VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POPIS JAPONSKÝCH ZNAKŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB HOLÁŇ
AUTHOR

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POPIS JAPONSKÝCH ZNAKŮ

JAPANESE CHARACTER DESCRIPTION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE JAKUB HOLÁŇ
AUTHOR

VEDOUCÍ PRÁCE ING. PETR HORÁČEK
SUPERVISOR

BRNO 2008

Abstrakt

Japonský jazyk má z historických důvodů velmi komplikovanou ortografickou stránku skládající se z více různě propojených abeced, kde jednotlivé znaky mohou být jak logogramy (znaky reprezentující celá slova) tak fonogramy (znaky reprezentující jednotlivé zvuky či slabiky jazyka). Při vyhledávání nebo například sepisování slovníku japonštiny se musí použít různé netriviální metody řazení těchto znaků do skupin, například podle počtu tahů štětcem nebo řazení podle radikálů (základních stavebních kamenů znaku). Cílem této práce je najít ideální strukturu databáze popisující japonské znaky a vztahy mezi nimi. Zároveň byla vyvinuta ukázková výuková a pomocná aplikace pro studenty japonštiny. Aplikace je vyvinuta pro internetové použití a je postavena na nejmodernějších webových technologiích, jako jsou Google Web Toolkit a Java servlety.

Abstract

The Japanese language has from historical reasons a very complicated orthographical system based on more interconnected alphabets where the characters can be logograms (characters representing whole words) as well as phonograms (characters representing single sounds or syllables of a language). When searching or compiling a japanese dictionary various nontrivial methods of sorting these characters have to be used, e.g. sorting them by number of strokes or by radicals (basic building elements of a character). The aim of this thesis is to find an ideal database structure to accurately describe japanese characters and relationships between them. For good measure, a demonstration application has been developed, which can be used as a learning and revision tool for japanese students. The application is intended for internet use and it has been put together using the most up to date web technologies, such as Google Web Toolkit and Java servlets.

Klíčová slova

Japonština, kanji, jazykověda, výuková aplikace, Google Web Toolkit, MVC, Java servlets.

Keywords

Japanese language, kanji, linguistics, learning tool, Google Web Toolkit, MVC, Java servlets.

Citace

Holáň Jakub: Popis japonských znaků, bakalářská práce, Brno, FIT VUT v Brně, 2010

Popis japonských znaků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Horáčka
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jakub Holáň
19.5.2010

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Petrovi Horáčkovi za jeho čas, cenné rady a informace.
Také bych rád poděkoval svojí rodině za podporu a gramatickou kontrolu textu.

Obsah

1 Úvod

Cílem této práce je seznámit se se skladbou japonštiny a to hlavně její ortografickou stránkou a určit ideální metodu reprezentace tohoto jazyka v databázovém modelu. Dále je potřeba vytvořit ukázkovou aplikaci, která dobře předvede schopnosti takového modelu. Systém by měl splňovat základní nároky kvalitního softwaru, funkčnost, přehlednost a příjemné uživatelské prostředí. K tomu jsem se rozhodl přidat dodatečné nároky jako multiplatformnost, snadná instalace a veřejný přístup.

První část dokumentu se soustředí na seznámení s japonštinou jako takovou a její analýza, což má pro práci esenciální význam. Japonština není jednoduchý jazyk, je zatížený mnoha historickými zvraty, měnila se výslovnost souhlásek i samohlásek, přicházely nové a nové znakové sady, probíhaly standardizace, které byly později opět přecházeny (např. příchodem evropských abeced nebo arabských číslic), proto poslouží lepší seznámení. Následuje návrh aplikace, který se zabývá jak návrhem samé databáze, tak i základním rozvržením ukázkové aplikace a jejích součástí. Nakonec se dozvíme něco o implementaci. Ta stojí na moderních technologiích, které spatřily světlo světa jen pár let zpátky. Nestandardní mix, který jsem použil, má dobrý důvod a dobře posloužil.

2 Úvod do japonštiny

Většina jazyků náleží do jedné z více rodin, jako například jazyky indo-evropské, do kterých patří například angličtina nebo i naše čeština. Proto nám nedělá významný problém naučit se většinu evropských řečí. Máme podobnou gramatiku, některá podobná slovíčka, stejný systém psaní a hlavně společnou historii. Japonština není členem indo-evropské rodiny. Ve skutečnosti je japonština doslova jediným členem japonské rodiny. Korejština, která je nejbližším příbuzným japonštiny, patří do jiné rodiny. Většina evropanů si neuvědomuje rozlišnost asijských řečí. Přestože korejská gramatika je podobná japonštině, má velmi odlišný slovník. Čínština ovlivňovala japonštinu po dlouhá léta, ale její gramatika nemá žádný vztah k japonštině. [1][2][3]

Studium neevropských jazyků je pro evropana dobrý způsob jak porozumět všeobecným principům lidské komunikace a pochopit specifika evropských jazyků. Můžete si myslet, že prohození pořadí podmětu a přísudku v otázkách není nic zvláštního, ale tento gramatický prvek se téměř neobjevuje mimo Evropu. A z tisíců světových řečí je angličtina jediná na světě, která používá pomocné sloveso pro vyjádření otázky (např. Did you do this?). Typická věta v japonštině by vypadala takto:

Tomato wa yasai da. (Rajče je zelenina)

Skladba této věty je tedy podmět – oddělovač podmětové a přísudkové části – předmět – přísudek. Otázka se z této věty vytvoří přidáním tázací částice. To je mimochodem případ většiny světových neevropských jazyků:

Tomato wa yasai da ka? (Je rajče zelenina?)

Japonština tedy používá stavbu věty SOV (subject – object – verb), nikoli SVO (subject – verb – object) jako většina evropských jazyků. [4]

2.1 Japonské znaky

Japonština má tři sady znaků – hiragana, katakana a kanji. Japonština před dvěma tisíci lety neměla psanou verzi. Po kontaktu s Čínou Japonci dovezli znaky Han, které se v japonštině nazývají kanji. Kanji jsou ideogramy, to znamená, že symbolizují význam a nemají nic společného s výslovností. Velmi obvyklým znakem kanji je například:

人

Japonská výslovnost tohoto kanji je „hito“, což vůbec není podobné čínské výslovnosti „rén“, nicméně v obou jazycích má toto kanji význam *člověk*. Většina kanji ovšem vypadá mnohem komplikovaněji, kupříkladu tyto:

国 - země

劑 - lék

蠹 – můra

Ideogramy jsou užitečné v tom, že dovolují mnohem rychlejší čtení textů. Celá věta se může vejít do tří či čtyř znaků. Lingvisté také dokázali, že ideogramy a jejich význam jsou v mozku přímo propojené, kdežto fonogramy jsou spojeny pouze s jejich zvukovým vyjádřením. Toto propojení dále zrychluje čtení. Ideogramy se také snadno mohou přenášet mezi jazyky, aniž by si jejich mluvčí museli vzájemně rozumět. jak jsme si ukázali na příkladu se znakem *člověk*. To mělo význam i v historii při obchodu mezi kmeny.

Je ovšem také užitečné mít fonogramy, které popisují pouze zvuky. Před více než tisíci lety si Japonci vybrali kolem 50 znaků kanji a zjednodušili je, aby vytvořili nové sady znaků dnes zvané hiragana a katakana. Dohromady se tyto dvě abecedy nazývají kana. Kana značí narozdíl od kanji pouze zvukové vyjádření. Vzhledově jsou také mnohem jednodušší než kanji. Typický příklad kana:

ひと

Tyto dvě kany reprezentují výslovnosti „hi“ a „to“, dohromady „hito“ (*člověk*). Když chtějí Japonci vyjářit výslovnost nějakého znaku kanji, nejčastěji k tomu použijí právě hiraganu.

Tyto dvě abecedy, hiragana a katakana, jsou víceméně totožné, ale mají jiné grafické znázornění, podobně jako čeština má malá a velká písmena. Hiragana se používá pro japonská slova, zatímco katakana se používá pro slova dovezená z angličtiny, čínštiny apod. Katakana mívá obecně hranatější tvary, kdežto hiragana se považuje za kulatější a ozdobnější. Pro srovnání „hito“ napsané v katakaně:

ヒト

Japonština se píše vertikálně zvrchu dolů a řádky se přidávají zprava doleva. Ale je možno ji psát i evropsky, tzn. zleva doprava a řádky zvrchu dolů. Noviny a romány se téměř vždy píšou vertikálně a vědecké publikace téměř vždy horizontálně. Počítače používají rovněž horizontální způsob psaní, protože jsou málokdy schopné zobrazit znaky vertikálně.

