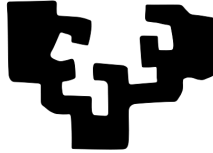


eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Facultad de Informática

Informatika Fakultatea

Ingeniaritza Informatikan
An Open and Social Spaced Repetition System
for Language Learning

Ander Martínez Sánchez

Supervisors:

Iñaki Alegria Loinaz

Montse Maritxalar Anglada

Acknowledgment

I would like to thank Prof. Iñaki Alegria and Prof. Montse Maritxalar for their mentoring and supervision.

I would also like to thank my parents for their patience and allowing me to skip some of my obligations during this work. Thanks go for my family and friends too for all the time I owe them now.

Thanks for all the community supporting Free and Open Source software, for without their work this project would be yet another assembler or something. Thanks for making computing awesome.

Contents

List of figures	10
List of tables	11
1 Introduction	13
1.1 Abstract	13
1.2 Laburpena	15
1.3 Motivation	17
2 Goals of the project	19
2.1 Goals	19
2.2 Tasks	20
2.3 Work breakdown structure	22
2.4 Time estimation	22
2.5 Methodology	24
3 Analysis	25
3.1 Licensing	25
3.2 Spaced repetition systems	26
3.3 Platform	27
3.4 Evaluation	28
3.5 State of the art	28
3.5.1 Comparison	36
3.6 Technologies	38
3.6.1 Mercurial	39

3.6.2	sphinx	39
3.6.3	Python	40
3.6.4	Django	40
3.6.5	HTML5 and ECMAScript	42
3.6.6	Dojo Toolkit	43
3.6.7	FreeBSD	43
3.6.8	PostgreSQL	44
4	Design	45
4.1	Architecture	45
4.2	Items	48
4.2.1	Item types	52
4.3	Client	56
4.4	Server	60
4.4.1	Apps	61
4.4.2	Scheduler	63
4.4.3	API	64
4.5	Importing data	66
4.6	Collaboration model	67
4.7	Socialization	68
5	Implementation	71
5.1	Resulting product	71
5.1.1	Registering as a new user	71
5.1.2	Searching a lesson	72
5.1.3	Creating a new lesson	73
5.1.4	Commenting	75
5.1.5	Studying	76
5.2	Deployment	77
6	Conclusion	79
A	Licenses	83
A.1	Leizea license	83

Contents	7
<hr/>	
A.2 Document license	84
A.3 Content license	85
References	97

List of Figures

2.1	Work Breakdown Structure diagram	22
3.1	Flashcards used for studying vocabulary.	27
3.2	SuperMemo2 featuring the SM2 algorithm.	29
3.3	Homepage of <i>smart.fm</i> while still working. <i>Source: wikipedia.</i> . .	30
3.4	Anki study session. The system asks you how well you knew the answer.	32
3.5	Mnemosyne study session. Evaluation is manual. <i>Source: mnemosyne- proj.org</i>	33
3.6	Memrise study session with automatic evaluation.	34
3.7	OpenCards study session. Evaluation is manual. <i>Source: wikipedia</i>	35
3.8	Dutch course in wikibooks.	36
3.9	Babelium practice exercise. <i>Source: babeliumproject</i>	37
4.1	Architecture of the system in the current deployment.	46
4.2	Different use cases.	47
4.3	A modifiable field.	57
4.4	Study session.	59
4.5	Apps used by the leizea server.	62
4.6	Simplified behaviour of the scheduler.	63
4.7	Fork and pull collaborative model.	68
5.1	Registering as a new user.	72
5.2	Browsing lessons by language.	73
5.3	Search lesson by keywords.	73

5.4	Creating a new lesson	74
5.5	Adding a new item to a lesson.	75
5.6	Adding an explanation to a lesson.	76
5.7	Adding an explanation to a lesson.	76
5.8	Current production server at home. Acer Aspire One 110L.	78

List of Tables

2.1	Time estimation of the tasks	22
3.1	Non-technical comparison of solutions	38
3.2	Technical comparison of solutions	38
4.1	how to create items	48
5.1	Details on the server	77

Chapter 1

Introduction

1.1 Abstract

In the last decades big improvements have been done in the field of computer aided learning, based on improvements done in computer science and computer systems. Although the field has been always a bit lagged, without using the latest solutions, it has constantly gone forward taking profit of the innovations as they show up. As long as the train of the computer science does not stop (and it won't at least in the near future) the systems that take profit of those improvements will not either, because we humans will always need to study; Sometimes for pleasure and some other many times out of need.

Not all the attempts in the field of computer aided learning have been in the same direction. Most of them address one or some few of the problems that show while studying and don't take into account solutions proposed for some other problems. The reasons for this can be varied. Sometimes the solutions simply are not compatible. Some other times, because the project is an investigation it's interesting to isolate the problem. And, in commercial products, licenses and patents often prevent the new projects to use previous work.

The world moved forward and this is an attempt to use some of the options offered by technology, mixing some old ideas with new ones.

Many times students take private lessons, despite knowing they could study by themselves, thinking they will avoid laziness and will be more persevering or because making a schedule for studying takes a lot of time and if they take private lessons they get it done. A learning system should to schedule the lessons for the user, for this takes most time from the student. For this, something called **spaced repetition** [T06] has been used before, which calculates the time optimal that should pass between reviews of a concept.

A learning system also needs to keep the attention of the user, as long as possible. One tactics to accomplish this is to present the study as a game. The **gamification** is introducing elements from games (like points or levels) into serious applications. For example, a lot of sites give *reputation* (sometimes called *karma*) or *badges* to their users as a prize, and then they show a “list of the top ten users with more points” somewhere. This kind of things motivate some students that want to go higher on that list.

Nowadays anybody has a computer, but not only that, a lot of people have more than one computer: be it at home, at work, the mobile phone, etc. This **ubiquity** is linked by Internet, which gives us the chance to have our data accessible anywhere, without any risk of losing them or forgetting them somewhere. Besides avoiding these risks it gives us the chance to **socialize** our studying making use of the other users to collect material (**collaboration**) and making the learning funner, staying in contact with other students.

The socialization has a very big importance specially in language learning because it can connect the student with native speakers of the language he is trying to learn. Without Internet the student had to go to some place where they spoke the language he was studying, if he wanted to move forward on his study.

To finish, lately the **open licenses** have been gaining terrain and, as seen in many projects, they give better results than models based on closed licenses. Open licenses not only for software, which is quite common, also for the contents. People is ready to work for free or to share work they’ve already done if they guarantee that they work will remain free. Take a look at **Wikipedia**, for which both the software and the contents are completely free and yet it is one of the top ten most

visited websites: making use of the collaboration it has promoted free contents. Knowledge and studying should be free, because the freedom of people depends on that as well, the equality in opportunities, above the economic situation of the individuals.

Keywords: *spaced repetition, gamification, ubiquity, socialization, collaboration, open license, Computer-assisted language learning (CALL)*

1.2 Laburpena

Azken hamarkadetan aurrera pausu handiak egin dira konputagailuz lagundutako ikasketa sisteman, informatikaren esparruan egindako aurrerakuntzez lagunduz; beti zertxobait atzeragotik, punta-puntako soluzioak erabili gabe, baina etengabe aitzinerantz aukera berriak aurkeztu orduko. Informatikaren trena geldituko ez den moduan (ez behintzat etorkizun hurbilean) bera erabiltzen duten sistemak ere ez dira urrunduko, gizakiok ikasteko premia baitugu; batzuetan atseginez eta beste askotan halabeharrez.

Konputagailuz lagundutako ikasketaren historian zehar egin diren saiakera guztiak ez dira beti norabide berdinean izan. Gehienegok ikasterakoan agertzen diren arazoetatik bat ala gutxi batzuk konpontzen edo txikitzen jartzen baitute arreta, aurretik egindako urratsak baztertuz. Honen arrazoiak era askotakoak litezke. Kasu batzuetan ebazpideak ez dira bateragarriak, bestetan ikerkuntzak direlako ezauzgarri bakar batzuk ikertzea baino ez da interesgarri eta produktu komertzialetan, lizentziak edo patenteak beste produktu batzuetan oinarritzea ekiditen dute.

Munduak aurrera egin du eta teknologiak eskaintzen dizkigun aukerak erabiltzeko saiakera da hau, ideia zahar batzuk berriekin nahastuz.

Askotan ikasle asko klase partikularretara eta joaten dira, beraien kabuz ikasi ahal dutela jakinagatik ere, horrela utzikeria ekiditen dutelakoan edo ikasteko plangintza egiteak denbora asko hartzen duelako eta klasetara joanda eginda jasotzen dutelako. Ikasketa sistema batek ikasgaien plangintza egin behar dio erabiltzaileari, honek hartzen baitu ikaslearen denbora gehien. Horretarako **errepikatze**

tartekatua [T06] deitzen den sistema erabili izan da aurretik, kontzeptu baten berrikuspenen artean iragan behar den denbora tarte hoberena kalkulatzear arduratzen dena.

Gainera, ikasteko sistema batek ikaslearen atentzioari eutsi behar dio, ahal den neurrian. Horretarako erabili daitekeen estrategia bat ikasketa joko moduan azaltzea izan daiteke. **Gamification**-a (joko bihurtzea) jokoen elementu batzuk (adibidez puntuak edo mailak) aplikazio serioetan txertatzean datza. Adibidez, gune askok *ospea* (batzuetan *karma* deitzen da) edo *dominak* ematen dizkiete erabiltzaileei saritzat, gero, nonbaiten, “puntu gehien dituzten hamar erabiltzaileen zerrenda” eta antzekoak erakusteko. Honelakoek erabiltzaileak motibatzen dituzte.

Egun edonork du ordenagailua, baina ez hori bakarrik, jende askok ordenagailu bakarra baino gehiago du: etxekoa dela, lanekoa, telefono mugikorra, etab. **Non-ahikotasun** honen lotura den Internetek, gure datuak nonahi atzigarri izateko aukera ematen digu, galtzeko edo nonbaiten ahazteko arriskuak gaindituz. Perilok ekiditeaz gain Internetek gure ikaskuntza **sozializatze**ko aukera ematen digu beste ikasleez baliatuz ikasmateriala biltzeko (**lankidetz**a) eta ikasketa erakargarriago egiteko, beste ikasleekin harremanetan jarritz.

Hizkuntzen ikasketan bereziki, berebiziko garrantzia du sozializazioak, ikasi behar den hizkuntzaren jatorrizko hiztunekin harremanetan jarri baitezake ikaslea. Internetik gabe ikasleak hizkuntza hitz egiten zuten herrietara joan beharra zuen ezinbestean, ikasketan aitzina egin nahi bazuen.

Bukatzeko, azkenaldian **lizentzia irekiak** indarra dute eta, zenbait proiektutan ikusi denez, lizentzia itxietan oinarritutako ereduak baino emaitza hobeak dituzte. Lizentzia irekiak ez soilik softwarean, nahiko ohikoa den bezala, baita edukietan ere. Jendea prest dago doan lan pixka bat egiteko edo jada egindako lana partekatzeko beren lana libre iraungo duela bermatzen badiote. Horra *Wikipedia*, bai software zein edukiak guztiz libre izanik munduko hamar webgune bisitatuenetakoa dena: lankidetzaz baliatuz eduki libreak sustatu ditu. Jakintza eta ikasketak libreak izan beharko lirateke, horren baitan baitago gizakion aukera berdintasuna, norberaren egoera ekonomikoaz gaindi.

Hitz gakoak: *errepikatze tartekatu, gamification, nonahikotasun, sozializazio,*

lankidetzza, lizentzia ireki, Ordenagailuak Lagunduriko Hizkuntzen Ikasketa (CALL)

1.3 Motivation

At the moment of writing this proposal, there are several free and non-free solutions providing some similar features to those described here but unfortunately not all of them at the same time and most of them fall in the category of non-free.

I have a strong belief that knowledge and the learning technology should always remain **free** for everybody. Not only free as in *free beer*, which is a strong point (because the people who need these resources most are precisely those without money) but also as in *freedom* permitting contributors to improve the learning system and learning material.

The purpose of this project is thus to develop a completely *free and open source* spaced repetition learning system, that could be further improved by contributors. One of the goals is to make those contributions easy.

This project was also brought up by personal motivations. Being a human language enthusiast I have used different spaced repetition systems for many years now and I have moved from one system to another as I found drawbacks to them. Being dissatisfied with the current choices, this project aims to create a tool I personally need.

Chapter 2

Goals of the project

2.1 Goals

The goal of this project is to plan (and also develop to some extent) an easily extendable community driven language learning system, meeting the requirements listed here.

The system should be fully **Free and Open Source**. The resulting software is planned to be released under BSD License, although it could change to GPL if necessary (for example if it uses some GPLv3 library). One reason for this is to allow people to extend it for making it usable for studying other kind of items, like those not language related which I might not be able to do myself (like, for example, chemistry related items).

To encourage developers to extend the system it should be highly **modular**, specially the the part where different item types can be added. The system should include some basic item types for studying, like vocabulary or characters.

Also the approach should be general enough for the system to be used to study material not related to languages but that require many data to be memorized like history or chemical formulas.

There should be an **API** allowing external applications create lessons and items

automatically, like after parsing a dictionary or a text.

Apart of everything mentioned the basic functionality for the system should be developed, like registering new users and creating lessons. Also some **social** trait should be included, like commenting on lessons and items.

