VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

SUDOKU

SUDOKU

2014

Štefan Horvát

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Štefan Horvát**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: SUDOKU
SUDOKU

Zásady pro vypracování:

Sudoku je logická hra s čísly, o kterou je v posledních letech velký zájem. Druhů her Sudoku existuje několik. Cílem práce je navrhnout a naimplementovat informační systém a zároveň algoritmus, na generování čísel pro klasický typ této hry (čísla 1 – 9). IS bude nabízet uživateli několik funkcí, například možnost uložení rozehrané hry, kontrolu správnosti vyřešeného Sudoku, případně uživatelem zadaného, nápovědu během hraní, hru na čas apod.

1. Seznamte se s problematikou SUDOKU.
2. Seznamte se s Yii Frameworkem.
3. Navrhněte a naimplementujte IS Sudoku s využitím PHP (Yii Framework) a MySQL.
4. Aplikaci otestujte.

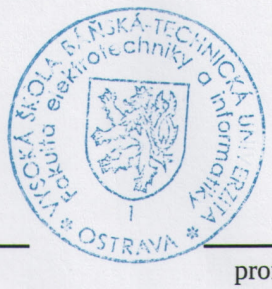
Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Súhlasím so zverejnením tejto bakalárskej práce podľa požiadaviek čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostrave 30. apríla 2014 .....

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 30. apríla 2014 .....

Ďakujem svojmu vedúcemu bakalárskej práce Ing. Petrovi Olivkovi za cenné metodické rady, odborné vedenie a usmerňovanie pri vypracovaní bakalárskej práce a pánovi Ing. Matúšovi Koprdovi za inštruovanie v štruktúre Yii Frameworku.

Abstrakt

Cieľom bakalárskej práce bolo navrhnúť a implementovať webovú aplikáciu venovanú klasickej hre sudoku pomocou Yii framework a databázy MySQL. Bakalárska práca je rozdelená na viac kapitol, ktoré v postupnosti vysvetľujú princíp hry sudoku a vybraných metód riešenia, charakteristike Yii framework a popisu jeho funkcií. Najdôležitejšou časťou bakalárskej práce je návrh a implementácia webovej aplikácie sudoku.

Kľúčové slová: Yii Framework, sudoku

Abstract

The aim of the thesis is to design and implement a web application dedicated to the classic game sudoku using Yii Framework and MySQL database. The thesis is divided into several chapters that explain the principle of sudoku and solving methods, characteristics of Yii framework and description of its functions. The most important part of the thesis is the design and implement of web application sudoku.

Keywords: Yii Framework, sudoku

Zoznam použitých skratiek a symbolov

CoC	–	Convention over Configuration
MVC	–	Model-View-Controller
Yii	–	Yes, it is!
kód	–	zdrojový kód

Obsah

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Sudoku je logická hra určená pre cibrenie myslenia riešiteľa. Podľa náročnosti ju môžeme zaradiť do stupňov od najľahších, ktoré sú riešiteľné bez pomocných poznámok, po najťažsie, kde musíme použiť tak poznámky, ako aj premyslieť rôzne stratégie riešenia.

Sudoku je európska hra. Prvé zmienky o nej sú datované vo francúzskych časopisoch devätnásteho storočia. Hra prešla rôznymi obmenami, ktoré vyvrcholili do podoby dnešného klasického sudoku. Neskôr nastal rozmach tejto hry aj do sveta. Útlm bol počas prvej svetovej vojny, kedy Sudoku v starej podobe aj zaniklo. Znovuobjavenie sa datuje k roku 1979, kedy bola hra publikovaná pod názvom "miesto pre čísla" Howardom Garnsom. Súčasný názov sudoku nie je ale prvotným. Je vytvorené z japonského "Suji wa dokushin ni kagiru".

V súčasnosti je hra sudoku považovaná za veľmi obľúbenú. Dôkazom sú prílohy rôznych novín a časopisov vo väčšine krajín.

Bakalárska práca je rozdelená na teoretickú a praktickú časť. Teoretická časť je zameraná na priblíženie tématiky sudoku a metód jej riešenia, a zároveň charakteristike Yii framework a popisu jeho funkcií. Najdôležitejšou časťou práce je praktická časť, v ktorej je návrh a implementácia webovej aplikácie sudoku. Cieľom práce bolo vytvoriť funkčnú aplikáciu sudoku s možnosťami výberu riešenia podľa obtiažnosti.

2 Sudoku

Sudoku je logická hra s číslami (môžu byť nahradené aj inými znakmi alebo symbolmi). Klasické sudoku pozostáva z hracieho poľa rozmerov 9x9, ktoré je ešte rozdelené na bloky 3x3. Hra sa riadi jediným pravidlom a to, že v každom riadku, stĺpci a bloku sa môže každé číslo nachádzať iba raz.

Vo všeobecnosti sa dá povedať, že sudoku pozostáva z hracieho poľa o veľkosti NxN blokov, kde každý blok pozostáva z NxN políčok a celé hracie pole je potrebné zaplniť NxN rôznymi číslami podľa vyššie popísaného pravidla.

Existujú rôzne varianty tejto hry, ktorých pravidlá sa trochu líšia, niektoré z nich spomeniem neskôr 2.4.

2.1 História hry

Už koncom 19 storočia sa začali vo Francúzskych novinách objavovať hlavolamy podobné dnešnému sudoku. Išlo o hru magické štvorce s rozmermi 9x9 a vnorenými štvorcami 3x3, kde platilo, že súčty čísiel v riadkoch, stĺpcoch a vo vnorených štvorcoch musel byť rovnaký, avšak polia mohli obsahovať aj viacciferné čísla.

O niekoľko rokov neskôr sa začali vo viacerých Francúzskych novinách objavovať hry, kde mriežka 9x9 mohla obsahovať iba čísla 1-9 ale neobsahovala vnorené štvorce 3x3. Tieto hlavolamy sa objavovali v novinách až do prvej svetovej vojny, počas ktorej však zanikli.

Podľa amerického experta na hlavolamy Willa Shortza, súčasné sudoku pravdepodobne prvykrát vytvoril nadšenec hlavolamov Howard Garns z Indiany a prvykrát ho publikoval v roku 1979 pod názvom "miesto pre čísla". Svoj hlavolam publikoval niekoľko rokov vo viacerých časopisoch. Garns umrel v roku 1989 skôr ako sa sudoku stalo celosvetovo populárnym hlavolamom.

Svoj súčasný názov získalo sudoku až o niekoľko rokov neskôr v Japonsku po druhom premenovaní zo "SUUJI WA DOKUSHIN NI KAGIRU"na sudoku [1].

2.2 Obtiažnosti hry

Existuje viacero spôsobov na určenie obtiažnosti hry sudoku napríklad podľa počtu nevyplnených políčok. V práci použijeme členenie obtiažnosti podľa náročnosti techník, ktoré je potrebné na vyriešenie daného sudoku použiť. Dané techniky rozdelené podľa obtiažností popisujem nižšie.

2.3 Techniky riešenia klasickej varianty hry sudoku

2.3.1 Veľmi ľahké a ľahké

Na rozdiel od ťažších spôsobov riešenia si riešiteľ nepotrebuje robiť žiadne poznámky na to, aby sa pomocou nich dopracoval k riešeniu. (Riešiteľ by mal prísť k riešeniu sudoku bez použitia pomocných poznámok.)

2.3.1.1 Intersections Technika spočíva v hľadaní "priesečníka"pomyselným vyčiarknutím riadkov a stĺpcov, kde sa už nachádza určité číslo a pokiaľ sa v niektorom bloku (v ktorom sa dané číslo ešte nenachádza) nachádza jediné voľné políčko, môžeme doň naše číslo umiestniť. Táto metóda patrí medzi prvé spôsoby hľadania riešenia.

2.3.1.2 Forced Moves Iba o niečo náročnejší spôsob hľadania spočívajúci v tom, že pri vybranom poli neexistuje v danom riadku, stĺpci, alebo bloku iné číslo, ktoré by doň mohlo byť umiestnené.

Príklad 2.1
V stĺpci sa nachádzajú čísla 1, 3, 8, 9 v riadku čísla 2, 4, 7, 8 a v bloku sú už umiestnené čísla 4, 6, 9. Z uvedeného vyplýva, že dané políčko musí obsahovať číslo 5. ■

2.3.2 Priemerné

Pri týchto technikách sa predpokladá, že sa pomocou použitia "ľahších"spôsobov riešenia urobia poznámky o možnostiach výskytu jednotlivých čísiel nasledovne: do každého poľa sa zapíšu všetky čísla, ktoré sa nenachádzajú v danom riadku, stĺpci ani bloku.

Príklad 2.2
V stĺpci, riadku a bloku sa nachádzajú čísla 1, 2, 3, 6, 8. Potom si do uvedeného poľa poznačíme, že sa v ňom môžu vyskytovať čísla 4, 5, 7 a 9 ■

2.3.2.1 Pinned Squares Aj napriek tomu, že máme v poznámkach pre políčko viacero čísel, sa už určité čislo nenachádza ako možnosť v žiadnom inom poli v rovnakom riadku, stĺpci alebo bloku a teda sa musí nachádzať v našom políčku.

Príklad 2.3
Do políčka môžeme umiestniť čísla 3, 4, 7 a v rovnakom riadku máme ďalšie dve nevyplnené polia, obe s možnosťami 4 a 7. Keďže v druhých dvoch poliach sa nemôže nachádzať číslo 3, musí byť umiestnené v ňom. ■

2.3.3 Ťažké

K riešeniu pomocou ťažších spôsobov bude potrebné na základe logických dedukcií eliminovať niektoré čísla z možností niektorých polí.

2.3.3.1 Locked Sets Pokiaľ sa v bloku, stĺpci alebo riadku nachádza N polí, do ktorých je možné umiestniť práve N rôznych čísiel (nemusia byť všetky možnosti pre každé políčko), môžeme zo všetkých ostatných polí odstrániť ako možnosti všetky spomínané čísla.