V softwaru bakalářské práce používáme jak kanji (samotné jádro bakalářské práce, stejně tak jádro databázové struktury), tak obě kany, které jsou používány pro zobrazení čtení znaků kanji. Počítá se s tím, že uživatel našeho systému zná obě obecedy kana, tedy dohromady asi 100 znaků, které se dají naučit za tři dny. Náš systém je vhodný hlavně na zlepšení znalostí kanji a kanji složenin. Kanji znaků samotných je na tisíce. Znalost aspoň 2000 kanji je v Japonsku minimum gramotnosti – úroveň, na jaké opouštějí děti základní školu. Kanji složenin je nespočet.

2.2 Romanizace

Psaní japonštiny latinkou se nazývá romanizace. Existuje více mezinárodních způsobů romanizace, ale platí v nich, že čtení samohlásek je založené na španělské výslovnosti (tedy „a e i o u“ se čtou skutečně jako „a e i o u“ – podobně jako v češtině) a čtení souhlásek je založené na angličtině (tedy „j“ se čte jako „dž“ apod.). [5]

Existují dva hlavní systémy romanizace. Jeden se jmenuje Kunrei a druhý Hepburn. Kunrei je ustanoven jako mezinárodní standard (ISO-3602), protože velmi dobře reflektuje japonský zvukový systém, ale Hepburn je jednoznačně nejpoužívanějším a nejrozšířenějším systémem přepisu. Pro rodilého Japonce může být matoucí, protože je velmi nesystémový, ale pro evropana rychle pochopitelný. Kana se totiž skládá ze slabikových uskupení po 5 slabikách. Například hiragana pro „S“ – „さしすせそ“. Kunrei přepis by byl „sa si su se so“. Jenže rodilý Japonec vysloví „si“ tak, že to evropskému uchu zní jako „ši“. To právě zohledňuje Hepburnův přepis, který zmíněnou řadu přepíše na „sa shi su se so“. Podobná řada pro „T“ je „たちつてと“. Kunrei přepis je „ta ti tu te to“, Hepburnův přepis „ta chi tsu te to“. Rodilému mluvčímu takové změny v řadě mohou přijít nesystémové. Jenže kdyby se místo „tsunami“ psalo „tunami“, nevyjadřovalo by to dobře japonskou výslovnost tohoto slova.

Pro přepis japonštiny do češtiny se ve valné většině případů používá Hepburn podobně jako ve zbytku světa, nicméně některé souhlásky se přepisují na českou výslovnost (j => dž, y => j, ch => č apod.). V softwarové aplikaci stejně tak jako v této technické práci nicméně používám originální Hepburnův přepis, který je používán na celém světě, jak v populárních publikacích, tak ve vědeckých pracech a to nejen v angličtině, takže člověk studující japonštinu s ním vždy přijde do kontaktu.

V softwaru se nicméně romanizace v podstatě nepoužívá, protože má jít o výukovou pomůcku a tudíž je kana používána přímo, aby mohla být takto procvičována. Rychlé čtení kany je v japonštině esenciální. Pokud je potřeba provést překlad z latinky do kany nebo obráceně, slouží k tomu modul *editor kany*, který umí pracovat s romanizací.

2.3 Zvuková skladba jazyka

Pro lepší pochopení znakové skladby japonštiny (což je jádrem této práce) je vhodné vědět aspoň základy její zvukové stavby. Jednoduše vzato, souhlásky musí být v japonštině vždy následovány samohláskami. Výjimku tvoří polosamohlásky, které mohou být vloženy mezi souhlásku a samohlásku a speciální hláska „n“. Toto jednoduché pravidlo dělá výslovnost každé slabiky velmi jednoduchou. Každá slabika má tedy jeden z následujících tvarů:

- Samohláska (např. „a“, „i“ atd.)
- Souhláska + samohláska (např. „ka“, „ki“ atd.)
- Souhláska + polosamohláska + samohláska (např. „kya“, „kyi atd.)
- Hláska „n“

Věta ze začátku kapitoly 2 by se teda dala rozložit na slabiky takto:

To-ma-to wa ya-sa-i da ka? (Je rajče zelenina?)

Nejdůležitější pravidlo japonské vyslovnosti je, že je metronomická – tzn. každá slabika má stejnou časovou délku a to metronomicky přesně. Neexistují žádné dlouhé a krátké souhlásky (existují ale zdvojené, ty pak mají přesně dvounásobnou délku) a důrazy se týkají pouze výšky tónu, neprodlužují slabiku stejně jako je tomu například v angličtině. Co je ještě důležitější, každá takto definovaná základní časová jednotka jazyka (nazývá se *mora*) odpovídá jedné kaně. Tudíž přepis předcházející věty do hiragany by byl:

とまと は やさい だ か? *(Je rajče zelenina?)*

Tento slabikový systém japonské fonetické abecedy velmi dobře funguje pro japonštinu, +protože je, jak jsme si řekli, metronomická a používá jednoduché slabiky. Problém přichází až v momentě, kdy je potřeba importovat nějaké slovo z cizí řeči. Například anglické slovo *pro* (*profesionál)* se do japonštiny přepíše jako *puro* (プロ). Stejně tak v japonštině chybí například reprezentace pro hlásku „v“ nebo „l“. Tudíž takové slovíčko jako například *Vancouver*, dopadne v japonském přepisu jako *bankuubaa*. Za písmeno „v“ se většinou nahradí „b“, které je nejblíž (podobně jako ve španělštině).

2.4 Úvod do kanji

Znaky kanji vznikly, jak už bylo řečeno, postupným přejmutím a úpravou čínských znaků Han. Samotné znaky Han vznikly poměrně komplikovanou cestou z pravěkých piktogramů, které označovaly místa, věci apod. Postupně přibyly i abstraktní koncepty a nakonec gramatika. Počet znaků je nejistý. Některé slovníky uvádějí až 50000, jiné dokonce 100000. K běžnému přežití ale stačí 2000-3000 znaků. V Číně ovšem mnohem víc. Japonština dokáže příval moderních přejatých slov pojmout v přepisech do katakany, není potřeba vytvářet další znaky. [6] [7]

2.5 Čtení znaků kanji a složeniny

Kvůli komplikované historii přijímání kanji do japonštiny má každé kanji více různých čtení a dokonce i významů. To dělá digitální práci s kanji velmi nepříjemnou. Rozhodnutí, jak které kanji přečíst, závisí na kontextu a hlavně na použití ve složenině. Některé kanji mají i deset různých čtení. Čtení se kategorizují na dvě hlavní skupiny – **on’yomi** (doslova *zvukové čtení*) a **kun’yomi** (doslova *významové čtení*).

2.5.1 Čtení On

Čtení On je moderní pozůstatek japonské přibližné aproximace původní čínské výslovnosti znaku. Některé kanji se v Číně objevily na více různých místech zaráz a tak mají více čtení, mnohdy i více různých vyznamů. Kanji, které se objevily v Japonsku, většinou nemají žádné čtení On. Čtení On se primárně objevují ve složeninách více kanji. Pokud složíte více kanji za sebe, změní se jejich význam a mohou znamenat něco úplně jiného než původní samostatné kanji. Takováto složenina pak na sebe bere On čtení jednotlivých kanji. Jelikož slova v čínském jazyce jsou většinou jednoslabičná a čtení On vychází z čínštiny, délka takovýchto složenin stále zůstává únosná.

2.5.2 Čtení Kun

Čtení Kun je čtení založené na výslovnosti původního japonského slova, které bylo co nejbližším synonymem čínského slova, jež patřilo k danému čínskému znaku. Je to tedy čtení vycházející z japonštiny. Stejně jako se čtením On, pro každý znak může existovat více čtení Kun nebo žádné. Kun čtení bývají delší než On, japonština nemá tolik krátkých slov jako čínština. Také se někdy do Kun čtení započítává i takzvaná *okurigana*, praktičnost takového přístupu je ale diskutabilní, protože *okurigana* není nic jiného než skloňování přídavných jmen a sloves. Nicméně v našem softwaru je pro zjednodušení *okurigana* součástí čtení.