The reasons for this have already been stated in this document in the *Motivation* section and also in the *Abstract* so I refer the eager reader to those sections.

2.2 Tasks

The project has been divided in five stages or tasks that should be executed in sequential order. Each of this task has also been divided in several sub-tasks that don't need to be carried out sequentially.

Initiate These tasks should be done before starting with the work on the project.

Three sub-tasks are necessary at this stage, namely:

- Write a project proposal which should be accepted in order to continue with the project.
- Collect the basic requirements for the project, the final features of the developed product delivered at the end of the project is highly variant and very dependent on external factors, but a small subset of essential features must be defined.
- Some related documents and articles that will set the intellectual and scientific basis for the project and should be found in order to begin the work. Afterwards more documents will be added to the bibliography but some are necessary to undertake the design part of the project.

Design The design stage is the task when all the important planning of the project happens. First I shall make a rough design of the system, like what modules will compose it and what technologies or libraries should be used, in order to accomplish the requirements established in the previous stage. Then a

detailed design of all the components will be done, fixing all the interfaces and sub-elements.

The Work Breakdown Structure and its diagram is also part of this stage.

Implementation At this stage the system designed in the previous one will be implemented. The system should follow strictly the structure and interfaces described and if the need arises the design should be updated to match the implementation. Two layers will be implemented.

Server The RESTful API designed will be implemented and some other additional views like the homepage and some listings too.

Client The main app (the studying session app) will be developed and some basic styling (mainly layout relate) will be done.

Deployment Deploying the already implemented server and client will also be considered part of the project. For that purpose a domain name needs to be bought and a server configured.

Finalization At this stage the code and feature-set will freeze and I will analyze the results, and perform all the tasks that could not be undertaken while the requirements were fluctuating.

- Present the status of the project and introduce some possible future enhancements that have not been developed for lack of time.
- Draw some conclusions about what has been accomplished or learned by means of this project.
- Review all the documentation for imprecise or outdated information. Fill all the missing gaps.

Closer Before submitting the definitive document it must be formatted correctly according to the conventions¹ established by the *UPV/EHU*. Also prepare a presentation for the project.

¹ <http://www.sc.ehu.es/siwebso/Alumnos/PFC/MemoriaPFC.rtf>

2.3 Work breakdown structure

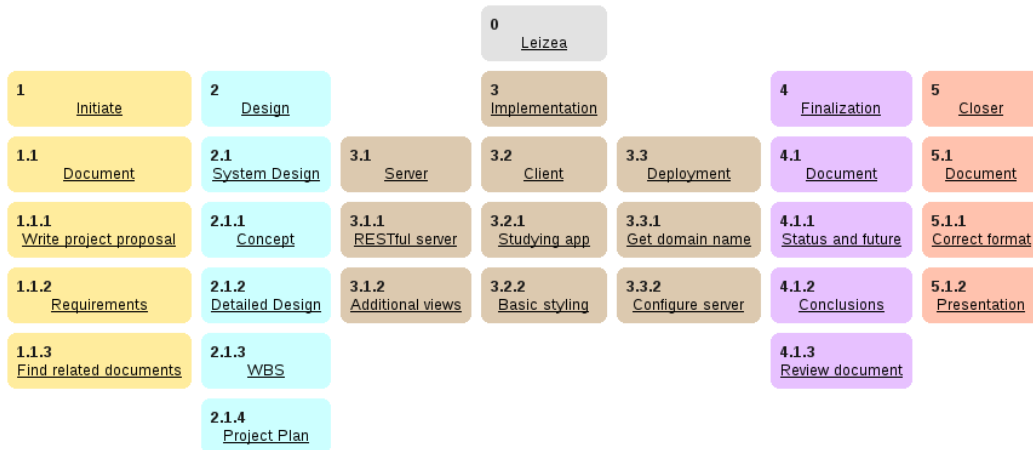


Figure 2.1: Work Breakdown Structure diagram

2.4 Time estimation

The next table shows the time estimated necessary to complete each of the tasks the non-leaf tasks are the addition of all their leaves and are displayed like *this* in the table.

Table 2.1: Time estimation of the tasks

Id	Task	Time (hours)
0	Leizea	384
1	Initiate	24
1.1	Document	24
1.1.1	Write project proposal	6
1.1.2	Requirements	4
1.1.3	Find related documents	14
2	Design	82
Continued on next page		

Table 2.1 – continued from previous page

Id	Task	Time (hours)
2.1	System Design	82
2.1.1	Concept	8
2.1.2	Detailed Design	44
2.1.3	WBS	6
2.1.4	Project Plan	24
3	Implementation	222
3.1	Server	<i>120</i>
3.1.1	RESTful server	80
3.1.2	Additional views	40
3.2	Client	92
3.2.1	Studying app	80
3.2.2	Basic styling	12
3.3	Deployment	<i>10</i>
3.3.1	Get domain name	1
3.3.2	Configure server	9
4	Finalization	<i>34</i>
4.1	Document	<i>34</i>
4.1.1	Status and future	16
4.1.2	Conclusions	14
4.1.3	Review document	4
5	Closer	22
5.1	Document	22
5.1.1	Correct format	6
5.1.2	Presentation	16

The proof-of-concept implementation will take most of the time. This is so because that same code will be extended in the future to hold a full set of features, so good programming practices and careful design is taken into account.

2.5 Methodology

This project will be carried out as described in the section *Tasks*, divided in five separate stages that should be undertaken one after another.

Chapter 3

Analysis

In this section I try to first research the different features that I found interesting for the project.

Then I analyze some systems that fulfill some of these features, but never all of them. This analysis is summarized in a table in the end of the subsection *State of the art*.

To close the section I review the different technologies that I chose, and how they help to build the system.

3.1 Licensing

Licensing plays an important role on the success of a project. Computer Aided Language Learning (CALL) systems use different licenses for the software and for the learning material used by the software.

Open licenses are a strong point for this project. They encourage individuals to participate, improving or creating content and technologies. Users normally don't want to give out their work for free if a company is going to benefit from it.

They also make sure the continuation of the project and its flexibility. If a project is discontinued by its developers it can be continued by another group, or if it's

taken into a different direction, the original spirit can be kept by forking it.

Making the content open helps adapting it to other systems and formats. For example, language teaching material could be printed in a book or exported for its usage in other software.

Socially speaking, it guarantees that underprivileged people will have access to the content and technology.

3.2 Spaced repetition systems

Spaced repetition is a method that is based in scheduled repetition of learning material. It takes longer to learn new information this way but on the other hand its memory lasts longer. Because of its characteristics it's best suited for the acquisition of vocabulary, but it's not necessarily restricted to that area.

Spaced repetition was first explored by Professor Cecil Alec Mace in his book *Psychology of Study* in 1932, based on previous studies of what is called *spacing effect*. According to this phenomenon it is easier to learn or remember items when they are repeated spaced in a long period of time rather than in a short period.

In the sixties and early seventies the idea of spaced repetition was further explored measuring the optimal times for repetitions. These *spaced repetition systems* used *flashcards* (cards with a question on one side and an answer overleaf).

Also in the sixties and early seventies *Paul Pimsleur* devised the *Pimsleur method* which consist of audio recordings with words and their translation repeated at certain fixed intervals. Those intervals where 5 seconds, 25 seconds, 2 minutes, 10 minutes, 1 hour, 5 hours, 1 day, 5 days, 25 days, 4 months, 2 years in his 1967 memory schedule.

SuperMemo was one of the first computer aided spaced repetition systems. In its subsequent versions it has used different versions of its *supermemo algorithm* for scheduling the items which have inspired other different algorithms in other software solutions. This system is later explained in the *State of the art* subsection.



Figure 3.1: Flashcards used for studying vocabulary.

One of the most important parts of a spaced repetition system is the method used to schedule the items. Most SRS software use the *SM2* algorithm or some variation of it.

3.3 Platform

In the *State of the art* subsection software for the desktop and web applications are collected. One of the benefits of web applications over desktop applications is that they make synchronization of study data easier, although some of the desktop applications listed also provide methods to synchronize the study records. Web applications work in a wider range of platforms, as nowadays smart phones, tablets and other devices are common and most of them include a full-fledged web browser. Also they can more easily boost collaboration and socialization of the users.

3.4 Evaluation

Some SRS with *SuperMemo* scheduler algorithms require you to evaluate how well you remembered an item from 0 to 5 (or some other ranges sometimes). This is not the best solution because the users need not only to remember the item they are asked for, but also to think how well they did it after they saw the answer. Further, the distinction between the different *marks* is not always clear. Some other systems assess the answers translating them to scores in that range. To decide the score they use how good the answer was (was there any orthography mistake or was it completely wrong), how long it took for the user to answer, etc.

3.5 State of the art

Here we do an overview of some alternatives and projects that were taken into account or inspiration while designing this project. Not all of them are alternatives to this project, some of them are complementary or are listed here because they have some relevance for us.

The previously described traits are analyzed for all of the solutions here listed:

- Supermemo
- Anki
- Mnemosyne
- smart.fm
- iknow.jp
- memrise
- opencards
- wikibooks
- babelium project

After all the descriptions in this section there is a table that summarizes the key points for each of these.

SuperMemo [<http://www.supermemo.com>]

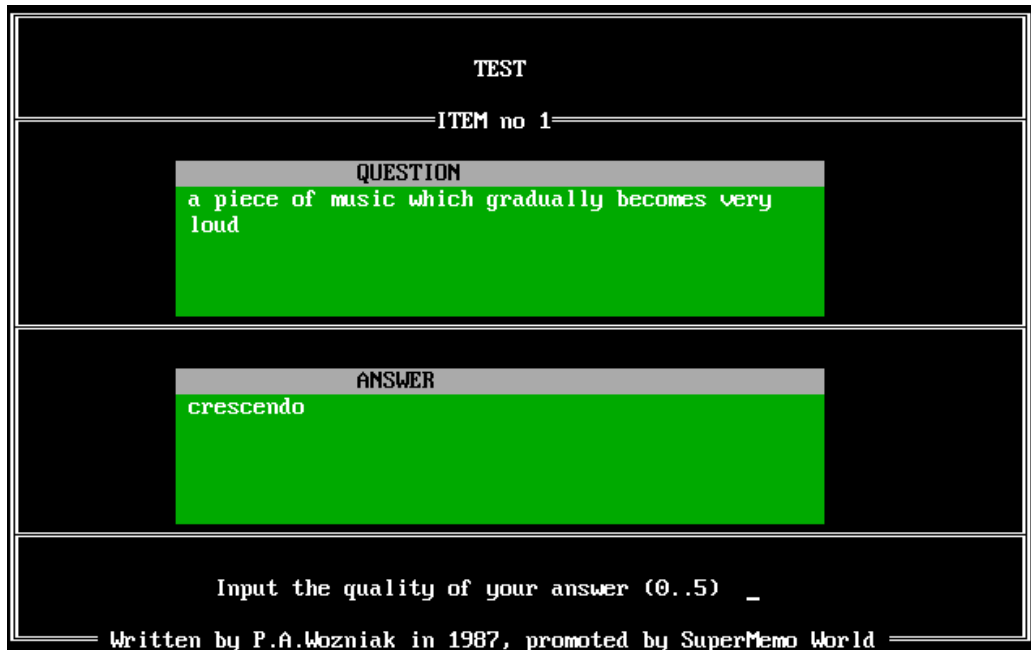


Figure 3.2: SuperMemo2 featuring the SM2 algorithm.

Pioneering commercial solution for a SRS (Spaced Repetition System) authored by Dr Edward Jacek Gorzelanczyk and Dr Piotr Wozniak. The authors have published multiple documentation and algorithms later used by other solutions.

In 1985 the algorithm SM-0 was designed to be used in spaced repetition without computer, employing cards. In 1987 SuperMemo 1.0 was published for DOS using the SM-2 algorithm, later used in SuperMemo 2.0 and SuperMemo 3.0. This algorithm is the most popular in other systems for its simplicity.

Both, the software and the contents, have closed license. The last stable release was in 2011, SuperMemo 15 for Windows 2000 or later

and it costs \$60¹, at the time of writing this.

In its last versions its scheduling algorithm is strongly based on statistic data collected for years from users of the application, which is a strong point. More on the differences of the SM2 (used by leizea) and the SM15 (used by the last version of supermemo) can be seen in *supermemo*'s website².