Príklad 2.4
v bloku sa nachádzajú polia s nasledovnými možnosťami:
1[1] - 1, 4, 7
2 - 4, 7, 8, 9
3 - 1, 4, 6, 8
4 - 4, 7
5 - 1, 4
6 - 2, 4, 5, 7, 8
políčka 1, 4 a 5 obsahujú možnosti 1, 4, 7 čiže sú to tri políčka, v ktorých môžu byť umiestnené práve tri rôzne čísla a teda možeme tieto čísla vylúčiť z ostatných políčok. Po uvedenej úprave nám vzniknú políčka s nasledovnými možnosťami:
1 - 1, 4, 7
2 - 8, 9
3 - 6, 8
4 - 4, 7
5 - 1, 4
6 - 2, 5, 8

■

2.3.3.2 Intersection Removal Táto pokročilá technika nám umožňuje odstrániť niektoré možnosti na základe toho, že o niektorých číslach vieme napríklad to, že sa v niektorom bloku musia nachádzať v konkrétnom stĺpci alebo riadku a teda môžeme danú možnosť vylúčiť na políčkach v ostatných blokoch v danom stĺpci, alebo riadku, prípadne v ostatných políčkach rovnakého bloku.

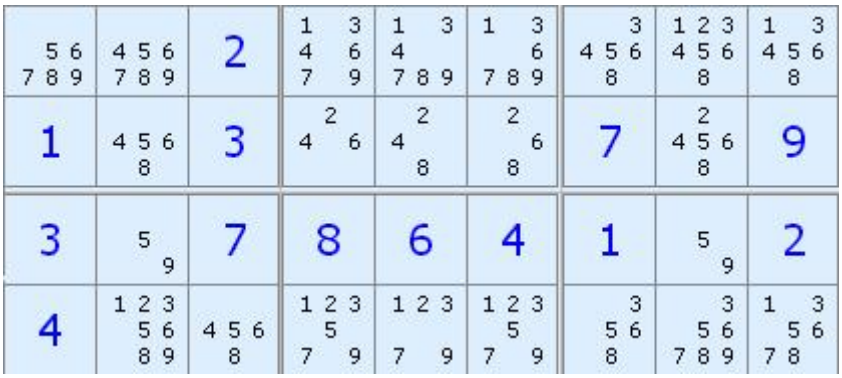
Obr. 1: Intersection

Príklad 2.5
Keď sa pozrieme na obrázok 2.3.3.2 vidíme, že v druhom bloku sa bude číslo 8 nachádzať v prvom riadku a teda v bloku č.1 musí byť číslo 8 v druhom riadku a teda ho môžeme vyradiť z ostaných políčok. ■

[1] číslo nevyplneného poľa - čísla, ktoré je možné zapísať do daného poľa oddelené čiarkou

2.3.4 Veľmi ťažké

2.3.4.1 X-Wing Pri tejto technike hľadáme dva riadky[2] ktoré obsahujú práve dve možnosti rovnakého čísla (možnosti sa nachádzajú v rovnakých stĺpcoch). Z uvedeného vyplýva, že ak bude dané číslo umiestnené na prvej pozícii v prvom riadku, musí byť zároveň umiesnené na druhej pozícii v druhom riadku a naopak a teda v daných stĺpcoch môžeme odstrániť ostatné možnosti nášho čísla.



Obr. 2: X-Wing

Príklad 2.6
Ako vidíme, v druhom a treťom riadku nášho príkladu 2.3.4.1 môžeme číslo 5 umiestniť buď v druhom, alebo ôsmom stĺpci. Aby sme dodržali podmienku, že v každom riadku sa musí každé číslo nachádzať, musíme umiestniť číslo 5 buď v druhom riadku do druheho stĺpca a tretieho riadku ôsmeho stĺpca, alebo ôsmeho riadku druhého stĺpca a druhého riadku ôsmeho stĺpca. V každom prípade bude už číslo 5 obsiahnuté v druhom a ôsmom stĺpci a teda ho môžeme ako možnosť vylúčiť v ostatných poliach daných stĺpcov. ■

2.3.4.2 Swordfish Technika Swordfish je rozšírením techniky X-Wing pri ktorej hľadáme tri riadky[2],v ktorých sa možnosť niektorého čísla nachádza práve v troch stĺpcoch a teda analogicky sa číslo nebude môcť nachádzať v ostatných poliach v uvedených stĺpcoch.

2.3.4.3 Remote Pairs Ďalšia z pokročilých techník riešenia, ktorej názov je odvodený z Locked pairs (locked sets 2.3.3.1), ktoré v tejto technike zohrávajú kľúčovú úlohu. Práve využitím nepárneho počtu Locked pairs môžeme vyradiť danú dvojicu v priesečníku riadka a stĺpca, prípadne bloku prvého člena prvého Locked páru a druhého člena posledného Locked páru. Nesmieme zabudnúť na skutočnosť, že druhý člen jedného páru je prvým členom nasledujúceho páru. Keďže používame nepárny počet párov, vieme, že prvý člen prvého a posledného páru bude obsahovať rovnaké číslo a teda druhý člen posledného páru bude obsahovať druhé z číselných možností.

[2] analogicky pre stĺpce

Príklad 2.7
Máme Locked páry obsahujúce (pre zjednodušenie) čísla 1 a 2. Prvý člen prvého páru sa nachádza v stĺpci X a druhý člen posledného páru sa nachádza v riadku Y. Potom môžeme z políčka v X-tom stĺpci a Y-tom riadku vylúčiť ako možnosti čísla 1 a 2 pretože pokiaľ bude v prvom políčku číslo 1 potom musí byť v poslednom políčku číslo 2 a naopak a teda sa každé z čísel 1 a 2 bude nachádzať v uvedenom stĺpci alebo riadku. ■

2.3.4.4 Hidden Sets Je obdobou metódy pinned square s tým rozdielom, že ide o n-ticu čísiel nachádzajúcich sa práve v n bunkách a teda môžeme z daných buniek odstrániť všetky ostatné možnosti.

Príklad 2.8
Majme v bloku políčka s nasledovnými možnosťami:
1[3] - 2, 3, 4, 5, 6, 7
2 - 1, 2, 3, 7
3 - 1, 2, 3, 4, 5, 6, 7, 9
4 - 1, 2, 3, 7
5 - 1, 3, 7
6 - 8, 9
7 - 3, 7, 8, 9
8 - 1, 3, 4, 6, 7, 8, 9
9 - 1, 2, 3, 7
Vidíme, že v čísla 4, 5, 6 možeme umiestniť iba do buniek č. 1, 3 a 8, čiže z nich môžeme odstrániť všetky ďalšie možnosti (to, že v bunke č. 8 nemáme možnosť umiestniť číslo 5 nám nevadí). Po odstránení spomínaných možností zistíme, že rovnakým postupom môžeme odstrániť možnosti 3 a 7 zo 7 bunky, lebo sa čísla 8 a 9 nachádzajú iba v bunkách 6 a 7 a teda tam iné možnosti niesú. Po konečnej úprave nám vznikne nasledovný výsledok:
1 - 4, 5, 6
2 - 1, 2, 3, 7
3 - 4, 5, 6
4 - 1, 2, 3, 7
5 - 1, 3, 7
6 - 8, 9
7 - 8, 9
8 - 4, 6
9 - 1, 2, 3, 7

■

2.3.4.5 Unique Rectangles Technika riešenia sudoku pozostávajúca z dvoch Locked párov, ktorých členy tvoria obdĺžnik a z toho je odvodený aj názov samotnej techniky riešenia. Keďže každé sudoku má iba jediné správne riešenie, nesmieme za žiadnych

[3] číslo nevyplneného poľa - čísla, ktoré je možné zapísať do daného poľa oddelené čiarkou

okolností pripustiť, aby nám vznikla situácia, ktorá by nám umožnila toto pravidlo porušiť. Z tohto pravidla bolo odvodených niekoľko postupov, ktorých základ tvoria štyri políčka, ktoré sa nachádzajú práve v dvoch riadkoch, dvoch stĺpcoch a dvoch blokoch. Všetky políčka obsahujú dve rovnaké možnosti.

2.3.4.5.1 Typ 1 V prvom type máme v jednom políčku jednu možnosť navyše. Keďže by sme nemohli mať vo všetkých štyroch políčkach iba dve rovnaké možnosti (mohli by sme dosadiť ľubovoľnú z nich a vzniklo by nám správne riešenie a to nieje povolené) a teda ich môžeme z uvedeného políčka obe odstrániť.

2.3.4.5.2 Typ 2 Podobne ako v prvom type, aj tu máme navyše jednu možnosť s tým rozdielom, že ju máme v dvoch políčkach jedného bloku. Takže máme v jednom bloku dve políčka s dvomi a v druhom bloku s tromi možnosťami. V uvedenom prípade vieme, že v niektorom z dvoch políčok obsahujúcich ďalšiu možnosť, sa bude práve táto možnosť nachádzať a teda ju môžeme vylúčiť z ostatných políčok v rovnakom bloku a riadku alebo stĺpci (podľa toho, či sa nachádzajú dané políčka v jednom riadku alebo stĺpci). Môže nastať aj situácia, keď sa tretia možnosť síce nachádza v dvoch políčkach, ale v iných blokoch (samozrejme sa nachádza v rovnakom riadku alebo stĺpci) a teda môžeme analogicky vylúčiť tretiu možnosť v ostatných bunkách daného riadku alebo stĺpca.

2.3.5 Extrémne ťažké

2.3.5.1 Unique Rectangles Ďalšie dva typy sa už svojou zložitosťou zaraďujú medzi najnáročnejšie techniky riešenia sudoku. Oba sú náročnejšie na samotné hľadanie ich výskytu a od svojich predchodcov sa líšia v tom, že dve políčka (v rovnakom riadku alebo stĺpci) neobsahujú rovnakú tretiu možnosť, ale každé obsahuje inú.

2.3.5.1.1 Typ 3 Pri tomto type hľadáme takpovediac locked pár k práve dvom možnostiam, ktoré sa v našich políčkach odlišujú. Keďže vieme, že sa bude v jednom z nich nachádzať prvá alebo v druhom políčku druhá z nich, predstavíme si ich, akoby to bolo práve jedno políčko, ktoré by ich obsahovalo a budeme v rovnakom riadku, stĺpci, prípadne bloku hľadať políčko, ktoré bude mať ako možnosti svojho riešenia práve naše dve odlišné a vytvorí nám k nim locked pár. Po úspešnom nájdení pomyselného locked páru môžeme z ostatných políčok v rovnakom riadku, stĺpci alebo bloku vyradiť tieto dve možnosti.

2.3.5.1.2 Typ 4 Posledný typ sa od svojho predchodcu odlišuje v tom, že k našim možnostiam žiaľ nenájdeme locked pár, ale zistíme, že jedna z pôvodných možností musí byť obsiahnutá v jednom z týchto políčok (nieje ako možnosť už v žiadnom inom políčku rovnakého riadka, stĺpca alebo bloku) a tým pádom sa v nich nebude môcť nachádzať druhá z nich a preto ju vylúčime.