2.6 Typování a řazení kanji

Jelikož jsou kanji poměrně amorfní prvky písemného vyjádření jazyka, vznikl problém jak vlastně sdružovat kanji do skupin, případně je řadit. Mnoho učenců se v historii snažilo vymyslet ideální způsob, ale každý má své úskalí a specifika. První, který se nabízí, je seskupovat kanji podle počtů tahů štětcem nutných pro jejich znázornění. Další je řazení podle radikálů. V softwaru používáme oba.

2.6.1 Tahy štětcem

Historicky se kanji nejprve ryly do hliněných tabulek, po vynálezu papíru (který byl vynalezen poprvé v Číně) se ale začaly kreslit štětcem. To přineslo potřebu standardizovat tvary jednotlivých kanji a taky zajistit jistý standardizovaný způsob, jak kanji kreslit – v jakém pořadí pokládat jednotlivé linie a také obecnější pravidla, jak vlastně kreslit kanji. Dnes máme přesně nadefinované tahy štětcem a jejich typy, od kratkých, po dlouhé, a aby toho nebylo málo, máme i složené tahy, které se považují za jeden tah, ale ve skutečnosti se skládají z více tahů. Stejně tak máme pro každé kanji přesné pořadí tahů, jak je po sobě kreslit. V některých slovnících se tedy používá řazení kanji podle počtu tahů a u každého kanji je také uvedeno několikero obrázku kanji v jednotlivých fázích kreslení, aby bylo poznat, jakým způsobem kanji správně zobrazit. Obor umění zabývající se ozdobným malováním kanji se nazývá *kaligrafie*. [8]

2.6.2 Radikály

Mnohem používanější metodou seskupování kanji jsou ale takzvané radikály. Radikál je grafický prvek ve znaku kanji, který se dá najít ve více různých znacích. Každý znak má aspoň jeden radikál, případně víc a i radikál sám o sobě může být kanji. Radikál je součást kanji, která dává klíč k původu, skupině nebo významu kanji. Známe 214 radikálů. Jedním z nich je například výše zmíněný znak pro slovo *člověk* - 人. Jako radikál ho najdeme vměstnaný na levé straně mnoha znaků, např:

僣儉倹倆倅俸

Nemusí však být umístěn pouze na levé straně, může se objevit i uprostřed, vpravo, nahoře, dvakrát uprostřed, vpravo dole, či kompletně ztracený v komplikovaném znaku:

囚以内卒亥劾

Existuje 14 typů radikálů podle jejích umístění a navíc kanji se často skládají z jiných kanji, aby vznikl zcela nový znak. To ale nemá nic společného se složeninami! Jen to komplikuje klasifikaci umístění. Příkladem složeného kanji je například poslední uvedené kanji (劾), které se skládá z kanji uvedeného před ním (亥) a znaku 力. [9]

3 Návrh

Specifickým cílem této práce je vytvořit platformu pro popis japonských znaků a vztahů mezi nimi. To v základu implikuje tvorbu databázového systému, ve kterém se projeví propojení všech elementů, které jsme probrali v předchozí kapitole, jako jsou kanji a jejich radikály, různé způsoby čtení kanji, jejich grafický tvar a kanji složeniny. Jelikož čtení se uvádí v kaně, bude potřeba do systému zakomponovat všechny tři japonské abecedy a je také potřeba zvolit vhodnou informační platformu se snadnou podporou UTF jak na databázové, tak na aplikační úrovni.

Databázový systém je tedy jádrem problému a aplikační stránka je volitelná, ale měla by demonstrovat funkčnost databáze. Byla možnost zvolit aplikaci podle uvážení, nakonec došlo k výběru aplikace, která bude sloužit jako výuková a učební pomůcka s širším sortimentem využití jak pro začínajícího japonštináře, který bude používat aplikaci k vlastní výuce a opakování, tak pro pokročilejšího studenta, který ji může používat pouze jako pomůcku nebo třeba slovník.

Aplikace to bude webová, aby uživatel nebyl nucen procházet žádnou zdlouhavou instalací a díky možnostem, které dnes web přináší. Další výhody tohoto přístupu jsou multiplatformnost a přístup k aplikaci z více míst (domov, práce, v budoucnu mobilní přístroje apod.).

3.1 Databázové schéma

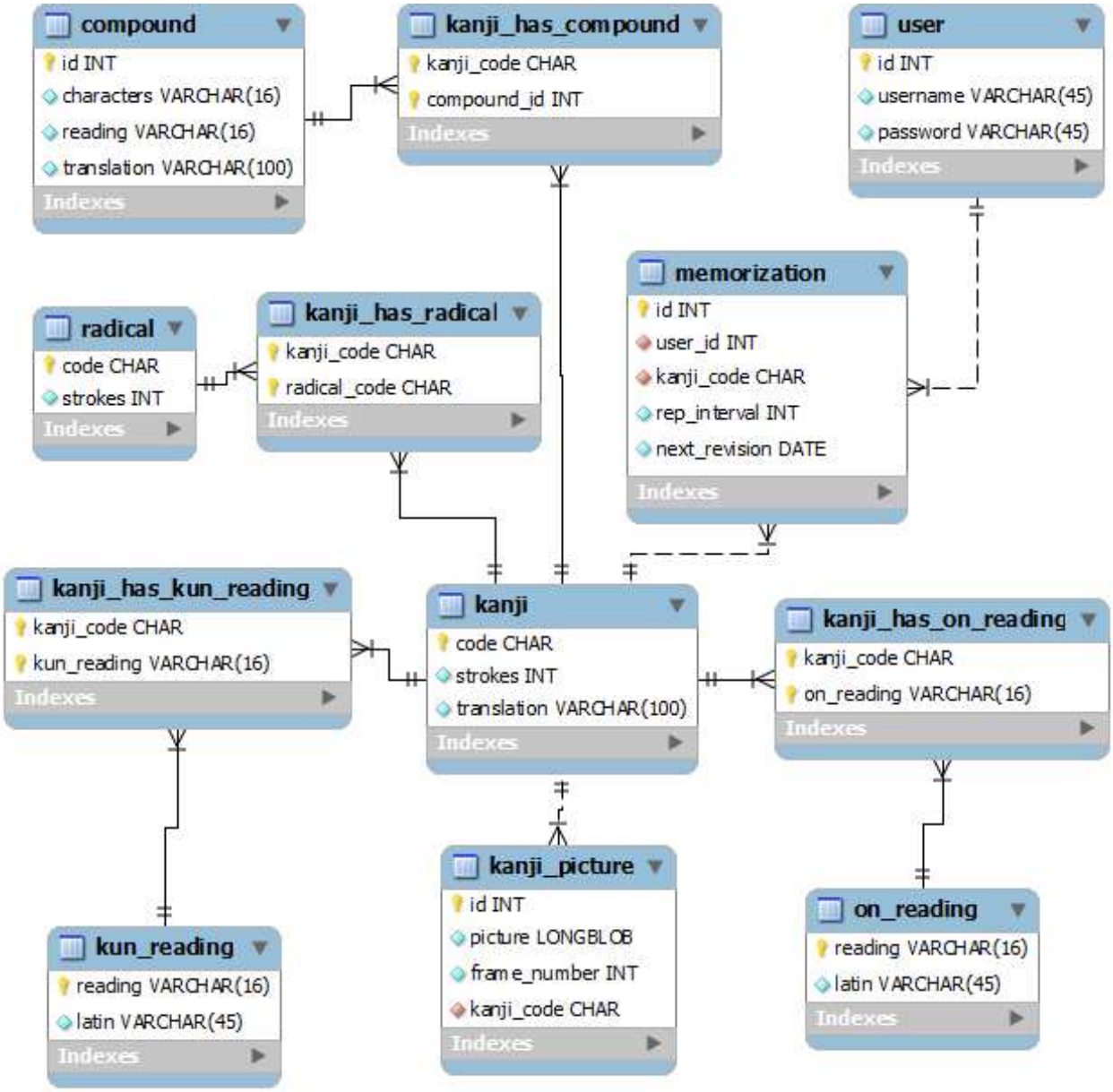
Návrh databáze je první krok k úspěšnému vytvoření tohoto systému. Pro tento účel byl použit ER diagram (na další straně). Centrální entitou celého schématu je entita kanji. Je to logické, protože všechny ostatní prvky japonštiny se dají odvodit z kanji, nebo k němu mají nějaký vztah, pokud tedy nejde o slova přejatá, která se do kanji velmi často nepřepisují. Databáze obsahuje velmi mnoho *m* ku *n* vztahů, zde ovšem nebyla možnost volby, vzhledem ke komplikovanosti jazyka.