Still, one has to decide how well they remember an item, which is a burden specially if the grading goes from 0-5, not only pass/fail.

smart.fm [<http://smart.fm>] and **iknow.jp** [<http://iknow.jp>]

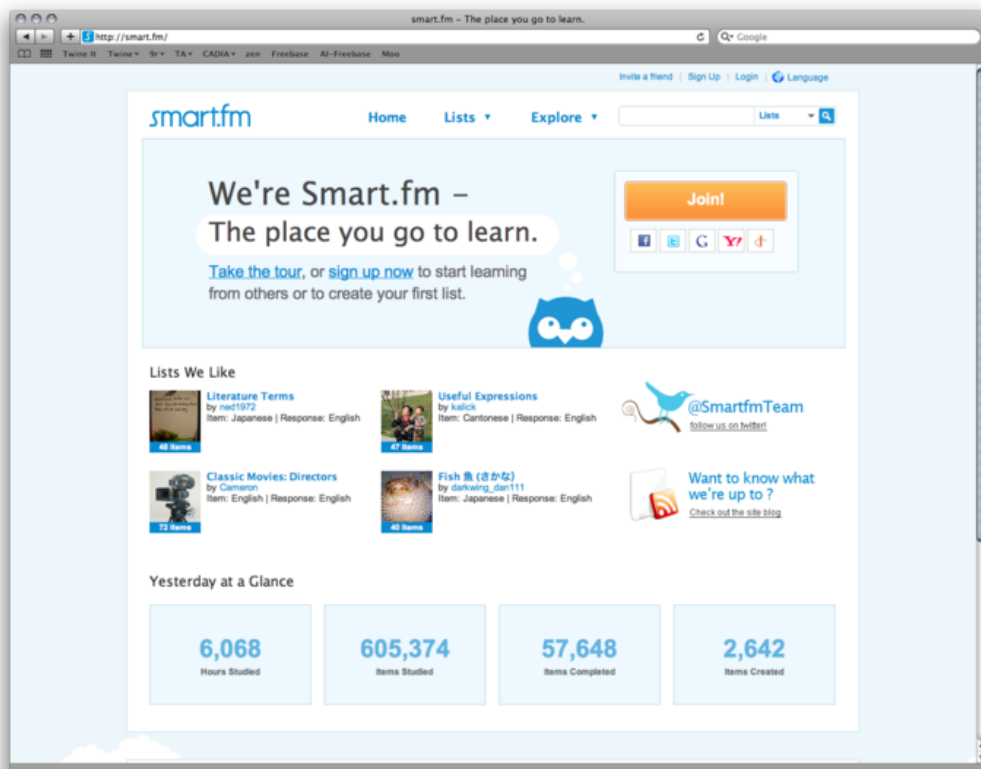


Figure 3.3: Homepage of *smart.fm* while still working. *Source: wikipedia.*

smart.fm was a *spaced repetition* community website by *Cerego Japan*,

¹ See <http://www.supermemo.com/english/which.htm>

² See <http://www.supermemo.com/help/smalg.htm>

Inc.. It was free of use but proprietary, also the lessons. The website was closed and replaced by the paid service **iknow.jp**.

iknow.jp also by Cerego, lacks the social part of *smart.fm* and is pay-for-use. One strong point of **smart.fm** was its social part. In **iknow.jp** the official courses and lessons are created by *Cerego* and although users can still create their own lists the site does not encourage users to sharing.

One weak point of both systems is that all the items are considered words, although you can select the *part of speech* (adjective, noun) this word is. You can also add sentences to the words to create extra questions on them.

iknow.jp focuses only in teaching Japanese, Chinese and English.

Smart.fm introduced a very important feature which this project takes which is automatic evaluation of the answers, without the need of the intervention of the user.

The studying interface of the proposed project is strongly inspired by the one in *smart.fm*.

anki [<http://ankisrs.net>]

It's a flashcard program using an scheduling algorithm based on SM2³ (SuperMemo2) algorithm. It is released under GPLv3 but the server for data syncing is proprietary. The card sets are owned by their creators with various licenses but many of them are distributed from Anki's website.

Anki is available for *Windows*, *Mac OS X*, *FreeBSD*, *Linux*, *iPhone*, *android* and *maemo*, among others. Including different features in some platforms. The server provides a syncing mechanism, which comes handy if you are studying sometimes on your computer and sometimes on your phone, for example.

³ http://ankisrs.net/docs/FrequentlyAskedQuestions.html#_what_spaced_repetition_algorithm_does_anki_use



Figure 3.4: Anki study session. The system asks you how well you knew the answer.

It collects some statistics on each item set and can display them like plots of the progress over time.

The grading is manual, and the user has to decide, after the answer has been shown, how well he recalled the item from zero to five.

mnemosyne [<http://www.mnemosyne-proj.org>]

Mnemosyne is another flashcard spaced repetition system, similar to *anki*.

Cards in **mnemosyne** are better organized than in *anki*. There are different card types with different fields, and they are saved in a more portable format.

It also displays some statistics like plots of the progress over time but these are less rich than those of *anki*.

It uses a variation of the SM2 algorithm and is released under GPLv2

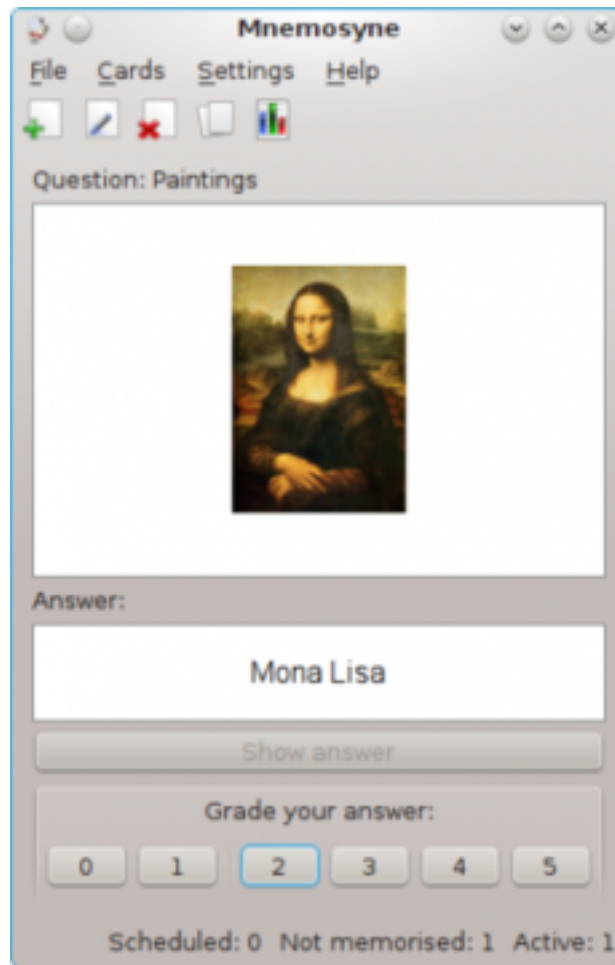


Figure 3.5: Mnemosyne study session. Evaluation is manual. *Source: mnemosyne-proj.org*

but no data synchronization is done.

memrise [<http://www.memrise.com/home/>]

Memrise is a very similar project to the one proposed. It uses javascript instead of flash, which makes it usable from a wider range of platforms.

Although it is closed source it is probably using some variation of SM2, or spaced repetition algorithm, in any case.

It fulfills most of the requirements for this project, but it is closed

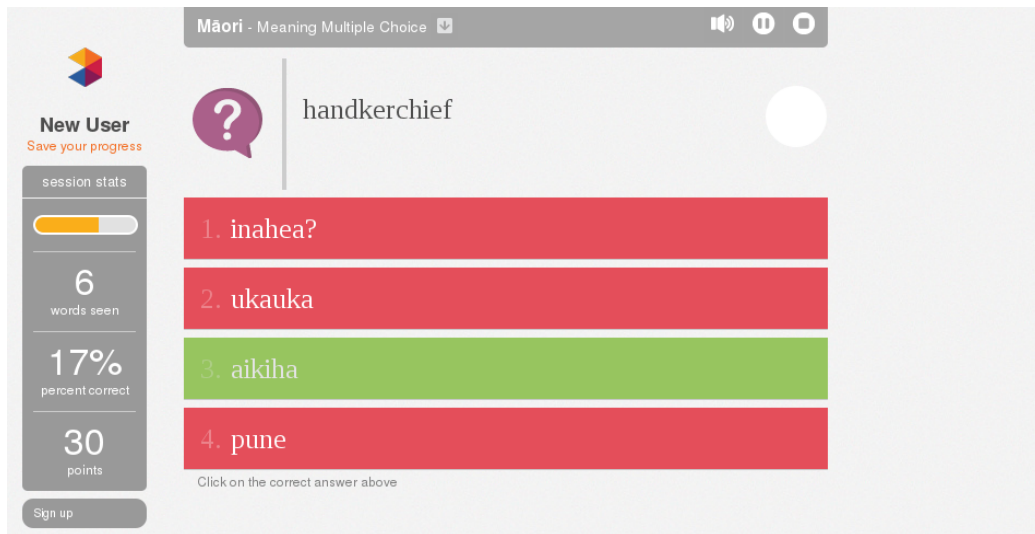


Figure 3.6: Memrise study session with automatic evaluation.

source and takes the ownership of the lessons published by its users⁴. For this reason the project does not guarantee the continuity and freedom of its contents.

OpenCards [<http://opencards.info>]

OpenCards is yet another *spaced repetition system*, with the distinction that it uses *PowerPoint presentation (.ppt)* files as flashcards and as such it makes easier the manual creation of items and allows a wide variety of formats.

On the other hand it's harder to create files from other programs, like when you want to parse a file for items.

It is released under BSD license.

wikibooks [wikibooks.org]

Wikibooks is a wiki using *MediaWiki*, the same software as Wikipedia, hosted by the Wikimedia Foundation.

Its aim is the creation of free content textbooks and annotated texts.

⁴ <http://www.memrise.com/terms/>

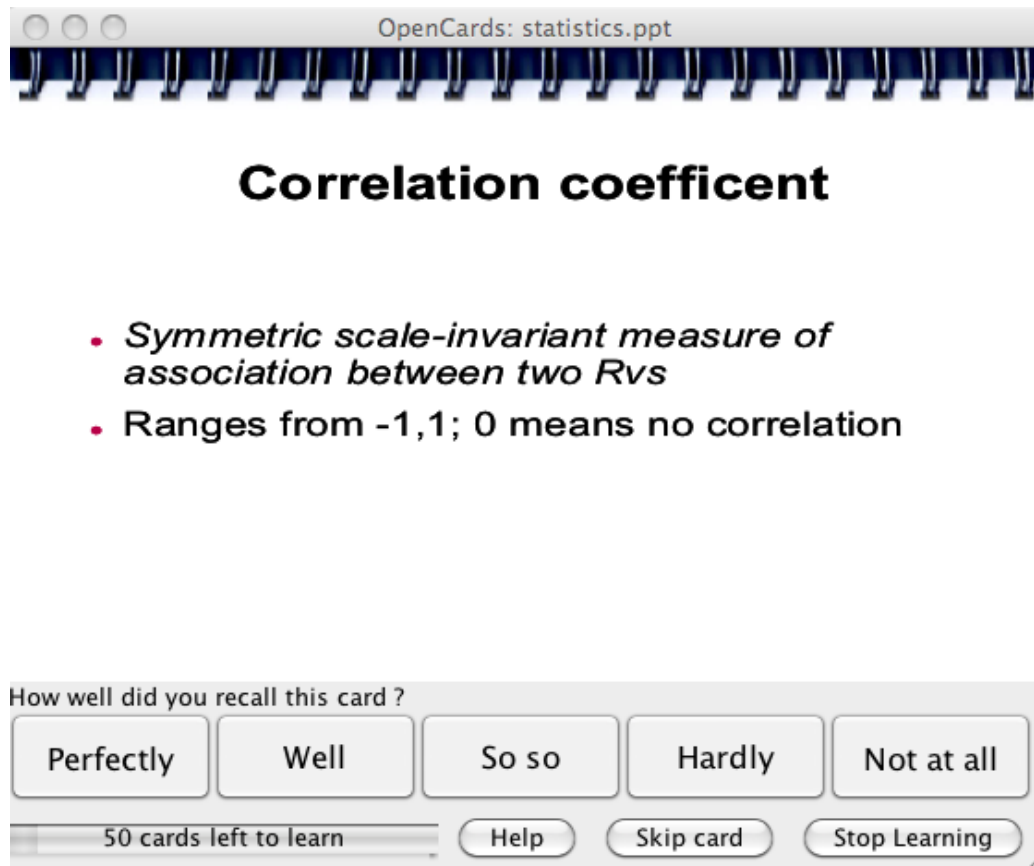


Figure 3.7: OpenCards study session. Evaluation is manual. *Source: wikipedia*

It hosts, among other things, textbooks for learning languages under a *Creative Commons Attribution-Share Alike* license.

Although the project collects a lot of free material for students, the format in which this is saved is suboptimal for studying.

This project takes the idea of the free content and the collaboration from Wikibooks and adds a learning system and a better organization for the courses.

Babeliumproject [<http://babeliumproject.com>]

Babelium Project is another web application for studying languages developed by the research group GHyM of the UPV/EHU.

Figure 3.8: Dutch course in wikibooks.

Babelium takes a different approach to the learning of languages from the one taken by this project. *Babelium* is designed for practicing languages in which the user already has some proficiency and deals mainly with the fluency of the student. Like the proposed system it also relies heavily in the collaboration of its members and the community effect.

Babelium could be a good accompaniment for the students using *leizea*.

It is released under GPLv3.

3.5.1 Comparison

The projects described previously are compared in two subsequent tables. The first one, *Non-technical comparison of solutions*, compares the different licensing for the software and for the contents they provide. If the contents are not part of the system, for example as in desktop applications, *not specified* is shown in the table as the contents are licensed differently by their different authors.