2.3.5.2 XY-Wing Spôsob riešenia XY-Wing spočíva v hľadaní troch políčok, ktoré spolu obsahujú tri možnosti XYZ, v každom z nich sú obsiahnuté dve z nich a každá možnosť práve dvakrát (prvé políčko čísla XY, druhé YZ a tretie XZ). Prvé s druhým a druhé s tretím sa nachádzajú v rovnakom riadku, stĺpci alebo bloku. Po nájdení daného útvaru možeme z priesečníku prvého a tretieho políčka vylúčiť z možností tú možnosť, ktorá je obsiahnutá v oboch uvedených políčkach, teda možnosť X, lebo pokiaľ by napríklad riešením prvého políčka bolo číslo Y, muselo by byť v druhom políčku ako riešenie číslo Z a z toho plynúc by riešením tretieho políčka muselo byť číslo X.

2.3.5.3 XYZ-Wing Podobne ako v XY-Wing, aj tu hľadáme tri políčka s podobnými vlastnosťami a rozdielom v tom, že druhé políčko obsahuje všetky tri možnosti (X, Y aj Z) a prvé s druhým políčkom sa musia nachádzať v rovnakom bloku. Následne môžeme v bloku v danom riadku (prípadne v stĺpci, podľa toho, čo majú spoločné) v ktorom sa nachádza druhé a tretie políčko vylúčiť z možností číslo X.

2.3.5.4 BUG Removal Názov tohoto spôsobu riešenia je odvodený z Bivalue Universal Grave. K tomuto spôsobu sa môžeme vínimočne dopracovať pri riešení mriežky sudoku, kedy nám vznikne situácia, že každé nevyriešené políčko bude obsahovať práve dve možnosti (bivalue) s vínimkou jedného políčka, ktoré obsahuje viac ako dve možnosti (polyvalue). Nakoľko by mriežka obsahujúca v každom nevyriešenom políčku práve dve možnosti bola buď neriešiteľná, alebo by musela mať viac ako jedno riešenie nespĺňala požiadavku na jedinečné riešenie, nájdeme políčko, v ktorom je viac ako dve možnosti a vylúčime z neho možnosti, ktoré sa v danom riadku, stĺpci alebo bloku nachádzajú práve dva krát. Následne, po odstránení spomínaných možností, by sme mali jednoduchým spôsobom doriešiť celé sudoku [2].

2.3.6 a iné

O niektorých z nasledovných spôsobov riešenia by sa dalo povedať, že sú omnoho ťažšie ako spôsoby skôr popísané a pre zaujímavosť spomeniem iba zopár z nich. Samotný popis riešenia danými spôsobmi by bol veľmi náročný, zdĺhavý a pre našu prácu zbytočný. Ich podrobné vysvetlenie sa dá nájsť na internete [7].

- Coloring - vyfarbovanie.
- Pokus a omyl.
- Nishio - trochu vylepšená verzia pokusu a omylu.
- XY Chains.
- Forcing Chains.
- Bowman's Bingo.
- Pattern Overlay Method.

- Empty Rectangles.
- Cell Forcing Chains.
- Sue-De-Coq.
- Death Blossom.
- 3D Medusa.
- Hidden Unique Rectangles.

2.4 Niektoré typy hier

- *Diagonálne* - od klasického typu sa líši iba v tom, že rovnaké pravidlo platí aj pre uhlopriečky.
- *Párne - nepárne* - v tejto variante hry sú podfarbené políčka, do ktorých je možné zapísať iba párne čísla.
- *Magic sudoku* - hra obsahuje podfarbené políčka, do ktorých je možné zapísať iba čísla, ktorých hodnota nieje väčšia ako počet podfarbených políčok v danom bloku.
- *Geometrické* - hracie pole neobsahuje bloky rozmerov NxN, ale N blokov rôznych tvarov (každý blok však pozosáva z N^2 políčok).
- *Killer* - hracie pole obsahuje farebné skupiny políčok, ktorých súčet sa musí rovnať zadanému číslu.
- *Ďalšie* - Samurai, Sum Number place, Domino sudoku, Extra Regions sudoku.

3 Yii

3.1 Čo je to Framework

Framework je v počítačovom svete určitá abstrakcia, kde všeobecná funkcionalita programu môže byť podľa potreby menená užívateľským kódom, čím vzniká špecifická aplikácia. Framework je univerzálna opakovateľne použiteľná platforma určená pre vývoj aplikácií. Obsahuje podporné programy, knižnice, sady nástrojov a rozhraní. Spojenie týchto rôznych komponentov nám umožňuje rýchly vývoj projektov a riešení. Framework má charakteristické črty, ktoré ho odlišujú od bežných knižníc:

- Celkový priebeh aplikácie kontroluje framework a nie používateľ.
- Framework má presne definované správanie.
- Framework môže byť rozšírený užívateľským kódom, aby poskytol špecifickú funkcionalitu (pridávanie rôznych modulov a ovládacích prvkov).
- Kód frameworku by sa nemal meniť, ale iba rozširovať.

3.1.1 Web application framework

Je framework navrhnutý tak, aby podporil vývoj internetových stránok a aplikácií. Jeho úlohou je zmierniť réžiu spojenú s vykonávaním bežných (stále sa opakujúcich) činností pri vývoji aplikácií. Ako príklad, framework zväčša obsahuje knižnice pre prácu s databázami, vzory samotných internetových stránok, kde sa modifikuje už iba obsah, čiže jeho kód je použiteľný opakovane [4].

3.2 História Yii

Yii je dielom vývojára Qiang Xue, ktorý v minulosti vytvoril Prado framework. Roky skúseností a spätnej väzby získanej z tohto projektu (Prado) upevnili potrebu extrémne rýchleho, bezpečného a profesionálneho Frameworku šitého na mieru tak, aby spĺňal očakávania pre vývoj Web 2.0 aplikácií. S vývojom Yii framework začal 1. januára 2008 a už 23. decembra 2008 po takmer roku vývoja bol Yii 1.0 oficiálne uvolnený pre verejnosť. Autor Yii čerpal inšpiráciu z mnohých frameworks a aplikácií ako napríklad Prado framework, Ruby on Rails, jQuery, Symfony, Joomla a ďalších.

Jeho veľmi pôsobivé výkonové parametre, v porovnaní s ostatnými Frameworkami založenými na PHP, okamžite pritiahli pozitívne ohlasy. Jeho popularita a rozšírenosť stúpa rastúcou rýchlosťou.

Od verzie 1.0 do súčasnosti prešlo Yii mnohými inováciami a bolo odstránených mnoho chýb. Na vývoji framework sa začalo podielať viacero priaznivcov Yii, ktorí pomáhajú s vylepšovaním jeho funkcionality a odstraňovaním chýb, ktoré pri vývoji framework vznikli [5].

3.3 Systémové požiadavky

Správny beh Yii nevyžaduje nič náročné, postačuje mu, aby ste mali na počítači nainštalované PHP minimálnej verzie 5.1.0 a webový server. Po samotnej inštalácii Yii si môžete skontrolovať, či máte splnené všetky podmienky na správny beh Yii pomocou testu prostredia, ktoré sa nachádza v podadresári requirements. Pokiaľ sa pri teste prostredia nezobrazilo nič červené, môžeme spokojne začať tvoriť aplikáciu. Yii má vyriešené aj pripojenia na databázy, kde v konfiguračnom súbore stačí nastaviť prístupové údaje. Yii nepotrebuje lokálnu databázu, vie sa pripojiť aj k databáze umiestnenej na vzdialenom servri.

3.4 Základné princípy

3.4.1 Rýchlosť

Väčšina frameworks založených na CoC má nevýhodu v tom, že sú relatívne pomalé a zaťažujú server, na ktorom bežia. Frameworky, ktoré sú rýchle, toho často veľa nezvládnu a tu prichádza Yii, ktorý toho dokáže naozaj veľa a v porovnaní s najpoužívanejšími frameworks je o niečo rýchlejší.

Vďaka generátoru kódu nemusíme v Yii nič písať od úplného začiatku a veľkú časť kódu nám sám vygeneruje, ale o tom si napíšeme neskôr 3.7.

Yii používa tzv. lazy loading, čiže nenačítava všetky prvky a triedy, ale načítava iba to čo potrebuje. Yii využíva samonačítavanie súborov s triedami takže nepotrebujeme používať require [6].

3.4.2 Bezpečnosť

Bežpečnosť prichádza ako štandard Yii. Má v sebe zabudované overovanie vstupných údajov, filtrovanie výstupov, prevenciu pred SQL injection a Cross-site scripting.

3.4.3 Profesionalita

Yii nám pomáha vytvárať čistý a znovupoužiteľný kód, nasleduje vyor MVC, zabezpečuje jasné oddelenie logickej a prezenčnej vrstvy.

3.5 Inštalácia Yii

Najnovšiu verziu Yii si môžeme stiahnuť priamo z webových stránok framework [5]. Po úspešnom stiahnutí si framework rozbalíme do priečinka podľa vlastného výberu (nieje to priečinok samotnej webovej aplikácie). Po rozbalení framework otestujeme, či náš web server spĺňa všetky požiadavky framework pre svoj správny beh (ako bolo spomenuté vyššie) *http://nasserver/nasyiiadresar/requirements.*

Následne vieme pomocou príkazového riadka vytvoriť základnú webovú aplikáciu (samotný príkaz sa trochu líši v závislosti na operačnom systéme, ktorý máme na svojom webovom servri).

3.5.1 Nastavenie napojenia na databázu MySQL

Nastavenie pripojenia je celkom jednoduché, kde stačí urobiť niekoľko jednoduchých vecí v konfiguračnom súbore našej aplikácie. Konfiguračný súbor nájdeme na adrese */protected/config/main.php* . V konfiguračnom súbore si nájdeme nasledovnú časť,

```
// uncomment the following to use a MySQL database
'db'=>array(
    'connectionString' => 'mysql:host=;dbname=',
    'emulatePrepare' => true,
    'username' => '',
    'password' => '',
    'charset' => 'utf8',
),
```

Výpis 1: Konfiguračný súbor aplikácie - časť MySQL

ktorú podľa popisu odkomentujeme a vyplníme názov hosta, databázy, užívateľske meno a heslo (databázové). Týmto máme pripojenie na databázu úspešne ukončené.