Databáze se dá rozdělit na tabulku kanji a několik součástí, které se na tuto tabulku váží. Tabulka kanji samotná obsahuje pouze překlad a počet tahů štětcem. Součásti na kanji navázané jsou:

- **Čtení** – zahrnuje entity *on_reading*, *kun_reading*, *kanji_has_on_reading* a *kanji_has_kun_reading*. Entity *on_reading* a *kun_reading* obsahují jednotlivá čtení kanji. Ke každému čtení je pak přiřazen i přepis do latinky, pro potřeby vyhledávání. Jelikož každé kanji může mít přiřazeno žádné či několikero čtení a každé čtení může náležet jednomu či více různým kanji, bylo nutné přidat i vazební tabulky *kanji_has_on_reading* a *kanji_has_kun_reading*.
- **Radikály** – zahrnuje entity *radical* a *kanji_has_radical*. U radikálů je opět uveden počet tahů štětcem, protože radikály samotné se do slovníku řadí podle této metriky.

A i zde opět platí, že jeden znak může mít více radikálů a každý radikál mívá více znaků, proto opět vazební tabulka.

- **Složeniny** – zahrnuje entity *compound a kanji_has_compound*. Jednotlivé složeniny kanji obsahují několik znaků kanji plus případně nějakou hiraganu, která je jednoduše vložena do pole *characters* jako řetězec. Tomuto řetězci je pak přiřazeno čtení (v hiraganě) a překlad. Následně jsou mu pak přes vazební tabulku přiřazena kanji, která se v něm vyskytují.
- **Grafické znázornění** – zahrnuje pouze entitu *kanji_picture*. Zde je základním elementem obrázek kanji uložený jako Binary Large Object (BLOB). Aby mohl uživatel graficky vidět, jakým způsobem se kanji kreslí, je ke každému kanji uloženo tolik obrázků, kolik má kanji tahů, a každý obrázek má přiřazen index. Takže například pro kanji 人, které má dva tahy, jsou uloženy dva obrázky, jeden s nakresleným prvním tahem a druhý s nakreslenými oběma tahy.
- **Memorizace** – tato část se týká opakování slovíček a bude podrobněji popsána v následující kapitole, kde vysvětlení lépe zapadá do kontextu.



Obrázek 3.1: Databázové schéma

3.2 Funkce systému

V této sekci je slovně popsán návrh základních funkčních prvků systému a návrh jednotlivých modulů systému. V rámci analýzy specifik, jaké by měla ideální japonštinářská učební pomůcka mít, jsem dospěl ke čtyřem základním modulům, které by měla taková aplikace poskytovat. Systém je tedy modulární, což zajistí lepší přehlednost zobrazovaných dat a jejich funkce. Zároveň ale není potřeba výraznější editace dat, dochází tedy z větší části k zobrazování dat dané databáze, než k jejímu pozměňování. To má vliv také na tvar aplikace. K editaci bude docházet pouze v modulu pro opakování slovíček, který má být unikátní pro každého přihlášeného uživatele. Tudíž je potřeba také při startu systému zobrazit přihlašovací obrazovku, ve které dojde k autentizaci uživatele pomocí přihlašovacího jména a hesla. Následuje popis jednotlivých modulů aplikace.

3.2.1 Modul Kanji

Hlavní a nejzákladnější modul aplikace. Tento modul zobrazuje všechna kanji v databází i se všemi jejich dodatečnými informacemi, které byly probrány v kapitole 3.1. Zobrazení bude tabulkové a mělo by být přehledné. V tomto modulu však nebude pouze tabulka kanji, protože je potřeba více funkcionality. Okno tedy bude rozděleno do tří podoken – tabulka kanji, detail kanji a vyhledávání v kanji.

Samotná tabulka kanji bude zobrazovat základní informace ke kanji, jako např. samotný znak, počet tahů štětcem, čtení on a kun (z databáze budou vytažena např. všechna čtení on a poté vložena do jediného řetězce, odděleného čárkami a vložena do tabulkového pole) a radikály (ty budou opět vytaženy všechny z databáze a spojeny do jediného řetězce). Je nutné, aby tabulka uměla sortovat, nejlépe pouhým kliknutím na hlavičku sloupce, podle kterého se bude řadit. Také by bylo vhodné, aby se tabulka uměla zobrazit v provedení po skupinkách, stejně jako tomu bývá v tištěných japonských slovnících (ať již jsou skupinky nebo chcete-li kapitoly oddělené po radikálech nebo po počtu tahů – často je to obojí).

Podokno detail kanji bude zobrazovat informace, které se ze zřejmých důvodů do hlavní tabulky nevejdou. Když tedy uživatel klikne na vybrané kanji v hlavní tabulce, v tomto podoknu se mu zobrazí detail. Zde nás zajímají hlavně složeniny pro každé kanji. Každá taková složenina má navíc své vlastní čtení a svůj překlad, takže je potřeba zvláštní tabulka jenom pro složeniny, která se zobrazí v tomto podokně. Dále je vhodné mít grafické znázornění znaku v obrázku, aby člověk získal lepší představu o tom, jak znak vypadá a hlavně jak se kreslí. Představu o kreslení uživatel získá tím, že pod obrázkem budou tlačítka doleva a doprava, což umožní uživateli, aby přidával nebo odebíral jednotlivé tahy štětcem a naučil se tak správné pořadí kreslení znaku. Nakonec by v tomto podokně

mělo být i tlačítko, které umožní uživateli přidat kanji do databáze opakování, pokud má zájem se ho naučit.

Poslední podokno, vyhledávání v kanji, bude obsahovat seznam textových polí, které budou sloužit jako filtry pro kanji v hlavní tabulce. Když tedy uživatel zadá text do jednoho či více těchto textových polí a klikne na tlačítko „Hledat“, které se bude nacházet pod nimi, hlavní kanji tabulka se překreslí, aby zobrazila pouze kanji odpovídající zadaným filtrům. Pokud se pole opět zpátky vynulují, zobrazí se zase celá kanji tabulka.

3.2.2 Modul Slovník

Modul Slovník poslouží uživateli jako rychlý nástroj pro vyhledání žádaného slovíčka a vypsání všech možných variant zapsání tohoto slova. Při vyhledání slovíčka totiž může dojít nejen k vyhledání kanji, ale také složeniny. Aby toho nebylo málo, může být vyhledáno více různých čtení, proto má tento modul smysl nejen sám o sobě, ale i při případném dalším vylepšování aplikace. Uživatel tedy zadá vyhledavané slovíčko do textového pole, zvolí hledat a aplikace vrátí tabulku, ve které bude kanji nebo složenina, včetně čtení a překladu všech jeho možných významů, aby uživatel mohl vybrat, které se mu bude hodit nejvíce.

3.2.3 Modul Editor kany

Při průzkumu internetu na japonštinu vyplynulo najevo, že by bylo vhodné mít nějaký jednoduchý nástroj na rychlou konverzi znaků kany do latinky a zpět, aspoň do chvíle, než student kanu ovládne. A nejenom to, dnešní osobní počítače většinou nemají implicitně nainstalovanou podporu japonské abecedy a hlavně podporu psaní japonských znaků. Pokud chce tedy uživatel napsat nějakou větu v kaně, musí zdlouhavě vyhledávat znaky na internetu, aby je pak mohl použít jinde (třeba při vyhledávání v internetových vyhledávačích). Tento problém bude řešit modul Editor kany.