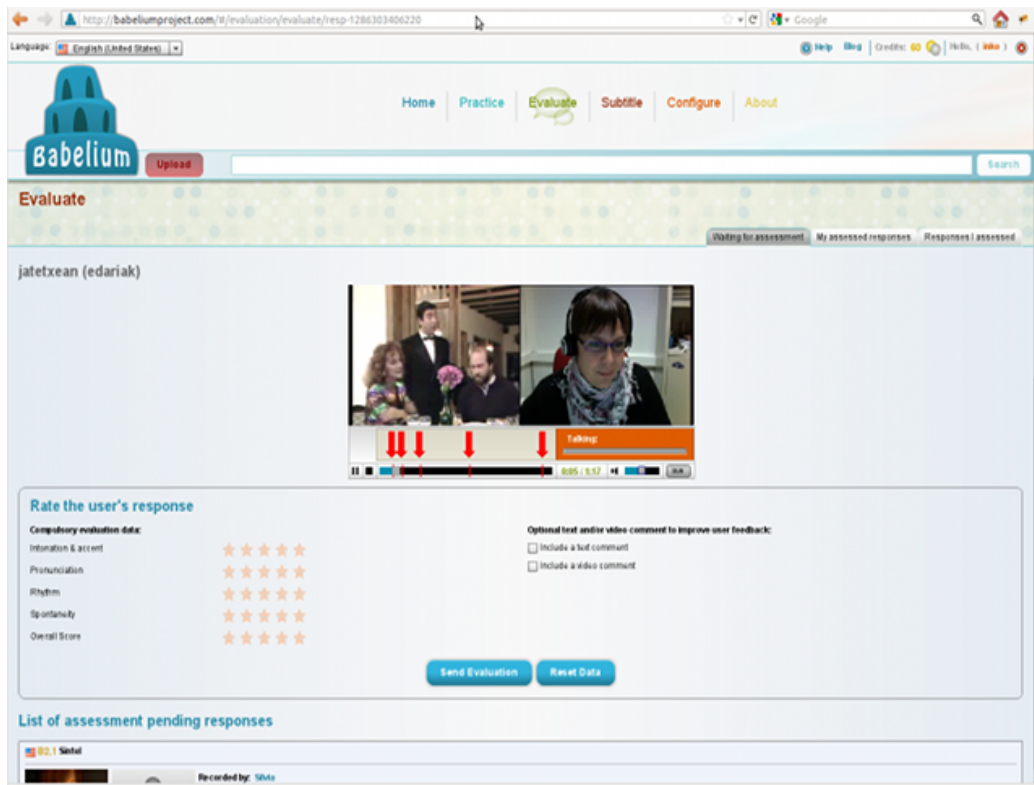


Figure 3.9: Babelium practice exercise. *Source: babeliumproject*

The table shows the following fields:

- **License:** The license for the software.
- **Content:** The license under which the content or lessons are released.
- **Notes:** Other comments.

The second table, *Technical comparison of solutions*, summarizes the more technical aspects of those projects mainly related to implementation. It shows the following fields.

- **Scheduling:** The algorithm used by the scheduler, if any.
- **Evaluation:** Whether the evaluation is done automatically or the users need to grade themselves.

Table 3.1: Non-technical comparison of solutions

Solution	License	Content	Notes
SuperMemo	proprietary	private	\$60 (last ver.)
Anki	GPLv3, proprietary server	not specified	N/A
mnemo syne	GPLv3	not specified	N/A
smart.fm	proprietary	private	not running
iknow.jp	proprietary	private	¥1000 a month
memrise	proprietary	private	N/A
opencards	BSDL	not specified	Uses .ppt files
wikibooks	GPLv2	GNU FDL	Not a learning system
babelium project	GPLv3	free choice	Developed in the EHU
leizea	BSDL	CC-by	This project

Table 3.2: Technical comparison of solutions

Solution	Platform	Scheduling	Evaluation
SuperMemo	desktop	SM15	manual
Anki	desktop	based on SM2	manual
mnemosyne	desktop	SM2	manual
smart.fm	website	custom	automatic
iknow.jp	website	custom	automatic
memrise	website	custom	automatic
opencards	Desktop	SM2, last-minute learning	manual
wikibooks	website	none	none
babelium project	website	none	collaborative
leizea	website	SM2	automatic

3.6 Technologies

In this section some of the tools and technologies used are described. They can be classified into three categories. Two of them are general; *mercurial*, a distributed version control system and *sphinx*, a tool for writing and generating documenta-

tion. On the server, *python* is used as the programming framework and *django* as the web framework. The client uses the upcoming fifth revision of the *HTML* standard which is still under development but widely supported. The *dojo toolkit* is used to simplify the work. For the deployment *FreeBSD* (OS), *PostgreSQL* (DBMS) and *nginx* (HTTP server) are used.

3.6.1 Mercurial



Mercurial (or hg) is a distributed revision control system. Mercurial is mainly developed in *Python* and it is supported by different platforms like Windows, Mac OS X and most Unix-like operating systems. I used mercurial since the beginning of the project to control different versions and to have an updated backup.

Because of the distributed nature of Mercurial I could commit my atomic changes even when I did not have Internet access and thus revert the changes after a bad decision.

Further, I was able to have multiple copies of the whole repository, with the compromise to keep at least three of them (the ones from my pc, the server and *bitbucket*) as up-to-date as possible.

Mercurial is distributed under *GPL* license.

3.6.2 sphinx



Sphinx is a tool for developing documentation. It uses the format *Restructured text (rst)* for developing the content, and from this format documents in *LaTeX*, *html*, *man* and some other formats can be compiled. *Sphinx* adds some macros and tools to the human-readable *rst* format.

Sphinx is used by many important Python projects, such as *Bazaar*, *SQLAlchemy*, *MayaVi*, *Sage*, *SciPy*, *Django* and *Pylons*.

Sphinx allows creating new *builders* and *plugins*. For example, a special builder was developed, based on the *LaTeX* builder, in order to emit *PDF* files that follow the rules established by university for the master thesis.

3.6.3 Python



Python is a general-purpose, high-level scripting programming language whose design philosophy emphasizes code readability. Its syntax is said to be clear and expressive.

Development in *Python* is estimated to be 10 times faster than in *C*.

Its popularity for web development has been growing in the last years. *Google* and some other big companies have been using it and backing its development. *Guido van Rossum*, author of *python* is currently employed by *Google*.

I decided to use python because its simplicity and the wide range of libraries which make it ideal for prototyping and web development.

3.6.4 Django



Django is an open source web framework written in *Python*. As a web framework it enforces a specific development pattern, like dividing the different parts of the system in what it names reusable *apps* which follow the

model-view-controller architectural pattern.

Each app normally have all or some of the following files:

- **models.py**: The data models used by the application.
- **views.py**: The views provided by the application.
- **urls.py**: The URL patterns (in regular expressions) for the views provided.
- **tests.py**: Unit-tests for the application.

- **admin.py**: Definitions for the auto-generated admin BO (back-office).
- **forms.py**: Forms used in the application.

It also can contain templates and static files, or any other file needed by the application.

Django includes an *app* for generating a full admin back-office automatically, which enables the site administrator to administrate the data of the site, without any additional development effort.

The models in *django* define the persistent data, along with methods for them. The *ORM* (Object-Relational Mapping) takes care of creating the tables and the queries needed.

Here is a short example from the django website⁵ for creating a model and saving some data.

```
# in models.py
class Musician(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    instrument = models.CharField(max_length=100)

# ...
# in some view
jim = Musician('Jimi', 'Hendrix', 'guitar')
jim.save()
# add more data...
```

To retrieve some objects from the database:

```
guitarists = Musician.objects.filter(instrument='guitar')
```

It is licensed under *BSD*.

⁵ <https://docs.djangoproject.com/en/dev/topics/db/models/>

3.6.5 HTML5 and ECMAScript



HTML5 is the new HTML standard, which is, at the time of delivering this document, still under development. *ECMAScript* ([ECMA-262]) is popularly known as *javascript*. *HTML5* specifies new *APIs* that can be used with *ECMAScript*.

I have tried to use the new features in *HTML5*, like the new markup, the *canvas* tag for the character drawing question, the new *CSS3* styling properties and the audio *API* for the sounds in the study application.

I tried to use *HTML5* in order to avoid flash, or any non free technologies. As a result the learning system works in all modern systems, including those not supported by flash. The studying application works on android and iOS devices, although it has not been optimized for those.

Using *CSS3* allowed me to style the site more efficiently and getting faster loading times as background images for the different widgets are not needed this way.

A rectangular button with a red-to-black gradient background, a white border, and a subtle drop shadow. The word "SEARCH" is written in white, bold, sans-serif capital letters in the center.

For example, in *CSS2* to make a button like this needs a background image to do the gradient and the shadows.

With *CSS3* it can be styled like this.

```
button, .button {
    background: linear-gradient(center top , #C5000C 20%,
                                #451311 100%);
    border: 1px solid #451311;
    border-radius: 5px 5px 5px 5px;
    box-shadow: 0 1px 0 rgba(255, 255, 255, 0.3) inset,
                0 0 2px rgba(255, 255, 255, 0.3) inset,
                0 1px 2px rgba(0, 0, 0, 0.29);
    color: #FFFFFF;
    cursor: pointer;
```

```
padding: 5px 15px 3px;
text-align: center;
text-decoration: none !important;
text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.3);
text-transform: uppercase;
}
```

3.6.6 Dojo Toolkit



*Dojo*⁶ is an open source *ECMAScript* library for the development of *AJAX* web applications.

It is designed to be modular and from version 1.7 it uses *AMD* (*Asynchronous Module Definition*), which is a module definition format, later used by *requireJS*⁷ and widely adopted. From version 1.8 *AMD* will be the only method to require modules in *dojo*.

Dojo has a widget toolkit named *dijit*, which makes developing javascript browser applications similar to developing for the desktop,

Dojox is a collection of heterogeneous modules for *dojo* with different usages, like charting or phones.

I have used *dojo boilerplate*⁸ scaffolding the client. It provides a structure using *dojo*, *dijit* and *dojox*. Once you have some usable code you want to deploy, you can compile and minimize it using *Google Closure*⁹, for example.

Dojo is dual-licensed under *BSD* and *AFL*.

3.6.7 FreeBSD

⁶ <http://dojotoolkit.org/>

⁷ <http://requirejs.org>

⁸ <https://github.com/csnover/dojo-boilerplate/>

⁹ <https://developers.google.com/closure/>



FreeBSD¹⁰ is a free operating system based on UNIX. It is specially targeted for server, although it is also used as a desktop system.

It is licensed under BSD license. Because of the permissiveness of this license a lot of code has been reused by other operating systems, like Mac OS X.

Because of its long tradition (it's been actively developed since 1993) it is considered a stable OS for servers. The latest stable version 9.0 was released in January of 2012.

3.6.8 PostgreSQL



PostgreSQL

PostgreSQL is an open source ORDBMS (object-relational database management system).

The use of *PostgreSQL* is not strictly necessary in the project as *django* can handle different database management systems. While developing the system I used *sqlite3* for the database, but its usage is not recommended in production.

I decided for *postgresql* because of its stability and performance.

It is released under the PostgreSQL License, which is similar to MIT license.

¹⁰ <http://www.freebsd.org/>

Chapter 4

Design

In the following subsections some design decisions are introduced, together with an explanation.

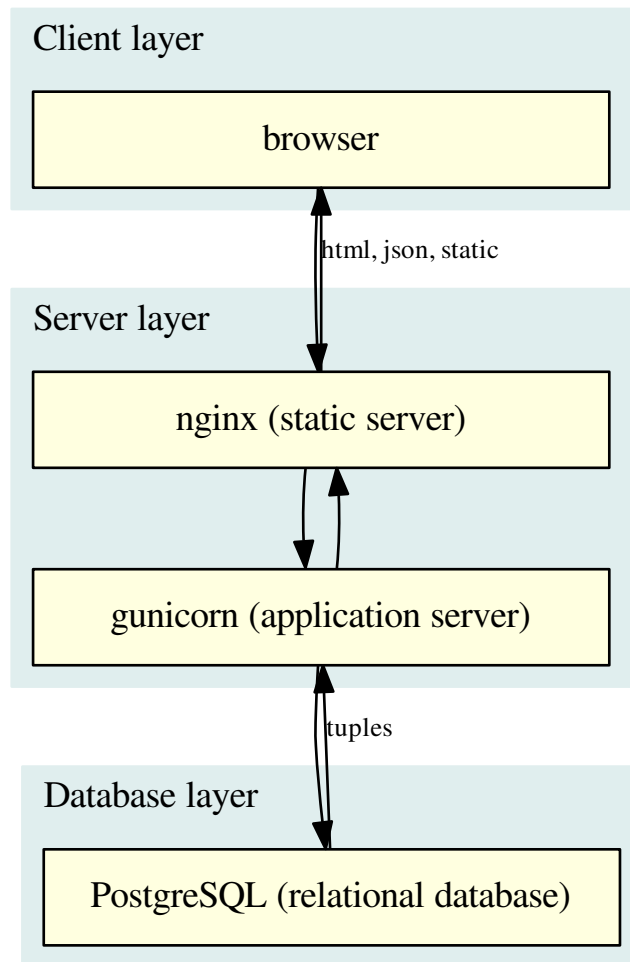
4.1 Architecture

Leizea is a web-based system. As most systems of this kind, it follows a three layer architecture. The following figure shows the architecture of the current current deployment, each box representing a separate process and the edges in the graph showing network traffic.

In the current version a lot of work is done by the server because the development was easier that way, but the strategy should be to evolve towards a thicker client and thinner server, with a well established API. The idea for future versions is to develop a full-fledged javascript client and leave only the RESTful API serving the content and synchronizing the items.