3.6 Adresárová štruktúra

3.6.1 Frameworku

- *demos* - ukážky aplikácií,
- *framework* - kód jadra framework - obsahuje ďalšie podadresáre ako napríklad:
  - *gii* - obsahuje modul generátoru kódu pre:
    - Controller,
    - Crud - create, read, update a delete,
    - Form - formulár,
    - Model - dátovy model,
    - Module,
  - *i18n* - jazykové mutácie základných vecí ako názvy mesiacov, dní, jazykov,
  - *messages* - chybové hlásenia rozdelené do podadresárov podľa zvoleného jazyka,
  - *validators* - validátory kontrolujúce správnosť parametrov,
- *requirements* - obsahuje vyššie spomínané overenie splnenia systémových a iných požiadaviek framework [6].

3.6.2 WEBovej aplikácie

- *assets* - súbory používané webovými prehliadačmi,
- *css* - súbory kaskádových štýlov,

- *protected* - kód našej aplikácie,
  - *components* - všeobecné pomocné triedy,
  - *config* - obsahuje konfiguráciu databázy a ďalších komponentov aplikácie,
  - *controllers*,
  - *data* - po vytvorení aplikácie obsahuje testovaciu databázu SQLite,
  - *extensions* - adresár, do ktorého sa kopírujú stiahnuté rozšírenia pre framework,
  - *messages* - sem idú súbory s vlastnými prekladmi,
  - *models* - modely,
  - *runtime* - dočasné dáta ako cache a log,
  - *views*.

3.7 Gii - generátor kódu

3.7.1 Spustenie

Pred tým, ako chceme začať používať generátor kódu, ho musíme povoliť v konfiguračnom súbore */protected/config/main.php* kde je ho potrebné odkomentovať a nastaviť si heslo, ktoré budeme pri jeho používaní potrebovať. Pokiaľ budeme chcieť ku generátoru pristupovať aj z iného miesta ako je *localhost* musíme v konfiguračnom súbore nastaviť aj parameter ipFilters, kde nastavíme IP adresy miest, z ktorých chceme ku generátoru pristupovať, alebo nastavíme parameter na *'ipFilters'=>array('*')* aby sme sa ku generátoru mohli dostať odkiaľkoľvek.

Po úspešnom odkomentovaní Gii, nastavení si hesla a IPfiltra sa ku generátoru dostaneme cez webový prehliadač na nasledovnej adrese *http://nasserver/index.php?r=gii.*

Následne vidíme odkazy na 5 generátorov kódu, ktoré si popíšeme nižšie a ukážeme si aj časti z ich výstupov. Vzhľadom k logickej postupnosti, ktorú je potrebné pri generovaní kódu použiť, ich budem v takom poradí popisovať. Nesmieme zabudnúť na to, že skôr ako budeme generovať kód, musíme mať nastavené pripojenie na databázu a databáza musí obsahovať potrebné tabuľky z ktorých štruktúry sa budú pri generovaní čerpať. Yii vygeneruje aj odkazy medzi jednotlivými tabuľkami.

3.7.1.1 Model Generátor Prvotný generátor, ktorý použijeme je práve Model generátor. Po zadaní názvu tabuľky v databáze a zvolení si názvu triedy (pokiaľ chceme aby sa volala inak ako tabuľka v databáze) nám vygeneruje základné metódy potrebné k práci so samotnou triedou. Pred samotným vygenerovaním si môžeme zvoliť nasledovné možnosti:

- triedu, ktorej bude naša trieda potomkom - štandardne je to trieda CActiveRecord,
- umiestnenie vygenerovaného súboru (prípadne súborov),
- zvolíme si, či sa majú vytvoriť aj vzťahy k iným triedam,

- máme na výber aj možnosť, či sa nám komentár stĺpcov z databázy prevedie do označenia atribútu v našej novej triede.

Po výbere spustíme vytvorenie náhľadu našej novej triedy a následne si ju môžeme nechať vygenerovať. Ako výstup generátoru dostaneme súbor s obdobným kodom.

```
class Nazovnasejtriedy extends CActiveRecord
{
    public function tableName()
    {
        return 'nazovnasejtabulkyzdatabazy';
    }
    public function rules()
    {
        return array(
            array('atribut1, atribut2', 'required'),
            array('atribut1', 'numerical', 'integerOnly'=>true),
            array('atribut2', 'numerical'),
            array('atribut5', 'numerical', 'required', 'on'=>'nasscenar'),
            array('atribut3', 'length', 'max'=>20),
            array('atribut1, atribut2, atribut3, atribut4', 'safe', 'on'=>'search'),
        );
    }
    public function relations()
    {
        return array();
    }
    public function attributeLabels()
    {
        return array(
            'atribut1' => 'atribut1',
            'atribut2' => 'Nejake policko',
            'atribut3' => 'atribut3',
            'atribut4' => 'atribut4',
        );
    }
    public function search()
    {
        $criteria=new CDbCriteria;
        $criteria->compare('atribut1',$this->atribut1);
        $criteria->compare('atribut2',$this->atribut2,true);
        $criteria->compare('atribut3',$this->atribut3);
        $criteria->compare('atribut4',$this->atribut4);
        return new CActiveDataProvider($this, array(
            'criteria'=>$criteria,
        ));
    }
    public static function model($className=__CLASS__)
    {
        return parent::model($className);
    }
}
```

Výpis 2: Vygenerovaný model

3.7.1.1.1 Popis jednotlivých metód

- *tableName* - vráti názov tabuľky v databáze, pri písaní aplikácie ju osobne pravdepodobne nepoužijeme,
- *rules* - veľmi dôležitá metodá vracajúca pole, ktorého obsahom sú pravidlá pre jednotlivé atribúty. Ako už bolo spomenuté, Yii má v sebe dobre prepracované validátory vstupných údajov a práve v tomto poli sa definujú podmienky, ktoré musí každý atribút spĺňať aby mohol byť spracovaný. Atribúty, pre ktoré nieje nastavené žiadne pravidlo budú odignorované. Pre zaujímavosť si vypíšeme niektoré základné pravidlá:
  - *required* - uvedený atribút musí byť vo formulári vyplnený,
  - *length* - udáva pravidlá týkajúce sa dĺžky vstupu, býva doplnené hodnotou napríklad max alebo min, kde sa presne určí minimálna, maximálna alebo iná hodnota, ktorá je prípustná,
  - *numerical* - atribút musí byť číselný,
  - *unique* - atribút musí byť unikátny a jeho hodnota sa nesmie zhodovať s hodnotou atribútu iného prvku, ktorý sa už v našej databáze nachádza.

  Pre každé pravidlo si ešte môžeme zvoliť, ktorého scenáru sa bude týkať (o scenároch si napíšeme neskôr pri tvorbe formulárov) a pre lepšie pochopenie sa pozrime do riadka, ktorý obsahuje atribut5 kde vidíme, že musí byť číselný, musí byť vo formulári vyplnený a ďalšia hodnota *'on'=>'nasscenar'* hovorí, že toto pravidlo platí iba pre scenár s názvom nasscenar.
- *relations* - tu sa nachádzajú prepojenia s inými triedami,
- *attributeLabels* - metóda, ktorá vracia označenia pre jednotlivé atribúty. Ako sme už spomínali vyššie, môžu sa prevziať z koméntárov v databáze,
- *search* - v tejto metóde vieme nastaviť, podľa ktorých atribútov môžeme vyhľadávať.

3.7.1.2 Controller Generátor Ako ďalší v poradí si predstavíme controller generátor, ktorý sám o sebe toho kódu až tak moc nevygeneruje (v porovnaní s ostatnými generátormi).

Podobne ako pri Model generátore, aj tu si môžeme vybrať triedu, ktorej bude naša trieda potomkom, názov controller, šablónu z ktorej sa bude kúd generovať a ešte si napíšeme zoznam akcií, ktoré bude náš controller spravovať (vypíšeme všetky oddelené čiarkami, alebo medzerami). Množstvo vygenerovaného kúdu a počet vygenerovaných súborov závisí od počtu akcii, ktoré zadáme pred generovaním. Prvý vygenerovaný súbor bude controller.

```
class NazovController extends Controller
{
    public function actionIndex()
```

```
    {
        $this->render('index');
    }
}
```

Výpis 3: Vygenerovaný controller

Ako môžeme vidieť, samotný controller nieje moc rozsiahly, pokiaľ máme uvedenú iba jednu akciu. Samotný názov metódy sa skladá z kľúčového slova action spojeného s názvom akcie (v našom prípade je to index) a kvôli prehľadnosti je názov s veľkým prvým písmenom. V tele metódy je odkaz na príslušný view. V prípade, že si navolíme viacero akcií, bude pre každú z nich vygenerovaná obdobná metóda a tým pádom bude aj samotný kód podstatne rozsiahlejší.

Súčasne generátor vygeneruje pre každú akciu aj príslušný view.

```
/* @var $this NazovController */
$this->breadcrumbs=array(
    'Nazov',
);
?>
<h1><?php echo $this->id . '/' . $this->action->id; ?></h1>

<p>
    You may change the content of this page by modifying
    the file <tt><?php echo __FILE__; ?></tt>.
</p>
```

Výpis 4: Vygenerovaný view

Kód každého view je praktický totožný s jediným rozdielom, a to v tzv. mravenčej navigácií, kde je pre každú akciu vygenerovaná prislúchajúca navigácia. V samotnom tele je ako nadpis načítaný nazov controller, ktorý sa stará o príslušný view a za ním názov akcie. Pod nadpisom môžeme vidieť cestu k súboru, v ktorom sa nachádza obsah samotnej stránky (samozrejme cestu nevidíme priamo v kóde, ale ju uvidíme po zavolaní príslušnej akcie vo webovom prehliadači).

3.7.1.3 Form Generátor Form generátor nám vygeneruje formulár vstupných údajov pre nami zvolený dátový model (ako pri predošlých generátoroch je ho potrebné zadať). Ďalším z údajov, ktoré je potrebné pred samotným generovaním zadať je názov view pod akým sa nám bude daný formulár zobrazovať.