Uživatel si vybere z vyskakovacího menu směr převodu (kana=>latinka nebo obráceně) a pak už jenom použije velké textové pole. V případě převodu latinka=>kana pak uživatel může psát do textu japonský text v romanizaci (použil jsem romanizaci Hepburn) a při každém zmáčknutí mezerníku se text převede do kany. Při psaní malými písmeny se text převede do hiragany, při psaní velkými písmeny do katakany. Převod také funguje pro tečky, čárky, otazníky apod., které jsou pro japonštinu specifické. Příklad použití by tedy byl:

- Uživatel zadá do pole text „watashi“ (*já*), zmáčkne mezerník a text se převede na „わたし“
- Poté píše dál a napíše „wa“ (oddělovač přísudkové části – tzv. partikule) a po zmáčknutí mezerníku se původní text pole „わたし wa“ změní na „わたし わ“
- Za to dále napíše „CHEKO“ (*český*, je potřeba velkými písmeny, protože cizí jména musí být v katakaně) a text pole se změní na „わたし わ チェコ“

- Nakonec doplní „jin desu.“ (*osoba* a *být* + *tečka*) a výsledkem je text „わたし わ チェコ じん です。“ (watashi wa cheko jin desu – *já jsem Čech*) – všimněte si kolečkového tvaru japonské tečky

Systém umí i dvojité kany a pomocné kany. Stejně tak, pokud uživatel změní převod textu na kana=>latinka, vloží kurzor do textového pole a zmáčkne spacebar, pole se celé přeloží zpátky do latinky. Tento směr překladu ovšem neslouží pouze ke zpětné kontrole převodu, ale hodí se například i při pročítání japonských textů na internetu, pokud si uživatel není jistý, že nějaký text v kaně čte správně, převaděč mu ukáže správnou výslovnost.

3.2.4 Modul Opakování slovíček

Modul obsahuje jednoduchou aplikaci na opakování znaků kanji, které si uživatel vybere v modulu Kanji. Mnoho lingvistů a polyglotů se shoduje, že pokud chcete umět nějakou řeč, z 99% stačí mít slovní zásobu. [10] Také to potvrzuje mezi lingvisty slavné pravidlo „Nauč se 10000 vět a umíš jazyk“.[11] Základem učení každého řeči jsou tedy takzvané učební kartičky (angl. *flashcards*), které by měl student nosit všude s sebou a ve volných chvílích si opakovat. Tento systém bude utilizovat simulaci takovýchto kartiček plus metodu opakování zvanou *Rozložené opakování.* [10]

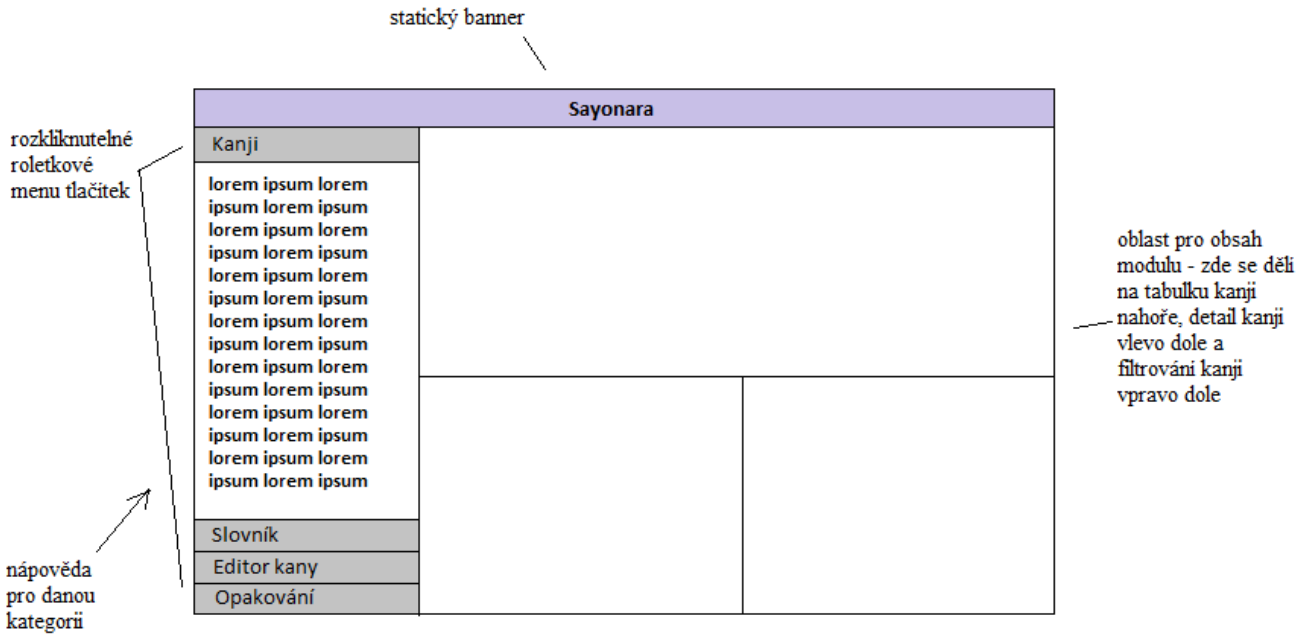
Rozložené opakování

Rozložené opakování je systém učení slovíček, ale i jakýchkoli jiných dat (jako třeba fyzikálních vzorečků), které je potřeba suše memorovat. Vychází ze studií z 60. let minulého století, které prováděl lingvista Paul Pimsleur, známý autor Pimsleurovy jazykové metody. Základem systému jsou neustále prodlužující se intervaly opakování slovíček. Vychází se z principu, jakým se učí lidský mozek, ten si totiž pamatuje pouze data, která se opakují, ale narozdíl od všeobecně rozšířeného povědomí se nemusí opakovat pravidelně, ale intervaly se mohou exponenciálně prodlužovat, čímž se ušetří mnoho energie a nakonec dojde k zapamatování trvalému. Pimsleur došel k intervalům 5 sekund, 25 sekund, 10 minut, 1 hodina, 1 den, 5 dní, 25 dní, 4 měsíce, 2 roky. Stačí tedy 9 dobře načasovaných opakování a slovíčko je zapamatováno natrvalo. [12]

Od zavedení rozloženého opakování do komerčních i nekomerčních systémů jako jsou Supermemo nebo Anki se provedlo mnoho úprav algoritmu a některé použiji i já. Uživatel tedy po otevření modulu bude přivítán prvním kanji, u kterého si bude muset vzpomenout na překlad. Po vzpomenutí klikne na zobrazení odpovědi a zkontroluje si, zda odpověděl správně. Poté pomocí tří tlačítek „lehké“, „střední“, „těžké“ ohodnotí, jak rychle si na slovíčko vzpomněl. Systém pak vyhodnotí interval pro další zopakování. Pokaždé, když uživatel otevře modul Opakování slovíček, systém vyhodnotí, kterým opakovaným kanji už vypršel interval opakování a pouze ty zobrazí. Po zopakování všech slovíček systém zahlásí konec opakování. Každý přihlášený uživatel má vlastní databázi opakovaných slovíček.

3.3 Návrh grafických prvků

Před vytvořením aplikace je vhodné aspoň předběžně rozvrhnout rozložení grafických prvků. Systém by měl být přehledný a pro uživatele intuitivní. Jelikož jde o webovou technologii a jedná se o aplikaci, ne o pouhou prezentační stránku, zakážeme prohlížeči veškeré scrollbary a webstránku implicitně roztáhneme na celé okno a to jak horizontálně, tak vertikálně. Vnitřní prvky tedy musí být schopné se roztahovat a stahovat podle toho, jak se mění velikost okna. Také by měla být možnost měnit velikost jednotlivých podoken systému ručně a dělat si tak místo pro data, které uživatel považuje za důležitější. Návrh pro otevřený modul kanji by tedy mohl vypadat například takto:



Obrázek 3.2 Grafické rozvržení

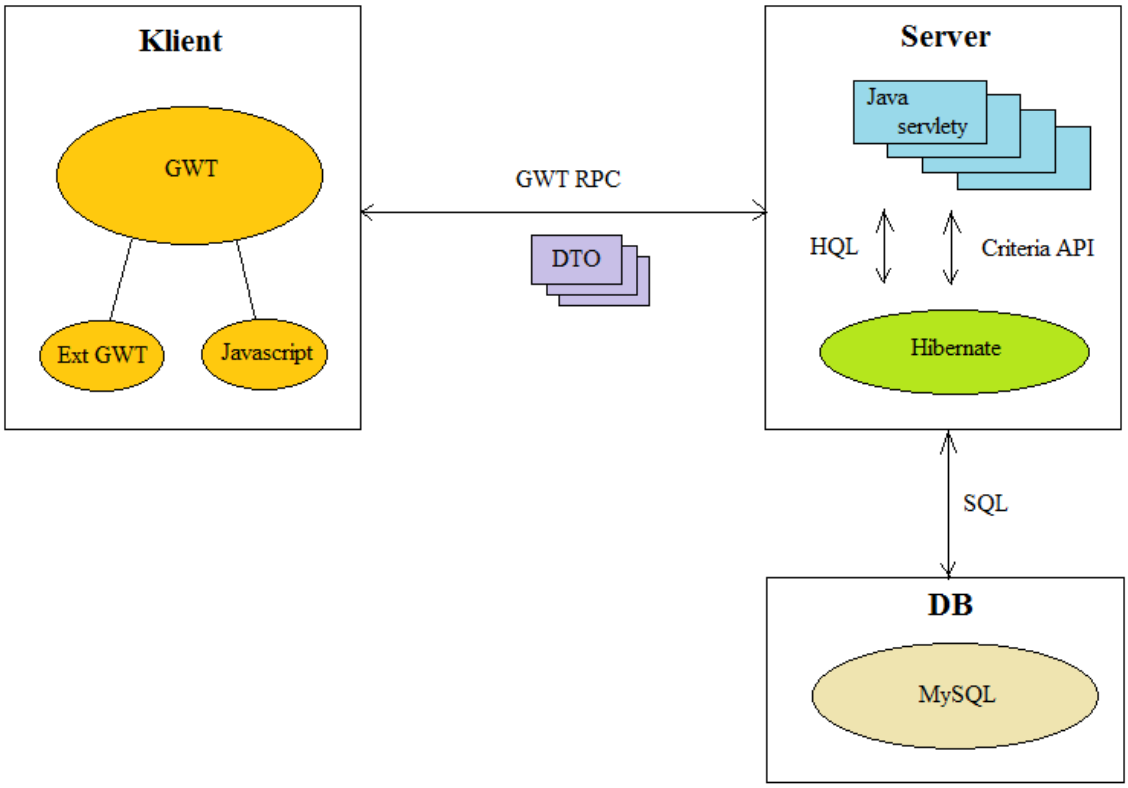
4 Implementace

Jak už bylo řečeno, aplikace byla zvolena jako webová. Jelikož autor této práce neměl před započetím vývoje žádné významnější zkušenosti s vývojem webových aplikací vybíhající nad rámec povinných předmětů na FIT, cílem bylo vybrat takové webové technologie, které usnadní do nejvyšší míry vývoj a údržbu kódu. Také by se měly držet dobrých programovacích zásad a principů, jako jsou objektově orientované programování, separace aplikačního kódu, dat a prezentačního kódu, průhlednou komunikaci serveru a klienta a měl by být snadno refaktorovatelný.

Tyto a další podmínky splňuje kombinace použitých a do sebe dobře zapadajících technologií, předběžně popsaných zde (viz také obr. 4.1):

- **Klientská část** – její základ tvoří knihovna Google Web Toolkit, jejímž principem je, že vývojář programuje kód v Javě, který je pak následně přeložen do tradičních webových technologií (XHTML, JavaScript, CSS). Dokáže zabalit i komunikaci se serverem, tedy Ajax/JSON je zabalen do GWT RPC neboli vzdáleného volání procedur na serveru. Pro usnadnění práce s grafickou stránkou je pak použita knihovna Ext GWT, což je nástavba GWT, která poskytuje velké množství předkonfigurovaných webových widgetů zabalených do javovských tříd. GWT umožňuje vkládat javascriptový kód přímo do svého javovského kódu a komunikovat s ním, což jsem ve svém projektu také využil.
- **Serverová část** – serverová část je složena hlavně z javovských servletů, které běží na webovém serveru a poskytují servisní metody pro komunikaci s databází, které jsou volány z klienta. Tyto metody většinou vrací objekty DTO (data transfer object), což jsou speciální serializovatelné objekty, které klient následně rozbaluje a používá ve své aplikaci. Ani při komunikaci s databází nemusí server vybočovat z používání pouze javovského kódu, protože databázové objekty jsou zabaleny do javovských tříd pomocí knihovny Hibernate, která vytváří s každé tabulky jednoduchou beanu se set a get metodami. S těmito třídami je pak možno komunikovat přes metody Criteria API, které programátora naprosto zbavuje nutnosti používat SQL a je elegantním řešením databázových volání. Případné jednodušší dotazy se dají řešit použitím HQL volání, což je speciální varianta SQL pro knihovnu Hibernate. Za databázový server byl zvolen MySQL server.

Výsledkem tohoto uspořádání je jednota kódu, který je kompletně napsaný v Javě od klienta přes server až po komunikaci s databází. Díky tomu mohlo být také použito přehledné vývojové prostředí Eclipse IDE, které velice usnadňuje refactoring, jeho doplňky umožňují rychlou změnu dat v databázi, snadný přístup na SVN a dokonce debugování klientského kódu spuštěného ve webovém prohlížeči. Navíc je možno současně debugovat klientský a serverový kód a jejich komunikaci. V následujících kapitolách budou podrobněji vysvětleny jednotlivé použité technologie.



Obrázek 4.1: Schéma použitých technologií

4.1 Použité technologie

4.1.1 Google Web Toolkit

GWT je vývojová knihovna pro sestavení a optimalizaci komplexních aplikací spouštěných v prohlížeči. Výsledná aplikace však používá pouze webové technologie jako např. HTML, Javascript a CSS, není tedy potřeba speciální plugin pro jejich spuštění, jako je tomu v případě technologií Flash, ActiveX nebo MS Silverlight. Cílem knihovny je vytvořit vhodné vývojové prostředí bez toho, aby programátor musel být expert na výstřednosti jednotlivých webových prohlížečů, XMLHttpRequest nebo Javascript. GWT stál v základu vývoje např. aplikací Gmail, Google Wave nebo AdWords.

GWT SDK poskytuje sadu javovských tříd, které umožňují programátorovi psát Ajaxové aplikace v Javě a zkompilovat kód do vysoce optimalizovaného Javascriptu, který běží na všech prohlížečích, včetně těch mobilních např. na Androidu nebo iPhone. Nicméně cokoli, co je možné udělat přímo v DOM nebo v Javascriptu, je možno udělat i v GWT. Co je nejdůležitější, GWT poskytuje rychlou cestu k debugování a krokování běžící aplikace přímo v prohlížeči pomocí GWT pluginů do známých prohlížečů.

4.1.2 Ext GWT

Ext GWT je grafická knihovna, která není vyvíjena v Googlu a slouží jako doplněk ke knihovně GWT. Obsahuje rozsáhlou sadu widgetů implementovaných do javovských tříd. Jsou zde prvky pro snadné rozložení stránky, tabulky, vyskakovací okna, záložky, grafy, menu apod. Třídy jsou plně přizpůsobitelné a obsahují velké množství funkcionality, které vyvojář může využít.

4.1.3 GWT Remote Procedure Calls

GWT RPC je součástí GWT, kterou Google nabízí pouze jako jednu z možných komunikací se serverem. Jelikož jsem pro serverovou implementaci zvolil javovské servlety, bylo logickou volbou vybrat pro komunikaci se serverem GWT RPC. GWT RPC framework se tedy používá na transparentní volání metod javovských servletů. Framework se pak postará o komunikaci samotnou, která probíhá nad HTTP a metody v mé implementaci vrací třídy DTO, které framework na serveru serializuje a na klientu deserializuje. DTO (data transfer object) jsou speciální serializovatelné třídy vytvořené na míru tomu kterému volání servletových metod.

Důležité je podotknout, že veškerá RPC volání jsou asynchronní, takže nezdržují práci klientské aplikace. Také je příznivé, že knihovna Ext GWT plně podporuje RPC a jeho asynchronní volání, takže například při vykreslení obrazovky, na které jsou tabulky s daty z databáze, Ext GWT nejprve vykreslí celou stránku a všechny tabulky se správným rozložením a teprve až přijdou asynchronně data z databáze tyto data doplní do tabulek. [13]

4.1.4 GWT servlety

Serverová část kódu, která je volána z klienta, se často pojmenovává jako *service*. Implementace GWT RPC *service* je založena na dobře známé architektuře Java servletů. V klientském kódu se používá automaticky vygenerovaná proxy třída, která spouští volání na servletu. GWT se stará o serializaci objektů, které běhají tam a zpět – parametry volání metod a návratová hodnota.[17]

4.1.5 Hibernate

Hibernate je ORM (object-relational mapping) knihovna pro jazyk Java. Její význam spočívá v mapování databázových objektů na doménový model javovských tříd.[14] Výhodou tohoto přístupu je zjednodušená práce s databází, kdy programátor při volání z databáze dostává kupříkladu seznam javovských objektů. Pro ilustraci, když například požádá z tabulky *Kanji* o všechny záznamy s překladem *Oběd*, daná metoda mu vrátí *List<Kanji>*, kde každý objekt *Kanji* pak obsahuje data o celém záznamu. Objekt *Kanji* je tedy javovská beana, která obsahuje pouze privátní proměnné s daty a veřejné get a set metody.