The communication with the server is done through a RESTful API. This API allows not only to develop clients for different platforms, but also to develop scripts that will automatically create lessons and items. In the next figure some possible use cases for the API can be seen.

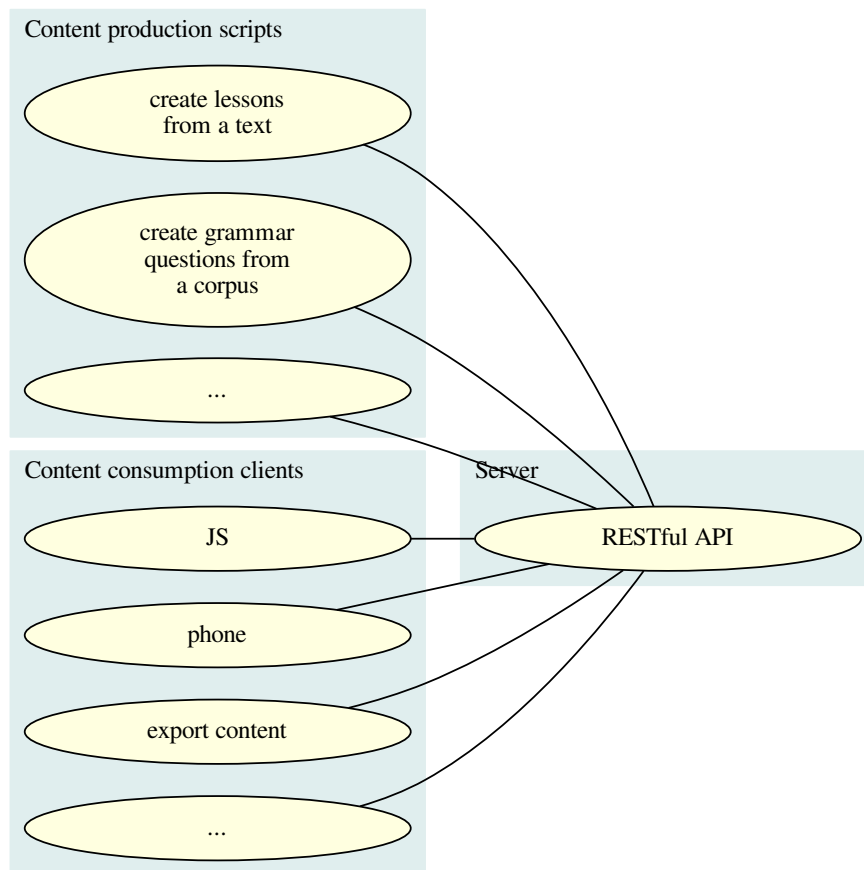
Figure 4.1: Architecture of the system in the current deployment.



Obviously the scripts for creating content don't need to be separated from the server and optimally they should be available for the users to create, for example, lessons from their own texts. But users could use the API to create lessons using their own scripts without the need to share their code if they don't want to do so.

A special effort has been done to make the system easier to translate.

Figure 4.2: Different use cases.



And one of the main tasks of the server is to schedule the items for next repetition according to what is explained in the *Scheduler* subsection.

One of the main tasks of the client is to display the questions and evaluate the answers grading them from zero to five as required by the scheduler (see *Scheduler*).

4.2 Items

The main and most important data in the proposed system are the *items*, that represent any concept that can be learned. The items are then collected in *lessons* and these in *courses*.

There are items of various types, each one dealing with different concept types, like word meanings or character readings, but all of them are handled equally by the *scheduler* because the studying dynamics should be the same for all kind of items.

The information contained in an item should be as simple and atomic as possible. Take a look at the table *how to create items*, which shows the different meanings of the word *arte* in basque. The different meanings for a word should appear as separate items, because a user might know one of the meanings of the word but not necessarily all of them.

Table 4.1: how to create items

Wrong		Right	
word	meaning	word	meaning
arte	holm oak; art; interval; up to	arte	holm oak
...	...	arte	art
...	...	arte	interval
...	...	arte	up to

Concepts should also be split even if they are related and taught always together in text books, like word meanings and readings. For example, a user could know that the word “你好” in Chinese language means “hello” but might not know that it is read “nihao”. In this case an item [你好 <-> hello] and an item [你好 <-> nihao] should be created.

Many spaced repetition systems take the *flash card approach*, that is, they define all the items the same way, as cards that have two sides: the question and the answer; or, in the case of words, word and translation. These cards can sometimes be reversed. For example, if we have the question “chien” with answer “dog”, we

can later reverse the card to ask “*dog*” with answer “*dog*”.

Some of these systems defining items as flashcards allow arbitrary content in both sides of the card like html markup, latex or images. On the one hand, this adds flexibility to the solution because they can add virtually any kind of questions and items, without any extra effort from the developers of the software. On the other hand, they sacrifice automatic checking of the answers because the software cannot know the nature of the question or the user cannot normally input markup. These systems, therefore, show the answer after the user has thought on the question and asks for the answer. Then the users are asked how well they knew the answer. This is error prone, because the user sometimes shows the answer before thinking carefully the answer or because they have to grade themselves in the range of 0-5, not just right/wrong, which is sometimes confusing.

For this reason, our solution takes the approach of defining different concepts differently. We define the class *Item* which holds the basic information for all the items, the one dealing with the scheduler and the organization of the items, and all the other *items* inherit from that class.

Because the system aims to be a general purpose learning system, the item system has been designed to be easily extended with new item types. In order to define an item type it has to be defined in the server and the client. The definition in the server only needs to specify the data that will be stored in the database. The definition in the client needs this same data-model definition and some questions for these kind of items. Ideally, the definition of the data required by the item should be done in JSON or some other format and used by the server and the client so it's not necessary to define twice.

Let's take a look at the definition of the *CharacterMeaning* item type, which can be used to study characters with meaning, like Chinese characters.

In the server it's defined like:

```
# simplified version, for explanation
```

```
@Item.register
```

```

class CharacterMeaning (Item) :
    name = u'character meaning'
    character = models.CharField(max_length=1)
    meaning = models.CharField(max_length=32)
    character_language = models.ForeignKey(Language)
    meaning_language = models.ForeignKey(Language)

    def item_definition(self) :
        return (self.character, self.character_language,
                self.meaning, self.meaning_language,)

```

We can see that the class inherits from the class *Item* directly and that we use a decorator¹ to register the item as an available type (i.e. this class is not abstract). The fields are defined like in other *django* model. The *item_definition* method is required for all item types. It should return a tuple with the fields that define the concept. This is done because some item types might contain other fields that are not part of the concept. The data returned by *item_definition* is used to construct a hash string that identifies the concept, so if the same item appears in two different lessons (for example, we could have the word *dog* in *Basic English words* lesson and in *Animals in English*) the two items will be treated as the same. This hash is called *iid*, which stands for *item identifier*.

An item type class in the client world inherits from the abstract *Item* class. This class contains the common methods needed by all the items and at the same time inherits from the class *_Model*, which implements the methods (*save* and *delete_*) that communicate with the *API*, through asynchronous *XMLHttpRequest* (XHR) for editing the data.

The class *WordItem* class, as an example, is defined as follows:

```

define([
    'dojo/_base/declare',
    'leizea/core/models/Item',

```

¹ In python a decorator is a special function that is used to transform a class or function, or simply to perform some operations on it. It is called adding a line starting with an at sign (@) plus the decorator we want to call.

```
'leizea/core/questions/multichoice',
'leizea/core/questions/inputquestion',
'leizea/language/views/WordItemInfo',
], function(declare, Item, Multichoice, InputQuestion, Info){
return declare(Item, {
  fields: ['translation', 'translation_language', 'word', \
          'word_language', 'model'],
  info: Info,

  getQuestions: function(){
    return [
      new Multichoice({
        word: this.word,
        answer: this.translation,
        distractors: this.translation.distractors
      }),
      new Multichoice({
        word: this.translation,
        answer: this.word,
        distractors: this.word.distractors
      }),
      new InputQuestion({
        question: this.translation,
        question_language: this.translation_language,
        answer: this.word,
        answer_language: this.word_language
      })
    ]
  },
});
```

The attribute *fields* is a list used to filter the data needed by the item when received from the API as JSON.

Every item type class should provide an *info* attribute. This attribute is the view that will be showed when the information about the item needs to be displayed, for example when the user fails a question the information about the item shows as a reminder, allowing the user to review it. The information shown is not limited to the information registered in the item. In the case of the *WordItem* objects, for example, it can include audio of the word, some example sentences containing the word loaded from a bilingual corpus or comments of other users on this item, like mnemonics.

Every item also needs a *getQuestions* method, returning a list of questions inheriting from the *_Question* class, which implements some common methods for questions. The questions should be ordered from easiest to most difficult, because the one to be asked is selected depending on the command of the item by the user. So in the case of *WordItem*, when the user does not know the item well the first question is asked, while if the user could know the user well the input question is asked.

Two stages are recognized while studying an item: the *recognition stage* and the *production stage*. As an example, a student of french might know the meaning of the word *chien* when reading it (recognition) but he might not remember it when asked for the word for *dog* in french (production). Item types should therefore, if possible, include both questions that test recognition and questions that test production.

4.2.1 Item types

The system could include the following item types. For each item type, I describe the usage of the item and the questions that could be done on it. If applicable I also give a list of the additional information that could be displayed when the user is studying it, besides the fields of the item and the comments of other users.

generic question [generic] A question consisting of the question and the answer, like “In what year did World War II end?”.

Questions

- Input the answer.

generic list [generic] Sometimes the user needs to learn the elements that compose a list, like “Which are the lobes of the cerebral cortex?”

Questions

- Select them from a list
- Input them

generic list [generic] The same as *generic list* but the answers have to be given in the correct order. Like “Navarrese monarchs of house Ximenez from 905 to 1234”.

image recognition [generic] An image and its name. Can be used, for example, to learn faces of people, or the name of a piece of art. The item should include also a question like, “Who is the one on the picture?”. It could also be used to teach vocabulary to children.

Questions

- Select the picture from a list of pictures.
- Select the answer from a list.
- Input the answer.

audio recognition [generic] An audio and its name. Can be used, for example, to learn names of songs or voices of people. The item should include also a question like, “What is the name of this song?”. It could also be used to teach the sounds of the animals to children.

Questions

- Select the picture from a list of pictures.
- Select the answer from a list.
- Input the answer.

word meaning [languages] A word and its translation to another language.

Questions

- Select the translation from a list.
- Select the word from a list.
- Input the word.

Additional information

- Audio pronouncing the word.
- Sentences from a bilingual corpus containing the word.

word reading [languages] A word and the way it is read.

Questions

- Hear the word and select it from a list.
- Select the reading from a list.
- Input the reading (or if technology allows it record the user pronouncing the word).

Additional information

- Audio pronouncing the word.
- Sentences from a bilingual corpus containing the word.

character meaning [languages] A character and its meaning. Maybe the meaning is in another language. Can be used for ideographs like in Chinese or Japanese, or for formal languages, like mathematical notation.

Questions

- Select the meaning from a list.
- Select the character from a list.
- Input the character on a canvas, given the meaning.

Additional information

- Words containing the character from a dictionary.
- If applicable, radicals forming the character.
- If applicable, animation of the writing (stroke order).

character reading [languages] A character and a way to read it.

Questions

- Hear the character and select it from a list.
- Select the reading from a list.
- Input the reading.

Additional information

- Audio pronouncing the character.
- Words containing the character using this reading.

grammar structure [languages] Defines a grammar structure, of the like of “There is X”. Sentences with gaps could be generated from a bilingual corpus.

Questions

- Fill a gap in a sentence, given different choices.
- Fill a gap in a sentence inputting it.

Additional information

- Explanation of the grammar.
- Sentences using this structure.

location question [geography] Is used to learn places, like country names or specific locations.

Questions

- A map is displayed and the user selects the name from a list.
- A map is displayed and the user inputs the name.

- The user selects the location on the map.

4.3 Client

The client can be split in two. In the one hand we have the interface for managing the contents in the site. It looks like a regular website but a different interface could be developed for devices with smaller screens, like mobile phones. On the other hand, we have the studying application which has been the main focus because of its singularity.

The web interface uses *960 grid system*² for the layout and displays the lessons classified by target language and a searcher. For each lesson the following data is shown always:

- name.
- description.
- the owner of the lesson.
- when was the lesson created.
- when was the lesson last updated.
- a long explanation on the lesson.
- users studying the lesson.
- comments on the lesson.

If the user is logged in, the following additional information is added:

- How many items were studied in the last two weeks.
- What percentage of the lesson has been completed.
- For each item how well it is known by the user. This percentage is calculated each time a user reviews an item, depending on the gotten score and the time

² <http://960.gs/>

elapsed since the last repetition.

- “*Study now*” button.

If the user is the owner of the lesson, he can edit its information too, on the same page by clicking on the icon depicting a pen by each of the modifiable field. He can also add explanations and items to the lesson.



Figure 4.3: A modifiable field.

The main part of the client is the studying application using *dojo* as a toolkit. The application has views which are basically *dijit* widgets. All the interface has been designed using *dijit*. The application receives the *id* of a lesson as parameter, the lesson that user wants to review. The lesson is then loaded by the client with its items and the progress on those items by the user.