Na rozdiel od predošlých generátorov si môžeme vo form generátore zvoliť aj takzvaný scenár, pre ktorý sa nám formulár bude generovať. Scenárom môže byť napríklad pri používateľoch registrácia a prihlásenie, kde pri každom musí formulár obsahovať iné polia z dôvodu potreby iných vstupných údajov. Pri prihlásení nám postačuje prihlasovacie meno a heslo, kdežto pri registrácií môžeme požadovať množstvo ďalších údajov ako napríklad bydlisko, emailovú adresu a podobne. Po navolení potrebných údajov nám generátor vygeneruje nasledovný kód:

```
<?php
/* @var $this NazovnasejtriedyController */
/* @var $model Nazovnasejtriedy */
/* @var $form CActiveForm */
?>
<div class="form">
<?php $form=$this->beginWidget('CActiveForm', array(
    'id'=>'nazovnasejtriedy-nasview-form',
    // Please note: When you enable ajax validation, make sure the corresponding
    // controller action is handling ajax validation correctly.
    // See class documentation of CActiveForm for details on this,
    // you need to use the performAjaxValidation()-method described there.
    'enableAjaxValidation'=>false,
)); ?>
    <p class="note">Fields with <span class="required">*</span> are required.</p>
    <?php echo $form->errorSummary($model); ?>
    <div class="row">
        <?php echo $form->labelEx($model,'atribut1'); ?>
        <?php echo $form->textField($model,'atribut1'); ?>
        <?php echo $form->error($model,'atribut1'); ?>
    </div>
    <div class="row">
        <?php echo $form->labelEx($model,'atribut2'); ?>
        <?php echo $form->textField($model,'atribut2'); ?>
        <?php echo $form->error($model,'atribut2'); ?>
    </div>
    <div class="row">
        <?php echo $form->labelEx($model,'atribut5'); ?>
        <?php echo $form->textField($model,'atribut5'); ?>
        <?php echo $form->error($model,'atribut5'); ?>
    </div>
    <div class="row">
        <?php echo $form->labelEx($model,'atribut3'); ?>
        <?php echo $form->textField($model,'atribut3'); ?>
        <?php echo $form->error($model,'atribut3'); ?>
    </div>
    <div class="row buttons">
        <?php echo CHtml::submitButton('Submit'); ?>
    </div>
<?php $this->endWidget(); ?>
</div><!-- form -->
```

Výpis 5: Vygenerovaný formulár

V najvrchnejšej časti nášho kódu máme zakomentovanú časť obsahujúcu názov controller, triedy, ktorej sa formular tyka a názov ovládacieho prvku formulára (v našom prípade je to CActiveForm). Ďalej nasleduje začiatok samotného formulára spustením ovládacieho prvku, výpis chýb (v prípade nesprávne zadaných vstupov) a následne idú samotné prvky formulára. Ako môžeme vidieť, jednotlivé prvky sú vo formulári zobrazené v takom poradí, v akom boli popísané v pravidlách. Pre každý prvok sú vygenerované tri riadky kódu, kde prvý je popis prvku tzv. label, následne je vypísané pole, do ktorého sa vpisuje vstupný údaj a v treťom riadku sa v prípade nesprávneho vstupu vypíše chybové

hlásenie pre daný prvok. Na konci je umiestnené iba tlačítko pre odoslanie formulára a ukončenie ovládacieho prvku. Vygenerovaný súbor nájdeme v adresári *protected/views* a musíme ho následne presunúť do správneho podadresára.

Okrem samotného formulára bude vygenerovaný kód, ktorý je potrebné doplniť do predtým vytvoreného controller.

```
public function actionNasview()
{
    $model=new Nazovnasejtriedy('nasscenar');
    // uncomment the following code to enable ajax-based validation
    /*
    if(isset($_POST['ajax']) && $_POST['ajax']==='nazovnasejtriedy-nasview-form')
    {
        echo CActiveForm::validate($model);
        Yii::app()->end();
    }
    */
    if(isset($_POST['Nazovnasejtriedy']))
    {
        $model->attributes=$_POST['Nazovnasejtriedy'];
        if($model->validate())
        {
            // form inputs are valid, do something here
            return;
        }
    }
    $this->render('nasview',array('model'=>$model));
}
```

Výpis 6: Vygenerovaný kód pre controller (pri generovaní formulára)

Ako sme si mohli všimnúť už pri formulári a rovnako pri kóde pre controller, v prípade, že chceme používať ajax validáciu, musíme v kóde formulára nastaviť hodnotu performAjaxValidation na true v parametroch pri spúšťaní ovládacieho prvku, odkomentovať a doplniť kód controller. Po spomínanom ajax kóde nasleduje načítanie zadaných vstupných údajov a ich validácia.

Samotný kód obsahuje aj časť, do ktorej sa dostaneme po úspešnom zvalidovaní údajov, kde je potrebné doplniť, čo treba v danom prípade urobiť. Väčšinou tu bude uloženie zadaných údajov do databázy a podobne.

3.7.1.4 CRUD Generátor CRUD (Create, Read, Update, Delete) generátor nám vygeneruje asi najväčšie množstvo čo do množstva kódu, tak do množstva súborov. Na začiatku, ako obvykle, vpíšeme názov triedy, zvolíme si názov controller a môžeme generovať. Pokiaľ sme už použili predchádzajúce generátory a zvolili sme si rovnaký názov controller, uvidíme, že niektoré súbory v našej aplikácií už existujú. Generátor nám ponúka možnosť pozrieť sa, aké zmeny by sa pri nahradení súboru uskutočnia v prípade, že si túto možnosť zvolíme. Riešenie tohto problému má nasledovné možnosti:

- zmeniť názov controller a tým pádom sa bude väčšina súborov generovať do iného adresára,

- pozrieť si novo vygenerované súbory a potrebný kód nakopírovať do už existujúcich súborov,
- ponechať pôvodné súbory, alebo nahradiť iba niektoré z nich.

Logicky by mohla byť prvá z možností asi najideálnejšou, ale pri reálnej tvorbe aplikácie pravdepodobne nebudeme využívať všetky generátory pre každú dátovú triedu a teda nám podobný problém vznikať nebude a v opačnom prípade už budeme presne vedieť čo urobiť a budume využívať nie prvú ale práve druhú z uvedených možností.

Z controller si ukážeme iba kód, ktorý nám nebol vygenerovaný controller generátorom.

```
class Nazovnasejtriedy2Controller extends Controller
{
    public $layout='//layouts/column2';
    public function filters ()
    {
        return array(
            'accessControl', // perform access control for CRUD operations
            'postOnly + delete', // we only allow deletion via POST request
        );
    }
    public function accessRules()
    {
        return array(
            array('allow', // allow all users to perform 'index' and 'view' actions
                'actions'=>array('index','view'),
                'users'=>array('*'),
            ),
            array('allow', // allow authenticated user to perform 'create' and 'update' actions
                'actions'=>array('create','update'),
                'users'=>array('@'),
            ),
            array('allow', // allow admin user to perform 'admin' and 'delete' actions
                'actions'=>array('admin','delete'),
                'users'=>array('admin'),
            ),
            array('deny', // deny all users
                'users'=>array('*'),
            ),
        );
    }
```

Výpis 7: Úryvok z controller (vygenerovaného CRUD generátorom)

Hneď na začiatku máme premennú layout, v ktorej sa nastavuje ako bude vyzerať obsah vygenerovanej stránky. Prednastavená hodnota našej premennej je *//layouts/column1* . Obe varianty nájdeme v adresári *protected/views/layouts*, kde si môžeme vytvoriť aj vlastnú šablónu pre zobrazenie. V našej ukážke sme si nastavili šablónu s dvomi stĺpcami, kde v druhom stĺpci budeme mať malé menu, ktoré sa bude týkať našich CRUD akcií.

Metóda filters vracia zoznam filtrov, ktoré sa budú v našom controller aplikovať. Parameter accessControl nám zapne filter pre kontrolu prístupu do jednotlivých sekcií,

resp. kto bude môct' vykonávat' jednotlivé akcie. Následne v metóde accessRules sú určené pravidlá pre prístup k jednotlivým akciám.

Z ostatných vygenerovaných súborov si vypíšeme iba malé najzaujímavejšie časti.

```
$this->menu=array(
    array('label'=>'List Nazovnasejtriedy', 'url'=>array('index')),
    array('label'=>'Create Nazovnasejtriedy', 'url'=>array('create')),
);
.
.
.
<?php $this->renderPartial('_search',array(
    'model'=>$model,
)); ?>
```

Výpis 8: Časti ostatného kódu (vygenerovaného CRUD generátorom)

Prvá časť kódu *menu* sú položky z menu, ktoré sa nám bude zobrazovať v druhom stĺpci (ako sme si už skôr spomínali). Druhá časť *renderPartial* je obdobou klasického include, kde načítame obsah, ktorý sa opakuje vo viacerých views a je uložený v samostanom súbore [3].

3.7.1.5 Module Generátor

Ako posledný si spomenieme modul generátor, ktorý na rozdiel od predošlých, nieje závislý na žiadnej dátovej štruktúre, a slúži nám na vytvorenie nového modulu, ktorý budeme chciet' používat' nielen v našej aplikácií.

Samotný generátor nám vytvorí tri súbory, z ktorých si ukážeme iba obsah jedného z nich, lebo ostatné dva sme si už popísali pri predchádzajúcich generátoroch.

```
class NasModulModule extends CWebModule
{
    public function init()
    {
        $this->setImport(array(
            'NasModul.models.*',
            'NasModul.components.*',
        ));
    }
    public function beforeControllerAction($controller, $action)
    {
        if(parent::beforeControllerAction($controller, $action))
        {
            return true;
        }
        else
            return false;
    }
}
```

Výpis 9: Vygenerovaný modul

Metóda init sa ako vo väčšine prípadov vykonáva hneď na začiatku práce a teda si sem môžeme dopísat' všetko, čo sa má pred samotným spustením, resp. pri jeho zavádzaní

vykonať a v druhom prípade v metóde beforeControllerAction môžeme napísať, čo sa má vykonať pred spustením akéhokoľvek controller.

Aby sme mohli používať nami vytvorený modul, musíme ho pridať do poľa modules v konfiguračnom súbore našej aplikácie 3.7.1.

4 Návrh

4.1 Špecifikácia zadania

Potrebujeme informačný systém, v ktorom bude možné hrať hru sudoku rôznych obtiažností a bude generovať samotné zadania. Systém budú používať návštevníci stránok, ktorí sú priaznivcami hlavolamu. Užívatelia sa budú môcť registrovať a následne ukladať svoje rozohrané hry a následne sa k nim bude môcť vrátiť. Pred spustením hry si vyberú obtiažnosť a zvolia si, či budú hrať hru na čas, alebo nie. Aplikácia umožní kontrolu správnosti riešenia a ponúkne nápovedu počas hry. Aplikácia bude ukladať všetky potrebné dáta do MySQL databázy umiestnenej na servri.

4.2 Algoritmus na generovanie čísiel

Samotný algoritnus by som rozdelil na dve časti a to:

4.2.1 Vygenerovanie platného zadania

Po určitej úvahe som dospel k dvom spôsobom, ako vygenerovať platné zadanie a ich princíp je opačný. V prvom prípade niečo uberáme a v druhom pridávame. Pri oboch následne kontrolujeme jedinečnosť vzniknutého zadania.