Mapování javovských tříd na databázové tabulky je možno udělat buď pomocí XML souboru nebo pomocí anotací přímo v javovském kódu. Tato implementace používá kombinaci obojího. Zároveň je zde možnost pomocí anotací namapovat vztahy n...1 a n...m.

4.1.6 HQL a Criteria API

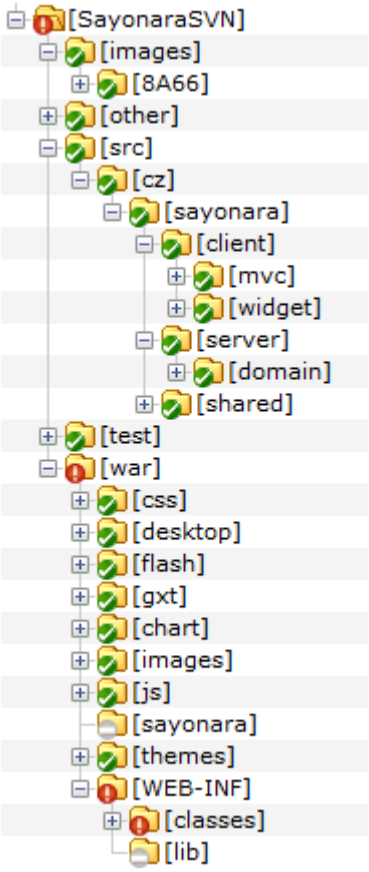
Pro komunikaci s doménovým modelem Hibernate se používá jeden z těchto dvou prostředků:

- **HQL** – neboli Hibernate Query Language je jazyk založený na SQL, který však jako parametry nepoužívá databázové tabulky, ale přímo třídy doménového modelu Hibernate. Jelikož jsou přímo javovské objekty, je také možno pomocí tečkové syntaxe získat přístup k jejich atributům apod. [15]
- **Criteria API** – je použitelnější pro složitější dotazy z databáze. Postupně se přidávají kritéria a filtry, podle kterých se mají data vybrat. Kritéria se dají budovat do složitých stromů a to vše bez použití SQL syntaxe. [16]Zde je například ukázka z mého softwaru, která získá z databáze pouze kanji, která uživatel odfiltruje podle složeného filtru:

```
Criteria crit = session.createCriteria(Kanji.class);
crit.createCriteria("kunReadings").add(Restrictions.eq("reading", kunReading));
crit.createCriteria("onReadings").add(Restrictions.eq("reading", onReading));
crit.add(Restrictions.like("translation", translation, MatchMode.ANYWHERE));
crit.createCriteria("radicals").add(Restrictions.eq("code", radical));
crit.createCriteria("compounds").add(Restrictions.eq("characters", compound));
crit.add(Restrictions.eq("strokes", strokes));
List<Kanji> listKanji = crit.list();
```

4.2 Implementace prostředí

Projekt byl vyvíjen v prostředí Eclipse, což s sebou přináší specifickou adresářovou strukturu, stejně tak GWT má svojí specifickou adresářovou strukturu. Popisovat obsah všech složek by pro tuto práci nemělo smysl, takže popíšu jenom ty pro práci podstatné. Adresáře, jejichž jméno začíná tečkou, jsou pomocné adresáře buď Eclipse nebo SVN klienta a jsou spolu s jinými adresáři vyfiltrovány z tohoto seznamu (to platí i pro obr 4.2).



Obrázek 4.2: Adresářová struktura

- **images** – obsahuje grafické znázornění kanji, každá vnitřní složka patří jednomu kanji a je v ní několikero obrázků kanji v různých stádiích kresby. Pojmenování složky pak vychází z UTF kódování daného znaku.
- **other –** obsahuje zaprvé soubor databázového modelu pro MySQL Workbench. Po otevření tohoto souboru v MySQL Workbench je možno vygenerovat celou databázi jediným kliknutím. Nelze ovšem vygenerovat data a k tomu slouží insert skripty v SQL formátu, které jsou také přítomny v této složce.
- **src –** obsahuje všech 46 zdrojových javovských souborů aplikace v javovské adresářové struktuře. Kromě toho je zde důležitý soubor *hibernate.cfg*, kde je stanoveno, které zdrojové soubory budou mapovány jako doménové objekty databáze. V klientské části je spouštěcí třída aplikace, několik tříd pro komunikaci se serverem a dvě složky. Složka **mvc** obsahuje controllery a view návrhového vzoru MVC použitého na klientovi, složka **widget** obsahuje přizpůsobené webové widgety, které jsou pak použity v aplikaci. Serverová část několik servletů a složku **domain**, která obsahuje zmiňované třídy doménového modelu Hibernate, které v podstatě zabalují databázové tabulky MySQL a komunikaci s nimi. Složka **shared** obsahuje pomocné třídy používané jak na klientovi tak na serveru a hlavně DTO objekty pro výměnu mezi serverem a klientem.
- **test** – obsahuje souboru použité v testování JUnit

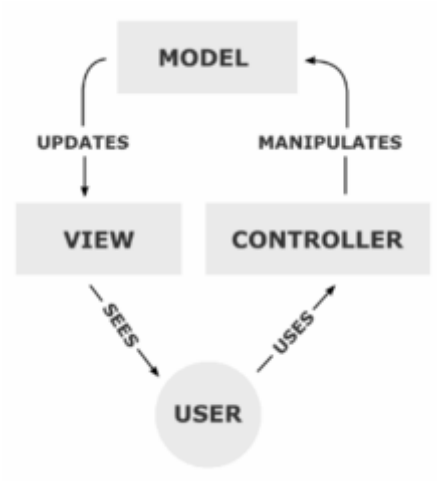
- **war** – obsahuje statické zdroje, které mohou být použity veřejně, jako jsou obrázky, styly, HTML stránky apod. Výjimku tvoří složka **WEB-INF**, která obsahuje zkompilované javovské třídy a složka **sayonara**, která obsahuje zkompilovaný javascript vytvořený z klientské části oněch javovských tříd. Ve složce **WEB-INF** je kromě toho i složka **lib**, která obsahuje .jar archivy všech zmíněných technologií, které jsou použity v kódu. Je zde *hibernate* a *hibernate-annotations*, *mysql-connector* pro připojení databáze, *slf4j* a *log4j* pro přihlašování servletů do Apache serveru, *gxt* a *gwt-servlet* (*gxt* je obvyklá zkratka pro Ext GWT) apod. Když se vrátíme zpátky k obsahu složky war, tak všechny zbývající složky jsou statické složky patřící buď GWT nebo GXT a obsahují soubory CSS, obrázky, javascripty apod. Pouze složka **js** obsahuje javascriptový kód, který jsem sepsal za účelem použití v Kana editoru a následně integroval do javovských tříd. Za zmínku stojí ještě soubor Sayonara.html uložený ve složce **war**, který je jediným editovatelným html souborem a vstupní stránkou do systému, a také soubor web.xml ve složce **war/WEB-INF**, který obsahuje seznam servletových tříd a jejich mapování na URL.

4.2.1 Klientská část

Vstupní třída klientské části je třída *Sayonara* a její metoda *onModuleLoad()*, zde se zaregistrují všechny serverové servisy do jednoho centrálního objektu (třída *Registry*), která slouží jako globální proměnná pro celou aplikaci. Také se zde zaregistrují všechny controllery návrhového vzoru MVC a pošle se zpráva jim *Init*. Ve stejné složce se také nacházejí dvě servisní proxy třídy, vždy v páru asynchronní a obyčejná. To vychází z designu GWT. *UserService* se stará pouze o serverové volání týkající se uživatele jako např. *login(), logout()* apod. Obsáhlá třída *SayonaraService* obsahuje všechny ostatní serverové volání. Třída *AppEvents* je pak výčtová třída, která slouží jako jediné místo s přehledem všech zpráv (eventů), které si mezi sebou posílají controllery a view MVC prostředí.