The main task of the application is to test the user on a set of items, grading each item in the range of 0-5 as described in the *Scheduler* section and give those results to the scheduler. This grading should be done automatically, not manually as done in other spaced repetition systems.

The studying is split in study *sessions*. In each session a configurable maximum number of items is reviewed, being the default 10. In order to start a session the system needs to select the ten items that will be studied this time. The client distinguishes these four kind of items:

New items Items that were never studied yet.

Scheduled items These are the items that were already studied at least and were scheduled for today or some day before.

Items already studied today Items that were studied today but can still be reviewed for moving them from short-term memory to long-term.

Items not in schedule These items cannot be reviewed yet and will be dropped from the selection.

There is a configurable maximum number of new items that can be introduced in a session, which is five by default.

The client does the following for selecting the items to be reviewed:

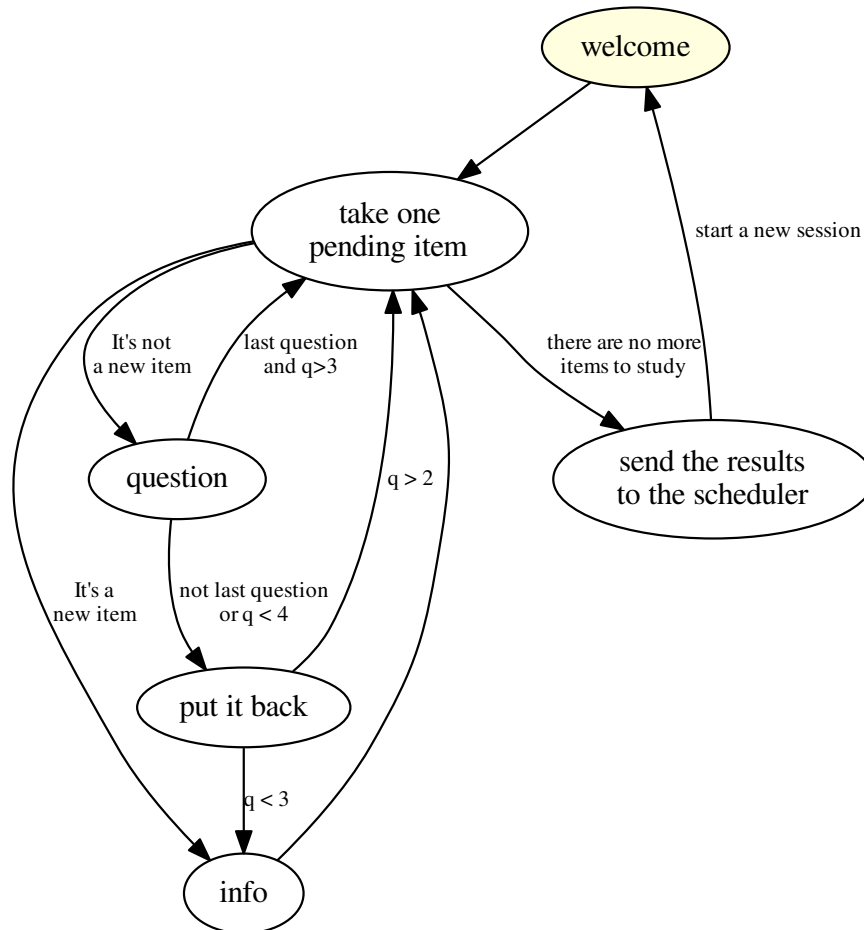
1. If there are scheduled items take up to the maximum, giving priority to the most urgent, that is, the ones that were scheduled to be reviewed first.
2. If items are still needed to reach the maximum, take new items up to the maximum, taking into account also the maximum for the new items.
3. If items are still needed to reach the maximum, take items that were already studied today for reviewing.

If after this the number of items for the session does not reach the maximum it does not matter, the session is done the same way.

The session is done in the following way:

1. Shuffle the *set of items to study*
2. An item is taken to review from the *set of items to study*
3. If the item is new the information of the item is displayed to the user, the question to be asked about the item is set to the easiest one and the item is put on a new *set of items to be studied*.
4. If it's been already studied but not on this session, a question is selected from the set of questions for that item depending on the score gotten by the user for this item the last time it was studied.
5. The selected question about the item is asked.
6. If the grading is below 3 (the answer was wrong) the item is put on the *set of items to be studied* next with the question to be asked next being an easier one (if any)
7. If the grading is over 2 (the answer was right)
 - (a) If there are more questions for this item, we set the next question to a more difficult one and the item is put in the *set of items to be studied*

Figure 4.4: Study session.



next.

- (b) If there are no more questions for this item and the quality of the answer was 3 the item is put in the *set of items to be studied next*.
8. If there are more items in the *set of items to study* go to (2)
 9. If there are no more items but there are items in the *set of items to be studied*

next, this set becomes the *set of items to study* and the *set of items to be studied next* is an empty set.

10. If there are no more items in either set save the session and start a new session.

At the end of a session for each item studied the quality of the response is sent to the scheduler. The worst score is sent to the server. That is, if the same item is asked three times in the same session and the qualities of the answers are 4, 3 and 5, the q sent to the server is 3.

Some questions, like the ones in which the user has to choose the right answer from a set of possible answers, need *distractors*. These distractors are compiled by the question using the other items in the lesson. Different methods can be used to compile these list of distractors, like using the levenshtein distance to find the distractors nearest to the right answer or keeping record of which answers are often confused. If necessary, the questions could load data from the server using the API.

The questions take homonyms and synonyms into account. So if the meaning for the basque word “arte” is asked and the user responds “art” which is correct, but the desired answer was “holm oak”, the systems should tell the user that the answer given is correct but that it is seeking another answer and ask again, without scoring negatively.

4.4 Server

The server has been implemented using *django* as a framework, for a faster development. I was specially interested in *django*’s *ORM* (Object-Relational Mapper) to avoid working with database tables and queries.

The server provides the traditional website part and an *API* working with JSON. This *API* is used by the studying application mainly, but it could also be used for other purposes, like automatically creating lessons and items from some software, for example. A mobile application is also planned, which uses this *API*.

Using *dango-rest-framework* made the development of the API easier because of its serializer and the web interface it provides for testing the API.

4.4.1 Apps

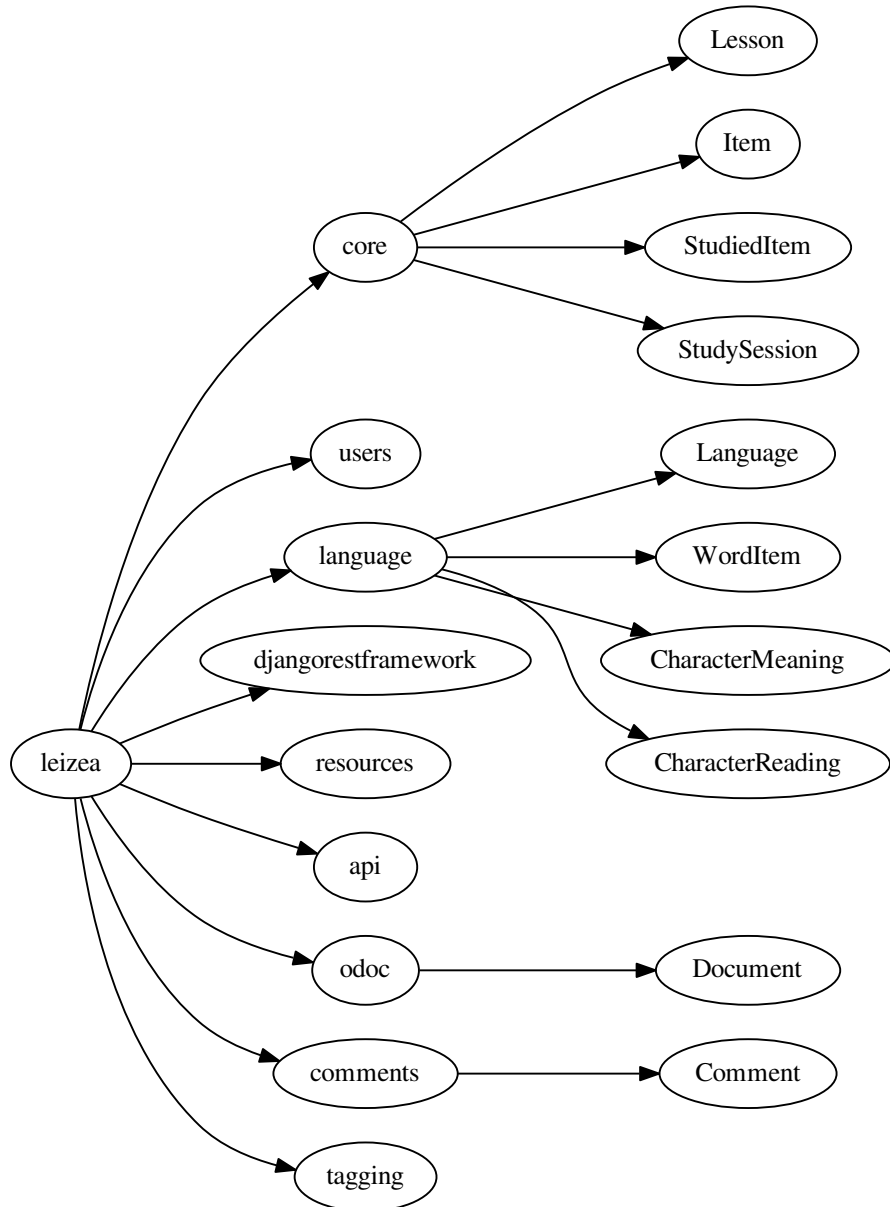
The following diagram shows roughly the apps used by the server and its data models, which are stored in the database. Some apps have been obviated for brevity.

Here are some explanations for the apps shown.

- The *core* app takes care of the scheduling. It has models for the lessons and the abstract *Item* type from which other models will inherit.
- The *language* app implements some items related to languages. It also has a model for the languages which store the language code according to [ISO639-3]. The list of codes can be downloaded from here³.
- The app *users* is related to the registering of users and the profiles.
- The *resources* app defines the abstract class *Resource* which all the resources in the API should extend.
- The *api* app holds the URL list for the API but does not implement any of the resources, which are implemented by the corresponding app.
- *odoc* is the app that implements the explanations for the lessons using *markdown* syntax. It could actually add written documents to any object, like users or items.
- The *comments* app implements all the commenting system.
- The *tagging* app allows users to tag arbitrary objects like lessons.

³ <http://www.sil.org/iso639-3/download.asp>

Figure 4.5: Apps used by the leizea server.

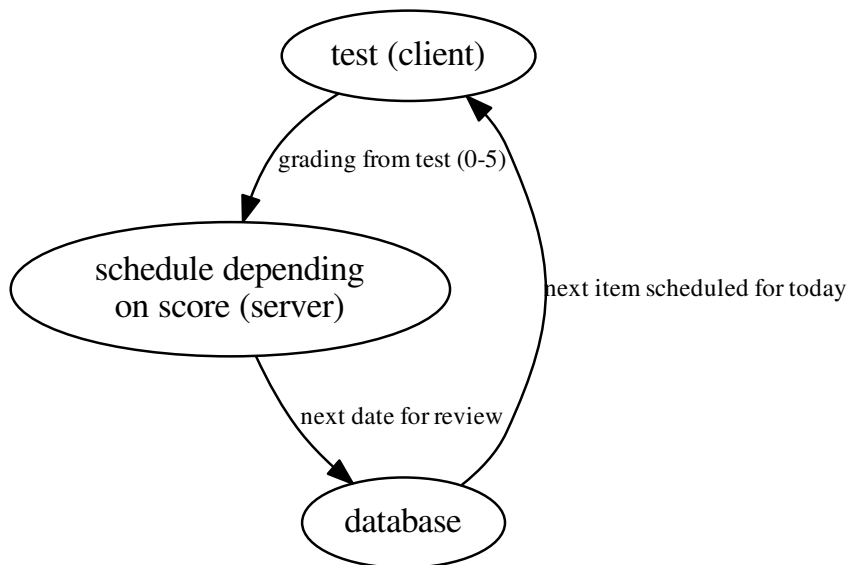


4.4.2 Scheduler

The scheduler is one of the most important parts of the project. It takes the score obtained in the test for each item and schedules the items accordingly.

The following diagram shows a very simplified diagram of the process.

Figure 4.6: Simplified behaviour of the scheduler.



The scheduler uses the algorithm SM2 described by P.A. Wozniak in [W90]. A very short explanation of the SM2 follows, although other explanations are also available on the web. [SM2]

For each item a user is studying the following data needs to be recorded:

$EF \in [1.3, 2.5]$ Easiness factor for an item. It defines how easy is this item for this user, being 2.5 the easiest and 1.3 the most difficult.

$n \in \mathbb{N}$ Number of times an item has been studied by this item.

In order to reschedule an item, the scheduler receives these data and $q \in [0, 5]$, which defines the quality of the response, like this:

- **5**: perfect response
- **4**: correct response after a hesitation
- **3**: correct response recalled with serious difficulty
- **2**: incorrect response; where the correct one seemed easy to recall
- **1**: incorrect response; the correct one remembered
- **0**: complete blackout

The number of days that need to pass before the item is repeated next is given by the function $I(n)$ (interval) defined next, if $q \geq 3$.

$$I(n) = \begin{cases} 1 & \text{if } n = 1 \\ 6 & \text{if } n = 2 \\ I(n-1) * EF & \text{if } n > 2 \end{cases}$$

If $q < 3$, the study is begun from the beginning without changing the EF , but resetting n and with an interval of 1.