4.2.1.1 Generovanie plnej mriežky Prvým spôsobom je vygenerovanie plnej mriežky (teda vyriešeného zadania), následne z nej odoberať možnosti a priebežne kontrolovať jedinečnosť vzniknutého zadania.

Samotné vygenerovanie vyplneného zadania nevyzerá na prvý pohľad tak zložito, pokiaľ sa o to sami nepokúsime. Jedna z možností, ako dané zadanie spraviť, je postupným vypĺňaním jednotlivých políčok náhodnými číslami v danom rozpätí a pri každom kontrolovať, či tam môžeme naše číslo umiestniť. Pri pokuse o vygenerovanie zadania týmto spôsobom (ručne) sme rýchlo zistili, že to nieje až také jednoduché ako to na prvý pohľad vyzerá. Prvých niekoľko riadkov išlo ako po masle až následne prišiel zlom a zrazu sa do políčka nedalo umiestniť už žiadne číslo tak, aby spĺňalo podmienky. Pre programátora nieje problém naprogramovať algoritmus, ktorý bude vedieť vyrobiť mriežku tak, aby všetky podmienky spĺňala.

Druhým spôsobom, akým sa môžeme dopracovať k plnej mriežke je premiešavaním dobre vyrobenej plnej mriežky výmenou riadkou, stĺpcov, prípadne zámenou čísiel medzi sebou. Týmto spôsobom sa nemusíme obávať, či naša nová mriežka bude spĺňať všetky podmienky. Teraz už len musíme vytvoriť prvú mriežku, z ktorej vyrobíme ďalšie. Najjednoduchším spôsobom, ako si vyrobiť plnú mriežku (aj ručne) je ísť postupne políčko po políčku po jednotlivých riadkoch a vždy do políčka doplniť najmenšie možné čislo. Plnú mriežku by sme teda mali vygenerovanú, ale to je iba prvá časť problému.

Drudou časťou problému je následné odstraňovanie náhodných možností a následná kontrola riešiteľnosti novovzniknutého zadania. Pri kontrole sa už nemusíme sústrediť

na to, či má dané zadanie riešenie, ale či má iba jedno jediné (na čo si samozrejme naprogramujeme kontrolnú funkciu). Koľko čísiel z mriežky daným spôsobom odstránime je len na nás a jediné čo musíme dodržať je to, čo už bolo spomenuté, aby mala býsledná mriežka iba jediné riešenie.

4.2.1.2 Náhodné generovanie zadania Druhým spôsobom, ako dostať dobré zadanie, je náhodným zapĺňaním mriežky (pre urýchlenie sa na začiatku náhodne vyplní 17 políčok, čo je zatiaľ najmenšie množstvo políčok, ktoré musí byť zaplnené, aby mohlo mať zadanie jedinečné riešenie) a následnou kontrolou, či má mriežka práve jedno riešenie.

4.2.2 Zistenie obtiažnosti vygenerovaného zadania

Keďže sme si v našej práci vybrali určovanie obtiažnosti sudoku podľa náročnosti techník 2.2 a zisťovanie náročnosti vygenerovaného zadania by bolo z pohľadu potrebného rozsahu práce príliš náročné. Od vlastného generovania zadaní sme upustili a rozhodli sme sa použiť už vygenerované zadania z internetových stránok.

4.3 Dátový model

4.3.1 Užívatelia

Z dôvodu potreby ukladania rozohraných hier a vzhľadom k tomu, že pri našej práci budeme používať MySQL databázu, budú hry ukladané na našom databázovom servri. Riešitelia musia byť z dôvodu jednoznačnej identifikovateľnosti zaregistrovaní. Každá uložená hra je naviazaná na registrovaného užívateľa. Nespornou výhodou uloženej hry na servri je možnosť pokračovať v nej kedykoľvek a kdekoľvek, kde má užívateľ prístup na internet. Pre naše potreby nám budú postačovať nasledovné údaje o užívateľoch:

- *login* - prihlasovacie meno,
- *heslo* - heslo, ktorým sa bude užívateľ prihlasovať,
- *typ* - typ užívateľa (pre potreby našej aplikácie nebude potrebný, ale v prípade rozširovania sa niekedy v budúcnosti môže zísť),
- *email* - emailová adresa používateľa.

4.3.2 Hry

Vzhľadom k tomu, že nebudeme zadania sami generovať si ich budeme musieť najskôr niekam do databázy uložiť. Z dovodu, aby nevznikol problém veľkého množstva polí v databáze, uloží sa celá mriežka ako textový reťazec s dĺžkou 9x9, čiže 81 znakov do jedného poľa a na mieste prázdnych políčok sa uvedie 0.

- *id* - číslo, ktoré bude jedinečné pre každé zadanie,
- *obtiaznost* - obtiažnosť nášho zadania,

- *rozloženie* - tu sa bude ukladať naše získané zadanie.

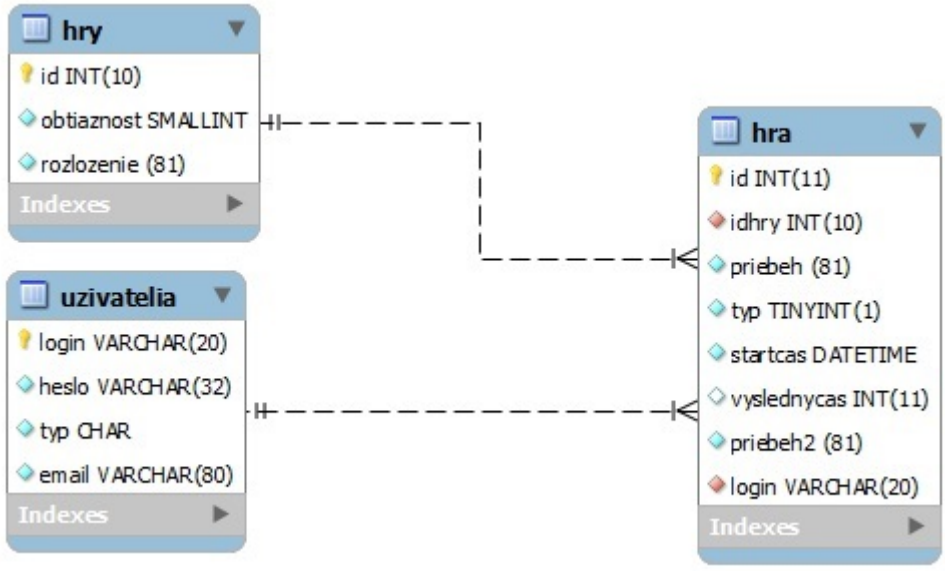
4.3.3 Hra

Ako sme už spomínali, každý riešiteľ si bude môcť svoje riešenie uložiť.

- *id* - číslo, pod ktorým bude naša rozohraná hra uložená,
- *idhry* - číslo zadania, ktoré sa rieši,
- *priebeh* - rovnako ako pri hrách, tu sa bude vo forme reťazca ukladať naše aktuálne riešenie,
- *typ* - rozlíšenie, či sme si vybrali hru na čas, alebo nie,
- *startcas* - čas počiatočného spustenia hry. Na základe startcasu vie užívateľ identifikovať hru, ku ktorej sa chce vrátiť,
- *vyslednycas* - v prípade hry na čas, sa sem bude ukladať ubehnutý čas strávený pri riešení zadania,
- *priebeh2* - pre rozlíšenie políčok získaných z pôvodného zadania od tých, ktoré sme vyplnili mi alebo aplikácia vo forme nápovedy, sme vyrobili reťazec, ktorý bude prídavným k nášmu priebehu s tým rozdielom, že bude obsahovať iné čísla:
  - *0* - zatiaľ nevyplnené políčko,
  - *1* - políčko z pôvodného zadania,
  - *2* - políčko vyplnené našim riešiteľom,
  - *3* - políčko obsadené nápovedou,
- *login* - aby sme vedeli identifikovať, komu rozohraná hra patrí, bude sa sem ukladať prihlasovacie meno riešiteľa.

Dôvody pre ukladanie celého priebeh2 do databázy sú: ukladať celý priebeh2 ale iba počet použitých nápoved a napokon som sa rozhodol pre túto variantu z nasledovných dôvodov:

- aby jednotlivé políčka boli pri vykreslení farebne rozlíšené podľa toho, akého sú pôvodu, je potrebné presne vedieť, ktoré políčko blo kým vyplnené, či už riešiteľom, nápovedou alebo na základe pôvodného zadania,
- že počet použitých nápoved je zistiteľný podľa počtu čísiel 3 v našom spomínanom reťazci.



Obr. 3: ER diagram

4.3.4 ER diagram

4.4 Užívateľské rozhranie

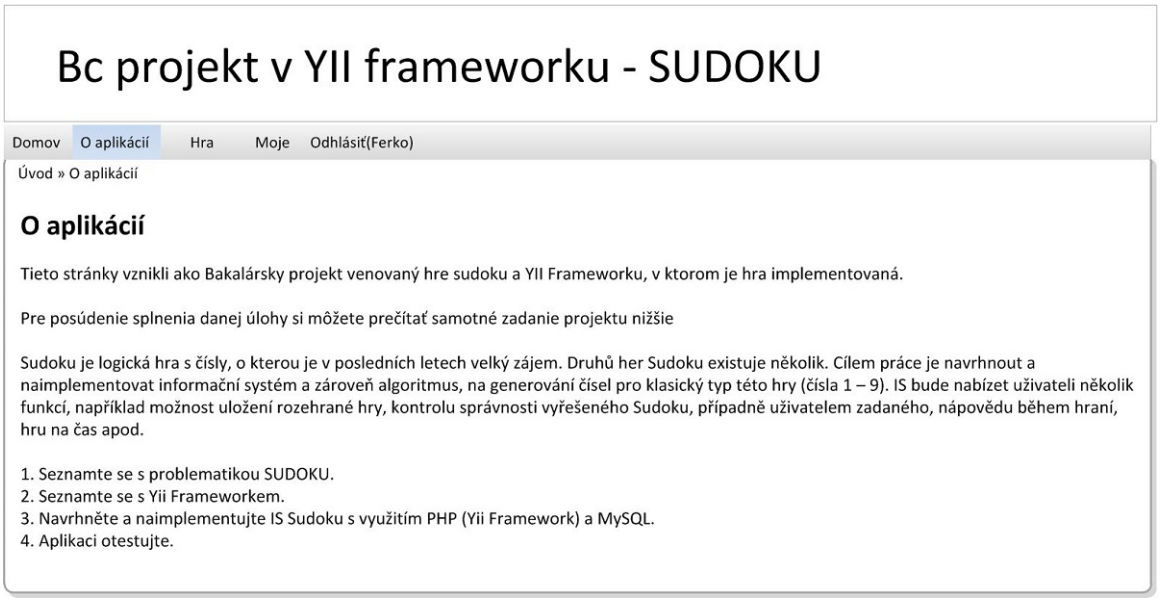
4.4.1 Hlavné menu

V aplikácií budeme využívať preddefinované vodorovné menu. Ako položky zachováme Domov, kde umiestnime uvítanie na našich stránkach. Ako ďalšie v poradí ponecháme položku O aplikácií, kde umiestnime text zadania bakalárskej práce.

