb el codi de concepte indicat existeix al sistema. • La llista i la paraula indicades no estan prèviament associades.	
Postcondicions	
• S'ha creat una associació entre la llista i la paraula, de manera que ara aquesta paraula pertany a la llista indicada. • Si tot ha anat correctament, es retornarà el codi OK.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si la llista no existeix, es retornarà un codi d'error (KO-2). • Si la paraula no existeix, es retornarà un codi d'error (KO-3). • Si la paraula i la llista ja estan prèviament associades, es retornarà un codi d'error (KO-4).	

Nom	**esborrar_llista**
Descripció	Aquesta funció permetrà, a partir d'un codi de llista indicat, esborrar aquesta llista del dispositiu de l'usuari. Les paraules que hi pertanyin deixaran d'estar-hi associades, però seguiran existint, sense estar associades a la llista esborrada (podran seguir pertanyent a d'altres llistes o bé no pertànyer a cap).
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que es vol esborrar.
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema.	
Postcondicions	
• S'ha esborrat la llista del sistema de l'usuari. • Les paraules que pertanyien a aquesta llista segueixen existint. • Si tot ha anat bé, es retornarà un codi OK.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si no hi ha cap llista amb el paràmetre indicat, es retornarà un codi d'error (KO-2).	

Nom	**esborrar_concepte_llista**

Descripció	Aquesta funció permetrà, donada una llista i un concepte_paraula, desfer l'associació entre l'una i l'altre. D'aquesta manera, la paraula en qüestió deixarà d'estar associada a la llista. Un cop finalitzada l'operació, tant la llista com la paraula continuen existint al sistema, simplement s'ha desfet l'associació entre elles.
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que es vol modificar.
id_concepte_paraula	El codi de la paraula que es vol esborrar de la llista.
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema. • El concepte_paraula amb l'ID especificat ja existeix al sistema. • Hi ha una associació entre la paraula i la llista indicades.	
Postcondicions	
• S'ha desfet l'associació entre la llista i la paraula indicades. • La paraula i la llista continuen existint, però ja no estan relacionades. • Si tot ha anat correctament, es retornarà el codi OK.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si la llista especificada no existeix, es retornarà un codi d'error (KO-2). • Si la paraula especificada no existeix, es retornarà un codi d'error (KO-3). • Si la paraula i la llista no estaven associades, es retornarà un codi d'error (KO-4).	

5.3.3.2 PETICIONS RELACIONADES AMB FLASHCARDS

En el cas de les flashcards, les peticions que hi intervenen són les següents:

- **init_boxes**, per inicialitzar les caixes.
- **get_card**, per obtenir la propera targeta d'una caixa.
- **go_on**, per mostrar la següent targeta quan l'usuari n'encerta una.
- **go_back**, per mostrar la targeta anterior quan l'usuari n'erra una.

Nom	**init_boxes**
Descripció	Aquesta petició s'encarrega de posar tots els conceptes d'una llista passada per paràmetre a la caixa inicial, i també d'inicialitzar les altres dues caixes (deixar-les sense conceptes o targetes). Després de fer aquesta inicialització es farà una petició al mètode get_card per obtenir la propera targeta de la caixa inicial (serà la primera de totes).
Paràmetres (els obligatoris estan en negreta)	
id_llista	El codi de la llista que volem usar.
Precondicions	
• La llista amb l'ID especificat ja existeix al sistema.	
Postcondicions	
• Els conceptes de la llista es col·loquen en la caixa inicial, la de l'esquerra, i s'extreu la primera targeta de la caixa.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si no hi ha cap llista amb el paràmetre indicat, es retornarà un codi d'error (KO-2).	

Nom	**get_card**
Descripció	Aquesta petició s'encarrega d'obtenir la targeta disponible de la caixa

	amb paràmetre id_box. En el cas que no hagin targetes disponibles a la caixa s'haurà de fer una altra petició a la caixa posterior/anterior, depenent de si ens trobem a la caixa inicial o a una caixa intermèdia.
Paràmetres (els obligatoris estan en negreta)	
id_box	El codi de la caixa de la qual es vol obtenir la següent targeta.
Precondicions	
• La caixa amb ID igual al paràmetre ha de ser una caixa existent.	
Postcondicions	
• Visualitzem la targeta disponible de la caixa. • Si les caixes inicials i intermèdies no contenen targetes s'haurà acabat el joc.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1). • Si no hi ha cap caixa amb el paràmetre indicat, es retornarà un codi d'error (KO-2). • Si no es troben targetes a qualsevol caixa que no sigui la final s'acaba el joc (OK)	

Nom	**go_on**
Descripció	L'usuari quan clica el botó "He encertat" el que fa es llançar la següent petició per a què el sistema mogui la targeta actual a la següent caixa i després faci un get_card de la caixa actual.
Paràmetres (els obligatoris estan en negreta)	
id_concepte_paraula	La paraula (flashcard) que l'usuari ha encertat.
Precondicions	
• El concepte_paraula indicat al paràmetre existeix al sistema • El concepte_paraula indicat al paràmetre no està a la caixa final.	
Postcondicions	
• La targeta ocupa un lloc a la següent caixa.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1).	

Nom	**go_back**
Descripció	L'usuari quan clica el botó "He fallat" el que fa es llançar la següent petició per a què el sistema mogui la targeta actual a la caixa inicial i després faci un get_card de la caixa actual.
Paràmetres (els obligatoris estan en negreta)	
id_concepte_paraula	La paraula (flashcard) que l'usuari ha fallat.
Precondicions	
• El concepte_paraula indicat al paràmetre existeix al sistema • El concepte_paraula indicat al paràmetre no està a la caixa final.	
Postcondicions	
• La targeta ocupa un lloc a la caixa inicial.	
Excepcions	
• Si no hi ha els paràmetres necessaris, es retornarà un codi d'error (KO-1).	

5.4 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 inconvenients, que mostrem a continuació.

5.4.1 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.

5.4.2 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.

5.4.3 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.

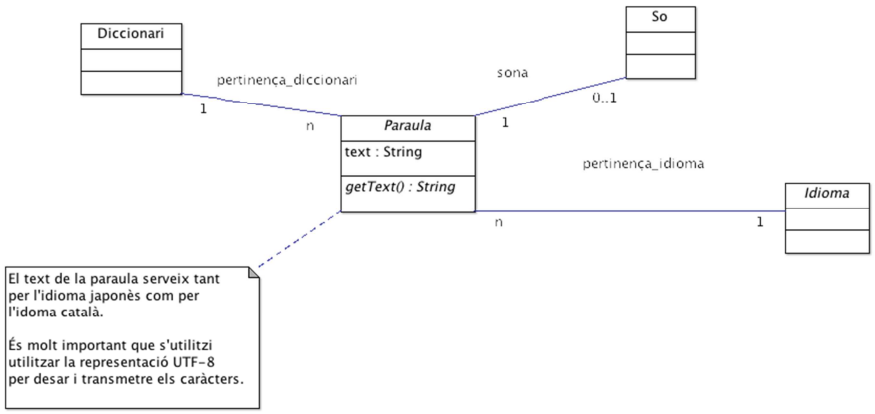
5.4.4 EL MODEL

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.

5.4.5 ELS OBJECTES DEL MODEL

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



Il·lustració 24. *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.

5.4.6 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.

5.4.7 COMUNICACIONS CLIENT – SERVIDOR

Cada comunicació entre client i servidor es farà mitjançant peticions JSON[3]. 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

5.4.8 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.	

[3] *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.

• 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: o text_catala o text_japones o so_catala o 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).	