The E-factor needs to be updated with each assessment, according to the following formula:

$$EF' = EF + (0.1 - (5 - q) * (0.08 + (5 - q) * 0.02))$$

If the new factor is lower than 1.3 it is set to 1.3 and if it goes over 2.5 to 2.5.

4.4.3 API

The API uses *djangoestframework* that helped with the serialization and visualization of the API. *djangoestframework* takes care of unmarshaling the request and marshaling the responses. Also, if the API is accessed through the browser it is user friendly.

/lessons [from the *core* app] The resource only displays the data of the lessons (i.e. it does not expand the items).

- *GET*: List of available lessons
- *POST*: Create a new lesson

/lessons/(?P<lesson_id>d+)

- *GET*: Information about a lesson
- *UPDATE*: Modify the information about the lesson
- *DELETE*: Delete lesson

/lessons/(?P<lesson_id>d+)/items

- *GET*: Items in a lesson
- *POST*: Add a new item

/lessons/(?P<lesson_id>d+)/items/(?P<item_id>d+)

- *GET*: Get information about specific item
- *DELETE*: Delete item from lesson

/lessons/(?P<lesson_id>d+)/users

- *GET*: Users studying a lesson
- *POST*: Start studying a lesson

/lessons/(?P<lesson_id>d+)/users/(?P<username>w+)

- *DELETE*: Stop studying a lesson

/users [from the *users* app]

- *GET*: Users registered in the system

/users/(?P<username>w+)

- *GET*: Information about a user
- *UPDATE*: Update information about a user

- *DELETE*: Delete a user

/users/(?P<username>w+)/lessons

- *GET*: Lessons owned by user
- *POST*: Create a new lesson owned by user

/users/(?P<username>w+)/studiedlessons

- *GET*: Lessons studied by user

/users/(?P<username>w+)/studiedlessons/(?P<lesson_id>d+)

- *GET*: Lesson studied by user

/users/(?P<username>w+)/studiedlessons/(?P<lesson_id>d+)/items

- *GET*: Items studied by user
- *UPDATE*: Change score for an item

/users/(?P<username>w+)/studiedlessons/(?P<lesson_id>d+)/sessions

- *POST*: Save a session

4.5 Importing data

The possibility of importing lessons and vocabulary lists is important too for generating freely available content. A lot of content is available on the Internet in various formats and for different systems. The users should be able to import their data from another system without the need to type it again.

Also, it should be possible to export the lessons from the proposed system to guarantee their freedom. After exporting the lessons it should be possible to import them.

The whole lesson is exported in a self-describing *json* file. If the lesson has an icon it could be included in the *json* file as a base64 string because the icons have low resolution.

Item sets from other systems like *Anki* and *Mnemosyne* should also be importable, although not all the items are importable because of the free format of their flash-cards that allow html markup, LaTeX and images.

Also, because many systems are using SM2 as scheduling algorithm, the study data could also be imported from those systems if the following data can be extracted for each item.

- Easiness factor
- Number of times it has been reviewed
- Next time scheduled

Anki stores the study data and the items in the same *sqlite3* database. *Mnemosyne* stores the study data separately.

Besides the lesson importing features the system should also include the choice to create lessons from other kind of material, like raw texts. For example, a user could download a short story from *Project Gutenberg*⁴, like Little Red Riding Hood, and create a lesson from that text. The system should do a lexycal analysis on the text, extract the words used and with a bilingual dictionary create the items. It could also extract some of the grammatical structures used.

4.6 Collaboration model

Any registered user can create lessons in the system. The lessons can only be edited by its owner.

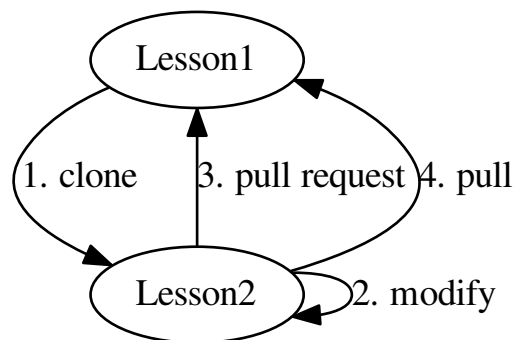
Users can comment on lessons and suggest changes to the lesson owner. Comments can be dropped on lessons, on items and also on users as public public messages.

But there is a better method for collaborating. This design proposes the *fork and pull* model as seen in some distributed version control systems. Following this

⁴ <http://www.gutenberg.org/>

model a user can clone a lesson from some other users and be the owner of the new cloned version. The user can then perform changes on the lesson and request the owner of the original lesson to pull those changes back. The owner of the original lesson can decide whether those changes are interesting for everybody.

Figure 4.7: Fork and pull collaborative model.



The good part of this method is that even if the changes are not accepted by the owner of the original lesson, the user can study his cloned lesson and because items are identified independently using their *iid* the system keeps track of the progress no matter from what lesson you are studying them.

4.7 Socialization

One step in the socialization of the site is the comment system and the private messages between users.

One way of further socializing the site is integrating it with other social networks like *twitter*⁵ and *Facebook*⁶ (or like its free counterparts *identi.ca*⁷ and *diaspora*⁸).

⁵ <http://twitter.com>

⁶ <http://facebook.com>

⁷ <http://identi.ca>

⁸ <http://joindiaspora.com>

One way of integrating them is letting the system write notifications automatically on the users account when some lesson is started or completed, this could encourage friends to study the same lesson.

The users could create their profiles using accounts from other sites implementing OAuth [RFC5849]. Also the avatars for the users can be changed from *gravatar*⁹.

Another step could be a friend system and clustering users with the same interest. For example, a user studying the same items as you could be suggested as a friend and be informed about the other lessons he is studying. Also, geographically close students could be suggested for real time practicing of a language.

⁹ <http://en.gravatar.com/>

Chapter 5

Implementation

5.1 Resulting product

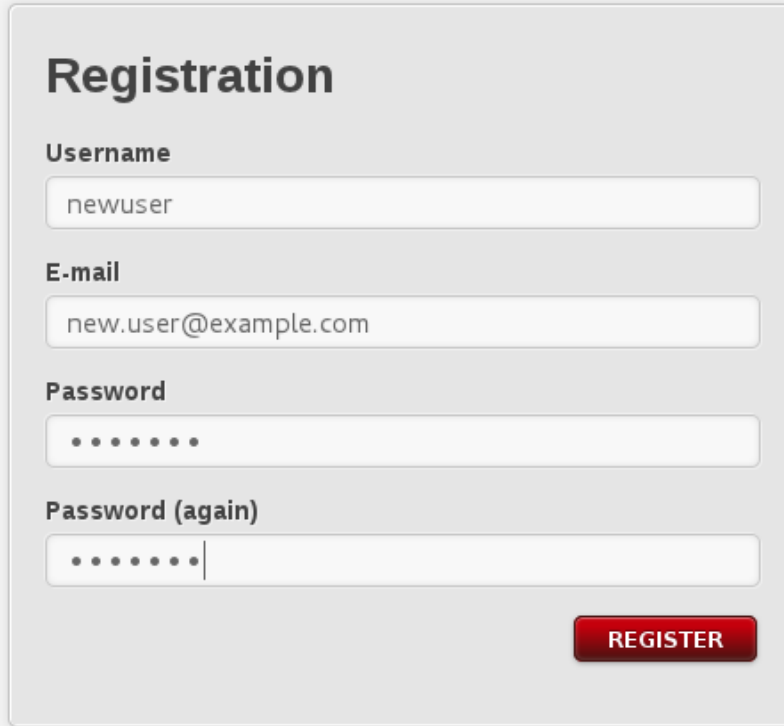
A simple working prototype of the proposed design has been implemented.

This section explains some basic use cases for the implemented application; like registering as a new user, creating a new lesson or studying.

5.1.1 Registering as a new user

A new user can be registered giving a username, email and password. The user will receive an email with an activation code to activate the account.

The application *django-registration* has been used to help with the registration process. Although it has not been implemented in the prototype, the possibility to create a profile using any OAuth [\[RFC5849\]](#) provider (Google, twitter, facebook, etc.) has been contemplated.



Registration

Username

E-mail

Password

Password (again)

REGISTER

Figure 5.1: Registering as a new user.

5.1.2 Searching a lesson

In the prototype the lessons can be browsed by language. This is so, because only language related features and items have been developed.

This classification is done through tags. Lessons can be tagged by users but some tags are automatically added by the system. For example when an English word is added to a lesson the tag *English* is added to the lesson automatically, so it will be listed when browsing by language.

There is also a search bar that will search lessons with matches to the searched terms in the name, description or in the tags. The search bar can be seen in all the lesson listings. Otherwise a search can be done like this.

http://leizea.net/lessons?search=SEARCH_TERM



Figure 5.2: Browsing lessons by language.



Figure 5.3: Search lesson by keywords.

5.1.3 Creating a new lesson

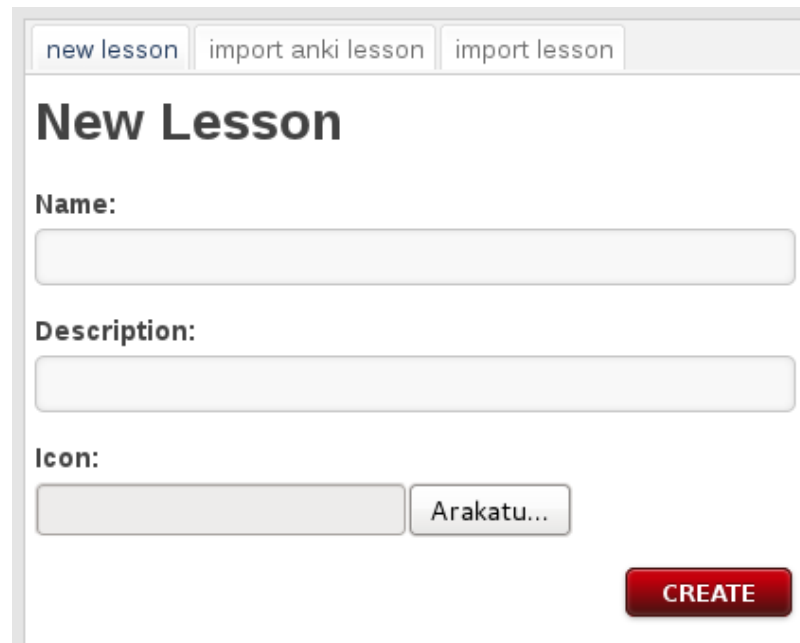
In all the lesson lists there is a *NEW LESSON* button. Lessons can be created from scratch or imported.

To create a lesson from scratch only the name and a description are mandatory. Optionally an image can be added to the lesson. This image, if added, will be shown when showing the lesson in the listing or when showing its details.

Lessons can be also imported. In the prototype the lessons can only be imported from *.json* files exported from the same system, or from *.anki* files. The anki import works only for vocabulary lessons and it does not import media, like images or audio because those are saved in separate files.

Anki files are *sqlite3* databases but the cards are HTML blobs. To import them all the html tags are stripped to get raw text, this works most of the time but a successful import is not guaranteed.

Lessons can be exported from the API.



The image shows a web form titled "New Lesson". At the top, there are three tabs: "new lesson", "import anki lesson", and "import lesson". The "new lesson" tab is selected. Below the tabs, the form has three main sections: "Name:" with a text input field, "Description:" with a larger text area, and "Icon:" with a text input field and a button labeled "Arakatu...". At the bottom right of the form is a red button labeled "CREATE".

Figure 5.4: Creating a new lesson

After creating a lesson the user is redirected to the detail page of that same lesson.

Adding items

Once a lesson is created items can be added.

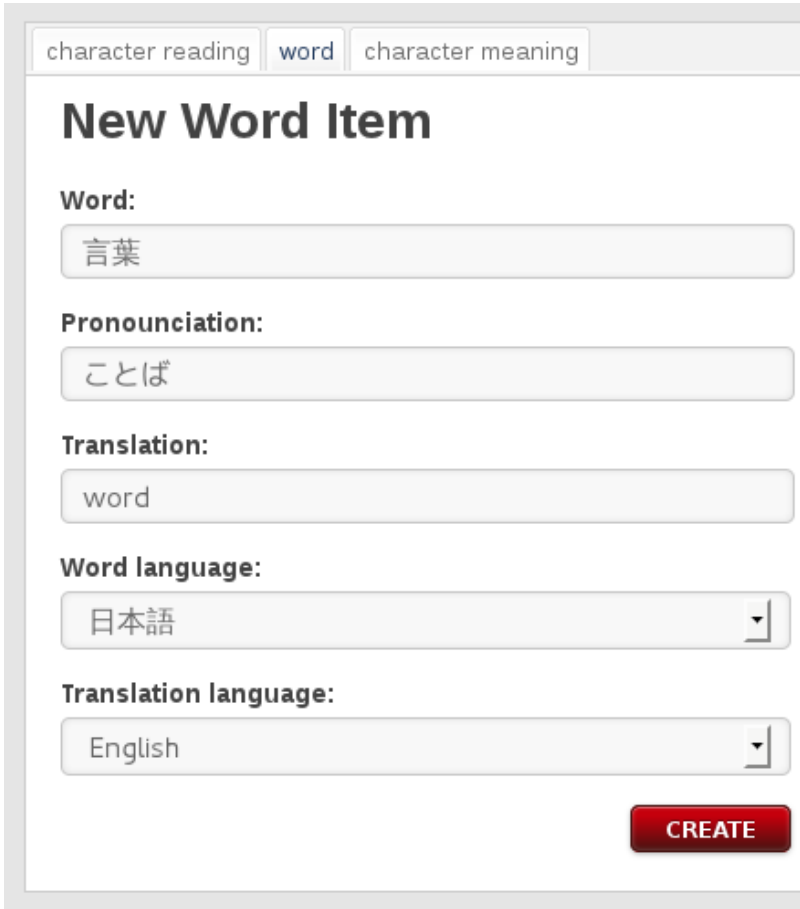
To create an item the item type has to be selected first. In the presented prototype the types *word*, *character reading* and *character meaning* have been implemented. Each of the item types requires different fields.