4.2.2 MVC

Model-View-Controller je často používaná softwarová architektura, která odděluje doménovou logiku od uživatelského vstupu a prezentace.[18] V našem systému se modelem myslí Hibernate třídy a Controller a View jsou uloženy na klientské části. Víceméně existuje pro každu ze čtyř základních modulů (Kanji, Slovník, Kana editor a Opakování) jeden Controller, který rozesílá eventy dvěma třídám View (jedna pro rozbalovací menu a jedna pro samotný obsah okna). Nad tím vším existují jeden hlavní Controller a jeden hlavní View, který rozvrhuje stavbu celého okna.



Obrázek 4.3: MVC

Komunikace mezi prvky probíhá systémem „uživatel klikne na tlačítko, view pošle zprávu controlleru, ten zažádá přes servlet databázi o data a jelikož jsou volání na servlet asynchronní, data pak příjdou rovnou zpátky na view, který je zobrazí“.

Podsložkou složky client je složka widget, která obsahuje několikero malých widgetů jako je například logovací dialog nebo box s obrázkem kanji v modulu Kanji.

4.2.3 Serverová část

Na serverové části stojí za zmínku třídy *ImageServlet, UserServiceImpl a SayonaraServiceImpl*.

- *ImageServlet* – malý servlet, který slouží k automatickému naplnění obrázků kanji ze složek do databáze, obsahuje jedinou metodu
- *UserServiceImpl* - slouží pro práci s uživateli, obsahuje metody pro login, logout, register apod. Upravuje session podle toho, který uživatel je přihlášený. S doménovým prostředím Hibernate komunikuje přes HQL.
- *SayonaraServiceImpl* – shrnuje do sebe všechny ostatní metody pro získávání dat z databáze a komunikuje jak přes HQL, tak Criteria API. Obsahuje například *List<VocabularyEntry> searchVocabulary(String text)*, která prohledá tabulky Kanji a Compound na překlad, která odpovídá textu, nebo části textu (prohledává tedy i podřetězce). Další důležitá metoda je *List<KanjiDTO> getKanji()*, která v modulu Kanji vrací obsah celé databáze pro zobrazení v hlavní tabulce. Z KanjiDTO objektů je pak na klientovi naplněna třída *Store*, kterou pak používá třída *Grid* pro zobrazení v tabulce.

5 Závěr

Výslednou práci jsem pojal jako ukázkovou aplikaci, která má vhodnou kostru a je postavena na snadno rozšiřitelném kódu, ale zasloužila by ještě mnoho úprav a rozšíření. Například by přišla vhod dobrá podpora okurigany (skloňování a časování), výborné by také bylo, kdyby editor kany uměl přepisovat nejen z latinky do kany, ale také následně do kanji (s možností výběru, které kanji měl uživatel na mysli). Taková aplikace by pak byla široce použitelná a po přeložení do více světových jazyků myslím, že by byla na webu i profitabilní.

Projekt jsem pojal jako dobrý způsob, jak si oprášit svoje zkušenosti s japonštinou, kterou jsem znal už před započetím projektu, ale kterou jsem nikdy nevyužil (do Japonska jsem se nakonec nikdy nepodíval). Hned po započetí projektu jsem udělal pár základních rozhodnutí, kudy se bude aplikace ubírat, a získal jsem velký zájem o programování webových aplikací, o kterých se dá nadneseně říct, že jsou Web 2.0. Udělal jsem si přehled o tomto poli a o jeho budoucnosti a uvědomil jsem, že znalosti, které jsem nabyl, vůbec nebudou zbytečné a budu je moci použít i v nadcházejících letech.

Literatura

[1] Lewis, M. Paul (ed.), 2009. Ethnologue: Languages of the World, Sixteenth edition. Dallas, Tex.: SIL International. ISBN: 978-1-55671-216-6

[2] Japanese Language Profile – How to learn any language [online]. [cit. 2010-05-05] Dostupné na URL: http://how-to-learn-any-language.com/e/languages/japanese/index.html

[3] Wikipedia: Japanese Language [online]. [cit. 2010-05-05] Dostupné na URL: http://en.wikipedia.org/wiki/Japanese_language

[4] Association for Japanese-Language Teaching : Japanese for Busy People (revised editition) , Kodansha International, 1995. ISBN-10: 4770018827

[5] Wikipedia: Romanization of Japanese [online]. [cit. 2010-05-06] Dostupné na URL: http://en.wikipedia.org/wiki/Romanization_of_Japanese

[6] Kaiser, Stephen (1991). Introduction to the Japanese Writing System. Tokyo: Kondansha International. ISBN 4-7700-1553-4.

[7] James W. Heisig: Remembering the kanji (A complete course on how not to forget the meaning and writing of Japanese characters) Fourth edition, ISBN-10: 0824831659

[8] Wikipedia: Stroke (CJK character) [online]. [cit. 2010-05-08] Dostupné na URL: http://en.wikipedia.org/wiki/Stroke_%28CJK_character%29

[9] About.com: all about radicals –[online]. [cit. 2010-05-08] Dostupné na URL: http://japanese.about.com/library/weekly/aa070101a.htm

[10] Barry Farber : How To Learn Any Language: Quickly, Easily, Inexpensively, Enjoyably and on Your Own, ISBN-10: 0806512717

[11] learnanylanguage.com: 10000 sentences [online]. [cit. 2010-05-10] Dostupné na URL: http://learnanylanguage.wikia.com/wiki/10000_Sentences

[12] Alan Baddeley: Human Memory: Theory and Practice, Prentice Hall, ISBN: 0205279481

[13] Communicating with server – Google Web Toolkit [online]. [cit. 2010-05-13] Dostupné na URL: http://code.google.com/intl/cs/webtoolkit/doc/latest/tutorial/clientserver.html

[14] Hibernate Reference Manual [online]. [cit. 2010-05-13] Dostupné na URL: http://docs.jboss.org/hibernate/stable/core/reference/en/pdf/hibernate_reference.pdf

[15] Introducing HQL – developer.com [online]. [cit. 2010-05-13] Dostupné na URL: http://www.developer.com/open/article.php/3322131/Introducing-HQL-The-Object-Oriented-Query-Language-from-Hibernate.htm

[16] The Criteria API – oreilly.com [online]. [cit. 2010-05-15] Dostupné na URL: http://oreilly.com/java/excerpts/harnessing-hibernate/the-criteria-API.html

[17] Making Remote Procedure Calls – Google Web Toolkit [online]. [cit. 2010-05-15] Dostupné na URL: http://code.google.com/intl/cs/webtoolkit/doc/latest/tutorial/RPC.html

[18] Steve Burbeck, Ph.D.: How to use Model-View-Controller [online]. [cit. 2010-05-15] Dostupné na URL: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

Seznam příloh

Sayonara - pomocník pro japanistu

Hlavní menu

- Kanji
- Slovník
- Editor kany
- Opakování slovíček

Kanji

Kanji	Počet tah	Překlad	Čtení On	Čtení Kun	Radikál(y)
試	13	pokus, experiment	シ	こころ, ため	言
丁	2	ulice, město	チョウ, テイ	ひのと	一
氷	5	rampouch, led, kroupa	ヒョウ	こおり, ひ	冫, 水
与	3	udělit, odměnit	ヨ	とも, ともに	一
尤	4	rozumný, přijatelný	ユウ	とが, もっと	冫, 尢
永	5	věčnost, dlouhý	エイ	と, のり	冫, 水
必	5	jistý, nevyhnutelný	ヒツ	かなら	冫, 心
内	4	uvnitř, mezi	ダイ, ナイ	いと, うち	丨, 入, 冂
中	4	uprostřed, střed	チュウ	あたる, なか	丨
弔	4	kondolence, pohřeb	チョウ	とぶら, とむら	丨, 弓

Detail Kanji

Složenina	Čtení	Překlad
試す	ためす	vyzkoušet
試みる	こころみる	pokusit se, zkusi
試合	しあい	zápas, hra
公試	こうし	státnice
試写	ししゃ	náhled, ukázka

試

< >

Přidat do opakování

Hledat v Kanji

On čtení:

Kun čtení:

Radikál:

Počet tahů:

Složenina:

Překlad:

Hledat