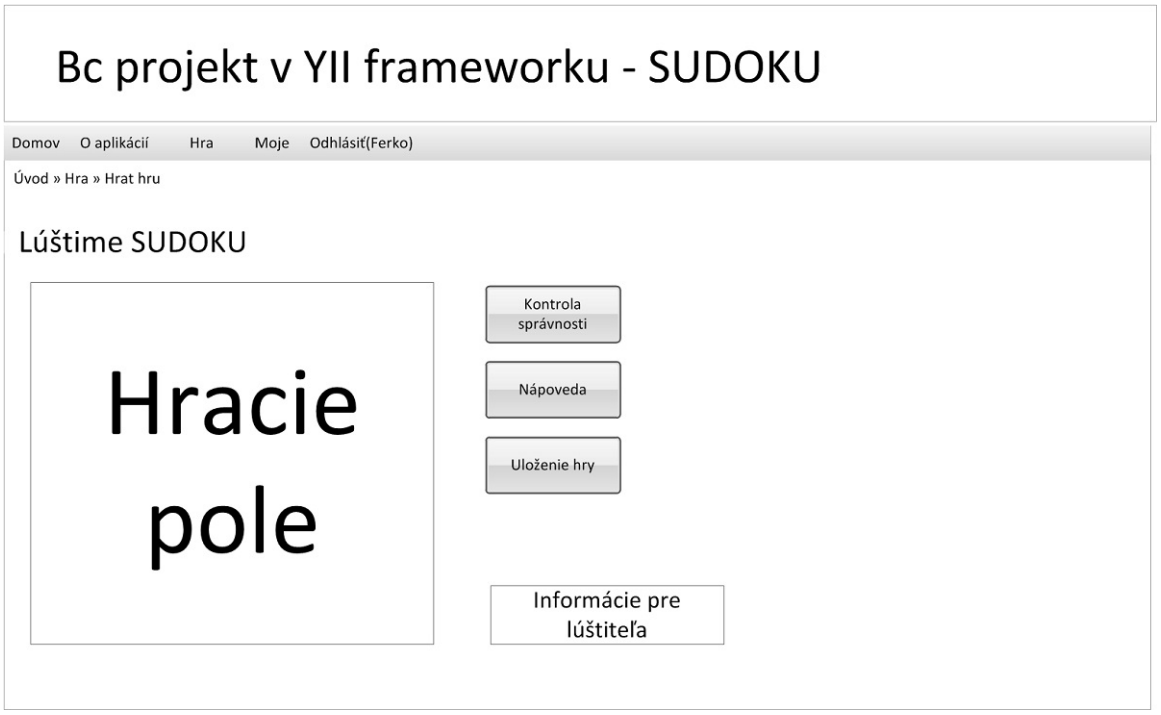
Pre našu aplikáciu budeme však potrebovať ešte niekoľko položiek. Prvá by sa mala týkať samotnej hry, respektíve jej spustenia. Druhá časť je iba pre prihlásených užívateľov a obsahuje ich rozohrané hry. Poslednú položku týkajúcu sa prihlasovania a odhlasovania ponecháme.

4.4.2 Riešenie sudoku

V časti Hra, z navrhnutého menu, bude umiestnený dropdownlist pre výber obtiažnosti a checkbox, ktorý slúži pre zvolenie možnosti hrať hru na čas, alebo nie. Po potvrdení sa nám načíta nahodná hra zvolenej obtiažnosti z databázy a spustí sa.



Obr. 4: Návrh užívateľského rozhrania - hlavné menu



Obr. 5: Návrh užívateľského rozhrania - hracia plocha

5 Implementácia

Po inštalácií apache, MySQL, Yii a úspešnom nastavení napojenia na databázový server môžeme začat'so samotným vývojom apikácie. Vývoj sme si rozdelili do niekoľkých častí, ktoré si popíšeme v poradí v akom vznikali.

5.1 Databáza

Prvým krokom, ktorý bolo potrebné urobit'po inštalácií Yii, je vytvorenie tabuliek v našej databáze, ktoré sme si vopred pripravili na základe navrhnutého dátového modelu v samostatnom súbore *create.sql* .

5.2 Vygenerovanie základných častí aplikácie

Ku každej tabuľke sme si následne pomocou modulu gii 3.7 vygenerovali súbory dátového modelu, príslušné controllers. Pred generovaním formuláru (v našom prípade budeme zatiaľ generovat' iba registračný formulár) sme si do triedy Uzivatelia doplnili atribút *overenie* a nastavili potrebné pravidlá.

```
public function rules()
{
    return array(
        array('login, email', 'required'),
        array('login, email', 'unique'),
        array('email', 'email'),
        array('heslo', 'required', 'on'=> 'Registracia'),
        array('login', 'length', 'max'=>20),
        array('heslo', 'length', 'min'=>8, 'max'=>32),
        array('overenie', 'compare', 'compareAttribute'=>'heslo', 'on'=>'Registracia'),
        array('email', 'length', 'max'=>80),
        array('login, email', 'safe', 'on'=>'search'),
    );
}
```

Výpis 10: Model Uzivatelia - Rules

Ako môžeme vidiet', nastavili sme si pravidlá pre dĺžku prihlasovacieho mena a hesla a aj nutnost' zhody nášho pridaného atribútu *overenie* s heslom pri registrácií. Po nastavení môžeme vygenerovat' registračný formulár.

5.3 Hlavné menu

Položky hlavného menu sme podľa návrhu doplnili do príslušného súboru *protected/views/layouts/main.php* , kde sme okrem samotných položiek nastavili, že časť *Moje* sa bude zobrazovat' iba prihlaseným užívateľom.

```
<div id="mainmenu">
    <?php $this->widget('zii.widgets.CMenu',array(
        'items'=>array(
            array('label'=>'Domov', 'url'=>array('/site/index')),
```

```
            array('label'=>'O aplikacii', 'url'=>array('/site/page', 'view'=>'about')),
            array('label'=>'Hra', 'url'=>array('/hra')),
            array('label'=>'Moje', 'url'=>array('/uzivatelia'), 'visible'=>!Yii::app()->user->
                isGuest),
            array('label'=>'Prihlasit', 'url'=>array('/site/login'), 'visible'=>Yii::app()->user
                ->isGuest),
            array('label'=>'Odhlasit ('. Yii::app()->user->name.')', 'url'=>array('/site/logout'),
                'visible'=>!Yii::app()->user->isGuest)
        ),
    )); ?>
</div><!-- mainmenu -->
```

Výpis 11: Položky hlavného menu

Okrem menu sme v danom súbore ešte nastavili jazyk na slovenčinu.
`Yii::app()->setLanguage("sk");`

5.4 Prihlasovanie užívateľov

Na prihlasovanie a odhlasovanie užívateľov sme použili prihlasovací formulár obsiahnutý v základnej aplikácii. Jediné, čo sme urobili, bola zmena kódu v komponente UserIdentity, kde sme vyriešili načítanie užívateľov z našej databázy. V prípade, že užívateľ ešte nieje zaregistrovaný, na prihlasovacej stránke nájde odkaz na registračný formulár (obsah si nebudeme popisovať, bol vygenerovaný).

```
\\ povodny kod
    $users=array(
        // username => password
        'demo'=>'demo',
        'admin'=>'admin',
    );
\\novy kod
    $uziv=Uzivatelia::model()->findAll();
    foreach($uziv as $a)
    {
        $users[$a->login]= $a->heslo ;
    }
```

Výpis 12: Autentifikácia užívateľov - napojenie na databázu

5.5 Hra

5.5.1 Databáza hier

Nakoľko sme nevytvárali samotný generátor zadaní, bolo nutné získať zadania z iného zdroja. Rozdelenie obtiažnosti sme čerpali zo stránok Brainbashers [2], z ktorých budeme čerpať aj samotné zadania. Spoliehame sa na to, že tieto zadania zodpovedajú príslušným obtiažnostiam.

Aby sme mali dostatok zadaní pre každú obtiažnosť (minimálne niekoľko stoviek), sme sa rozhodli rozmnožiť opísané zadanie jeho modifikáciou výmenou riadkov a pod.

popísanou vyššie 4.2.1.1. K modifikácií sme si vytvorili aplikáciu v c#, kde sme z jedného zadania získali rádovo desaťtisíce nových zadaní (reálne ich vieme vygenerovať skoro tri a pol milióna z jedného zadania, ale v našej aplikácií sme nechali vygenerovať iba malú časť z nich). Uvedenú aplikáciu sme napojili na našu MySQL databázu a vzniknuté zadania do nej rovno uložili.

5.5.2 Voľba hry (menu Hra)

V tejto časti aplikácie sme nemohli použiť vygenerovaný formulár 3.7, ale sme si ho vzhľadom na naše potreby museli sami vyrobiť celé telo formulára a v ukážke bude iba náš vlastný kód.

```
$atributy= array( 1 => 'Velmi lahke',
                  2 => 'Lahke',
                  3 => 'Priemerne',
                  4 => 'Tazke',
                  5 => 'Velmi tazke',
                  6 => 'Extremne tazke',
                  7 => 'Vlastne zadanie',
                );
echo $form->dropDownList($model, 'obtiaznost', $atributy , array('prompt'=>'Vyber obtiaznost') )
    .'<br />';
echo $form->labelEx($model, 'typ');
echo $form->checkBox($model,'typ', array('checked' => true))."<br />";
echo CHtml::submitButton('Hraj vybranu obtiaznost').'<br />';
```

Výpis 13: Výber typu hry

Ako prvé sme si naplnili `$atributy` potrebnými hodnotami a následne sme nechali vypísať náš dropdownlist, checkbox (hra na čas) a následne tlačítko na odoslanie formulára. Po odoslaní formulára sa na základe zvolenej obtiažnosti náhodne vyberie hra z databázy a vykreslí sa.

```
    $gamesa=Hry::model()->find(array(
   'condition' => 'obtiaznost = :o',
   'params' => array( ':o' => $model->obtiaznost ),
   'order' => 'RAND()'
    ));
  $model->prirad($gamesa);
```

Výpis 14: Náhodný výber hry z databázy

Ako sme si mohli všimnúť už pri prihlasovaní užívateľov 12, výber prvkov z databázy je v Yii naozaj jednoduchý, stačí zavolať príslušný model a spustiť vyhľadávanie so zvolenými parametrami. Ako prevencia pred možnými útokmi je kritérium vyhľadávania odovzdané formou parametra a teda nehrozí prienik do systému. V prípade, že si riešiteľ nevybral žiadnu obtiažnosť, bude aj obtiažnosť zvolená náhodne.