Adding an explanation

An unlimited number of explanations per lesson is allowed. Although most of the time it does not make sense to add more than one, if an explanation is too long it could be split in various shorter explanations.

Explanations are written using *Markdown*¹ as syntax. The prototype uses *page-*

¹ <http://daringfireball.net/projects/markdown/>



The screenshot shows a web form titled "New Word Item". At the top, there are three tabs: "character reading", "word" (which is selected), and "character meaning". Below the tabs, the form has the following fields:

- Word:** A text input field containing the Japanese characters "言葉".
- Pronunciation:** A text input field containing the Japanese characters "ことば".
- Translation:** A text input field containing the English word "word".
- Word language:** A dropdown menu with "日本語" selected.
- Translation language:** A dropdown menu with "English" selected.

At the bottom right of the form is a red button labeled "CREATE".

Figure 5.5: Adding a new item to a lesson.

*down*² as editor. *Markdown* was selected because it allows to import pages written in *HTML* while not sacrificing the easiness to write new content.

5.1.4 Commenting

Lessons, items and users can be commented. All of them use the same comment system which is based on the *django.contrib.comments*. It uses the same syntax and same editor as the explanations described above.

Commenting is only a small step on the socialization of the system.

² <http://code.google.com/p/pagedown/>

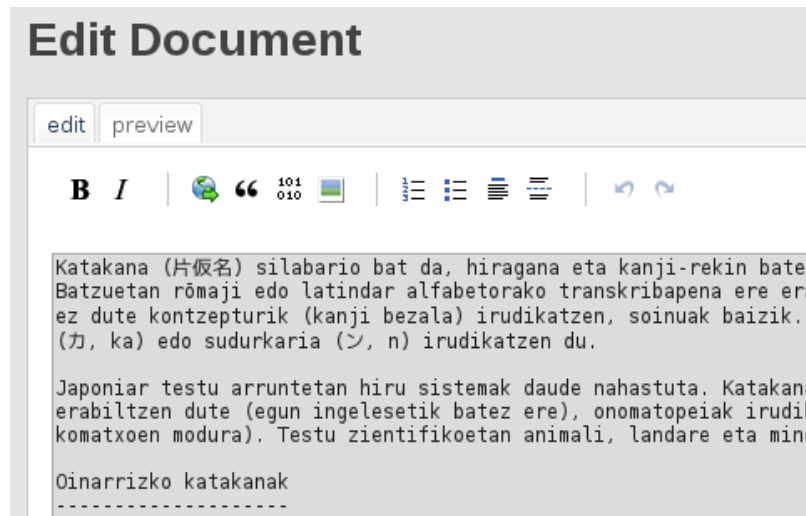


Figure 5.6: Adding an explanation to a lesson.

5.1.5 Studying

When logged in a *STUDY NOW* button will be visible in the detail page of each lesson. This link will load the study application.

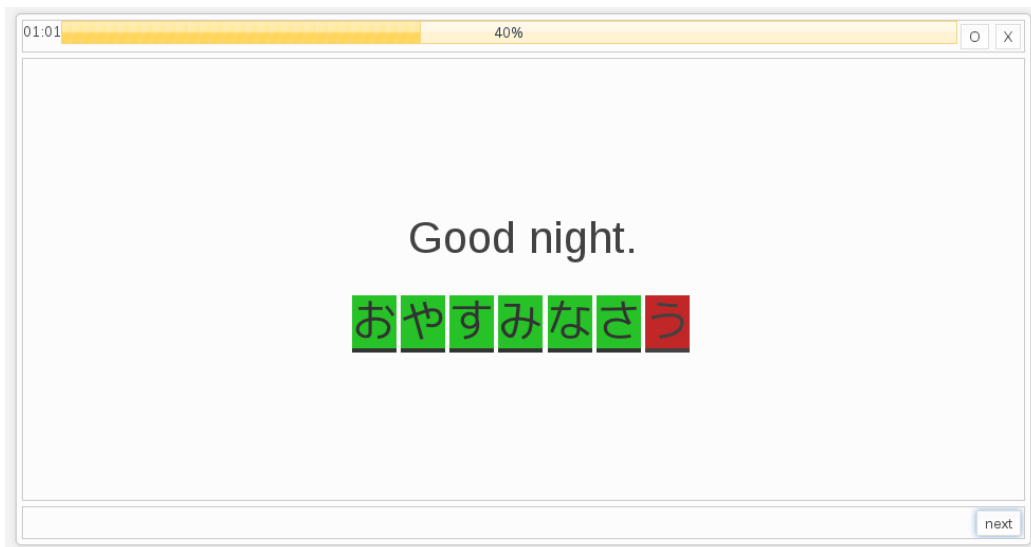


Figure 5.7: Adding an explanation to a lesson.

This application behaves like described in the design section. Only three of the

proposed item types have been implemented: word, character meaning and character reading.

In the prototype the character recognition has not been implemented and the user needs to grade themselves manually.

Each session is saved when all *items* have been discarded (all the questions have been answered correctly). Together with the scores obtained for the items the time taken is also sent for the statistics.

5.2 Deployment

The deployment has been done on an spare netbook I had at home with a cracked screen. The netbook is an *Acer Aspire One 110L*, which is not very powerful but enough for the experimentation. I bought the *leizea.net* domain for this purpose. It is running on a home network and therefore the speed is not so good and it varies a lot depending on other users being done. One idea is to migrate this to OpenShift³ servers, but this is beyond the scope of this project.

Table 5.1: Details on the server

Key	Value
Server	Acer Aspire One 100L
Domain	leizea.net
Operating System	FreeBSD 9.0
CPU	Intel Atom N270 Processor (1.6Ghz)
RAM	1GB DDR2
Disk type	SSD 8GB + SSD 16GB
Database	postgreSQL
HTTP server	nginx

I installed *FreeBSD 9.0*, which is the last version, in the laptop. The server holds two copies of the repository, one solely works as repository and the other one is

³ <https://openshift.redhat.com>

the working copy.

The HTTP server used is *nginx*⁴. It serves the static files of the project, such as javascript files, images and css files. It also proxies the application server which is *gunicorn*⁵ (*Green unicorn*). *Gunicorn* is used because its speed and its automatic worker process management.

*Supervisord*⁶ takes care of monitoring *gunicorn*, logging its stderr and restarting it if necessary.

In an attempt to isolate the application *virtualenvwrapper*⁷ is used, so the application has its own packages. This way the migration will be easier and we avoid the risk of breaking the application when updating a package accidentally.



Figure 5.8: Current production server at home. Acer Aspire One 110L.

⁴ <http://nginx.org>

⁵ <http://gunicorn.org>

⁶ <http://supervisord.org>

⁷ <http://www.doughellmann.com/projects/virtualenvwrapper/>

Chapter 6

Conclusion

An analysis of the different learning systems have been carried out in order to decide what are the features best suited for a modern *spaced repetition system*. As a summary, these features include:

- **Automatic scheduling of items** to reduce the time the student needs to schedule the study and thus increase the actual study time.
- The questions should be **automatically assessed** so the user will focus on the questions. Requiring the users to assess themselves disrupts the study flow.
- The system should be **social** to motivate the user into dilligent studying. The human-human interaction creates a positive stimulus in the student, same as the **gamification**.
- The study data should be **ubiquitous** as to be accessible from different places and devices and the system should be **platform independent** to guarantee this ubiquity.
- The system should facilitate and encourage **collaboration** between users for the common goal of free high-quality study materials.
- **Free and open source** to encourage **collaboration** and guarantee the continuation of the project.

Different choices for the scheduling algorithm have been analysed. The **SM2** algorithm is the most used mainly because of its simplicity as it does not need previous data about the user or the items to train the parameters. This simplicity comes at the cost of some loss in the performance of the student, but not enough to dismiss its choice. Another weak point of this algorithm is that it does not take into account short time memory, so the system can be sure that the users pay enough attention to their mistakes. This can be solved by testing the users also on items reviewed very recently but not taking the results of these tests into account for rescheduling.

Generally speaking all the goals have been fulfilled to some extent.

Although not all the features described in the design chapter have been implemented a fully functional prototype has been implemented and is available at <http://leizea.net>.

The tasks performed are not exactly the same as the ones described in section *Tasks* in chapter 2. On the one hand, the structure of this document has changed to have a more traditional and familiar format of *analysis-design-implementation* and as a consequence extra effort has been required in other areas of the project. On the other hand, all the tasks described have been carried out, but some unexpected have also been done and some have taken extra time.

The time taken by this project has been around 473 hours, but not all the tasks have been strictly measured so the real time could vary considerably. Some tasks have executed together which renders impossible to split the time necessary for each one.

This implementation has been tested to study some basic vocabulary of *Tagalog* and some *Basque* by two different users. The conclusions drawn from this testing are:

1. The necessity to work on the short time memory, as the users skipped too fast the most complicated items without paying enough attention to them. This was added to the original scheduler on a later stage of the project and showed to improve performance of the users.

2. More statistics on the progress of the users could encourages them to study more continuously as it helps them get the feeling that they're making advancements.
3. The viability of the project. Although a full implementation would require more time than the anticipated for this project.

Appendix A

Licenses

A.1 Leizea license

All the code in *Leizea* is released under the Simplified BSD license, while the used libraries and software retain the licenses established by their respective owners.

The BSD license is flexible. Simply explained everybody has the right to distribute, sell or change the code as long as they mention the source.

Copyright 2012, Ander Martinez.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.2 Document license

This document is released under the FreeBSD Documentation License (FBDL).

Copyright 2012, Ander Martinez.

All rights reserved.

Redistribution and use in source (Restructured Text) and ‘compiled’ forms (HTML, PDF, PostScript, DVI and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (Restructured Text) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.
2. Redistributions in compiled form (converted to PDF, PostScript,

RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY THE COPYRIGHT HOLDER “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.3 Content license

The content in **leizea.net** is published under the CC-BY (Creative Commons) license, reproduced below.

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE (“CCPL” OR “LICENSE”). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE,

YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- (a) **“Adaptation”** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image (“synching”) will be considered an Adaptation for the purpose of this License.
- (b) **“Collection”** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

- (c) **“Distribute”** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.
- (d) **“Licensor”** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
- (e) **“Original Author”** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
- (a) **“Work”** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science;

a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

- (b) **“You”** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
 - (c) **“Publicly Perform”** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
 - (d) **“Reproduce”** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.
4. **Fair Dealing Rights.** Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.
5. **License Grant.** Subject to the terms and conditions of this Li-

cense, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- (a) to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
- (b) to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked “The original work was translated from English to Spanish,” or a modification could indicate “The original work has been modified.”;
- (c) to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
- (d) to Distribute and Publicly Perform Adaptations.
- (e) For the avoidance of doubt:
 - i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
 - ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
 - iii. **Voluntary License Schemes.** The Licensor waives the

right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

6. **Restrictions.** The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:
 - (a) You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Col-

lection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(b), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(b), as requested.

- (b) If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution (“Attribution Parties”) in Licensor’s copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., “French translation of the Work by Original Author,” or “Screenplay based on original Work by Original Author”). The credit required by this Section 4 (b) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors.

For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

- (c) Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

7. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITH-

OUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

8. Limitation on Liability.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

9. Termination

- (a) This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- (b) Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to

withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

10. Miscellaneous

- (a) Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- (b) Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
- (c) If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- (d) No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- (e) This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and

You.

- (f) The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Bibliography

- [T06] Thalheimer, W. (2006, March). Spacing Learning Over Time. Retrieved March 22, 2012, from <http://www.work-learning.com/catalog.html>.
- [ECMA-262] Current ECMAScript standard in browsers. <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- [ISO639-3] ISO 639-3 *Codes for the representation of names of languages – Part 3: Alpha-3 code for comprehensive coverage of languages*
http://www.iso.org/iso/catalogue_detail?csnumber=39534
- [W90] Piotr A.Wozniak (1990). Optimization of learning. Available at <http://www.supermemo.com/english/ol.htm>
- [SM2] Description of the SM2 algorithm.
<http://www.supermemo.com/english/ol/sm2.htm>
- [RFC5849] E. Hammer-Lahav, Ed. “The OAuth 1.0 Protocol” IETF, RFC 5849, April 2010 [Online] <http://tools.ietf.org/html/rfc5849>