5.5.3 Herná plocha

Najdôležitejšou časťou aplikácie je práve miesto, kde prebieha samotné riešenie hlavolamu. Obrazovku sme sa rozhodli kvôli prehľadnosti rozdeliť na dve časti, jednu, v ktorej sa bude nachádzať samotná mriežka (ľavá časť) a druhú časť, kde budú umiestnené tlačítka a informácie ohľadom prebiehajúcej hry viď 4.4.2.

5.5.3.1 Hracia mriežka CSS Hraciu mriežku sme umiestnili do tabuľky 9x9 políčok a aby sme mali políčka súmernej veľkosti, vlastnosti tabuľky sme si nastavili v súboroch kaskádových štýlov.

```
table.sudotable, table.sudotable td, table.sudotable td input
{
  text-align:center;
}
input.chlievik
{
  font-size: 1em;
  border: 0px solid transparent;
}
td.policko3 input.chlievik
{
  color: #448800;
}
td.policko0 input.chlievik, td.policko2 input.chlievik
{
  color: #4444ff;
}
table.sudotable
{
  height: 500px;
  width: 500px;
  margin-right: 20px;
}
table.sudotable td
{
  border-width: 1px;
  border-color: #111111;
  border-style: solid;
  width: 45px;
  height: 45px;
  font-size: 20px;
  margin: 0 0 0 0;
  padding: 0 0 0 0;
}
table.sudotable tr:nth-child(3n) td
{
  border-bottom-width: 3px;
}
table.sudotable tr:nth-child(3n+1) td
{
  border-top-width: 3px;
```

```
}
table.sudotable td:nth-child(3n)
{
  border-right-width: 3px;
}
table.sudotable td:nth-child(3n+1)
{
  border-left-width: 3px;
}
.policko1
{
 font-style: bolder;
 color: #000000;
}
.napoveda
{
 font-style: bolder;
 color: #00ff00;
}
```

Výpis 15: Kaskádové štýly pre mriežku sudoku

Okrem samotnej veľkosti políčok sme si nastavili aj hrubšie čiary na oddelenie blokov, farby písma podľa toho, kto danú hodnotu do mriežky doplnil (čierna zo zadania, zelená doplnená nápovedou, modrá vyplnená riešiteľom). Nakoniec sme si nastavili rovnakú veľkosť tlačítok.

```
input[type="submit"]
{
  height: 3em;
  width:  13em;
  margin: 0px 10px 20px 0px;
}
```

Výpis 16: Veľkosť tlačítok - CSS

5.5.3.2 Vykreslovanie mriežky Na vykreslenie mriežky si voláme funkciu v našej triede, ktorá sa nám postará o vykreslenie celého hlavolamu. Samotný kód je vcelku jednoduchý, ide o prechod dvojrozmerným poľom a prepisovaním ho do tabuľky, pričom každému políčku definujeme vlastnosť podľa jeho typu (kto ho vyplnil).

```
echo "<table class='sudotable'>";
  for ($x=0;$x<9;$x++)
  {
    echo "\n<tr>";
    for ($y=0;$y<9;$y++)
    {
      echo "<td class=\"policko".$this->mriezka2[$x][$y]."\">";
      if ($this->mriezka[$x][$y]!=0)
      {
        switch ($this->mriezka2[$x][$y])
        {
```

```
                case 1 :
                case 3 : echo $form->textField($this,"a$x$y",array('value'=>$this->mriezka[
                      $x][$y], 'size'=>'1', 'class'=>'chlievik' ,  'maxlength'=>'1', 'readonly'=>true,
                       'onkeyup'=>'Change(this.id,event)'))."\n";  break;
                case 2 : echo $form->textField($this,"a$x$y",array('value'=>$this->mriezka[
                      $x][$y], 'size'=>'1', 'class'=>'chlievik' ,  'maxlength'=>'1', 'onkeyup'=>'
                      Change(this.id,event)'))."\n";  break;
                default: echo $form->textField($this,"a$x$y",array('value'=>$this->mriezka[
                      $x][$y], 'size'=>'1', 'class'=>'chlievik' ,  'maxlength'=>'1', 'onkeyup'=>'
                      Change(this.id,event)'))."\n";  break;
              }
            }
            else  echo $form->textField($this,"a$x$y",array('size'=>'1', 'class'=>'chlievik' ,  '
                 maxlength'=>'1', 'onkeyup'=>'Change(this.id,event)'))."\n";
            echo "</td>\n";
          }
          echo "</tr>";
        }
      echo "</table>";
```

Výpis 17: Vykreslenie mriežky

Na jednu vec by sme predsalen radi poukázali. Ako si môžeme všimnúť, každé políčko obsahuje udalosť `'onkeyup'=>'Change(this.id,event)'` volajúcu funkciu, ktorú sme pre zlepšenie uživateľského komfortu doprogramovali v JavaScripte. Ide o možnosť používať na pohyb v našom hlavolame šípky. Možno sa to zdá byť zbytočné, ale používať neustále myš je nepohodlné, nehovoriac o použití tabulátoru.

```
<script type="text/javascript">
var rowscells = 8;
function ChangeCurrentCell(x,y) {
    var idcko="Hra_a";idcko=idcko.concat(x,y);
    var z=document.getElementById(idcko);
    z.focus();
 }
function Change(el, e){
var posx = el.slice(5,6);
var posy = el.slice(6,7);
    if (e.keyCode == 37) {
        if (posy==0) {posy=rowscells+1;}
       posy--; ChangeCurrentCell(posx,posy);
       return false;
    }
  .
  .
    if (e.keyCode == 40) {
        if (posx==rowscells) {posx=-1; }
       posx++; ChangeCurrentCell(posx,posy);
       return false;
    }
};    </script>
```

Výpis 18: Javascript pohyb pomocou šípiek

5.5.3.3 Tlačítka a informácie Vypísanie tlačítok a informácií pre riešiteľa nieje ničím výnimočné a popíšeme si iba meranie času. Pri načítaní stránky sa ako hodnota jedného schovaného parametra odosle na stranu riešiteľa čas, ktorý strávil riešením pred načítaním aktuálneho okna. Z daného parametru si údaj načíta spustený JavaScript a pokračuje v počítaní ubehnutého času, pričom hodnotu pravidelne ukladá do formulára, aby bol v čase jeho odoslania aktuálny.

6 Záver

Cieľom práce bolo navrhnúť a implementovať informačný systém a algoritmus na generovanie čísiel pre klasický typ hry sudoku. Informačný systém mal užívateľovi ponúknuť niekoľko funkcií ako napríklad možnosť uloženia rozohranej hry, nápovedu počas hrania, hru na čas, kontrolu správnosti vyriešeného sudoku, prípadne užívateľom zadaného. Prácu bolo potrebné implementovať v jazyku PHP s použitím Yii framework a MySQL databázy.

V práci sme sa v prvom rade zamerali na samotnú problematiku hry sudoku, jej históriu, náročnosť riešenia z pohľadu riešiteľských techník. Popísali sme si mnoho spôsobov, ktoré používajú roznorodí riešitelia od úplných amatérov až po profesionálov zúčastňujúcich sa svetových súťaží v riešení hlavolamu sudoku. Pre ukážku sme si spomenuli množstvo rôznych modifikácií klasickej hry pre zpestrenie samotného riešenia a obmeny techník, ktoré je potrebné pri modifikáciach použiť. Niektoré modifikácie nemusia vďaka pridaným podmienkam obsahovať minimálne 17 vyplnených čísiel (17 je doteraz známe ako najmenšie možné množstvo vyplnených políčok mriežky sudoku tak, aby bola splnená podmienka jedinečnosti riešenia klasickej varianty tejto hry) a niektoré nemusia obsahovať dokonča žiadne.

V druhom rade sme si predstavili zatiaľ nie moc rozšírený PHP framework Yes it is! v skratke Yii. Ide o pomerne mladý framework, ktorý sa prediera vo svete frameworks a v mnohých ohľadoch predbieha i tie najpoužívanejšie. Spomenuli sme si jeho vznik a inšpiráciu jeho zakladateľa Qiang Xue (autora Prado framework), ktorý sa pri samotnom vývoji snažil čerpať to najlepšie z už existujúcich frameworks. Spomenuli sme si systémovú flexibilitu a základné princípy. Vysvetlili jednoduchosť tvorby moderných vebových aplikácií s použitím generátoru kódu vrátane popisu samotného generovania.

Nakoniec sme navrhli webovú aplikáciu pre hru sudoku a implementovali ju s použitím moderného Yii framework. S použitím generátoru kódu sme sami napísali minimum vlastného kódu. Vďaka prepracovanosti zabudovaného riešenia napojenia na MySQL databázu sme sa nemuseli starať o samotné čítanie ani zapisovanie do nej, nehovoriac o tom, že práca s dátami je zabezpečená pred SQL injection a podobne. Validátory vstupných údajov sa nám postarali o to, že žiaden nesprávny vstup nebol spracovaný. V prípade chýbajúcich alebo neplatných údajov nás Yii na dané skutočnosti upozornil.

V budúcnosti by sa mohla aplikácia rozrásť o možnosť riešiť aj iné ako klasické varianty hry sudoku, možnosť súperiť v riešení sudoku formou spoločného riešenia na rôznych miestach v rovnakom čase, kde samotná hra by bola zahájena všade v rovnakom čase (súťaž by sa spúšťala z vopred určenej hlavnej stanice) a podobne. Mohli by vzniknúť varianty hry s časovou penalizáciou za použitie nápoved, prípadne s ich vylúčením.

7 Literatúra

[1] *Sudoku.* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-02-09]. Dostupné z: http://sk.wikipedia.org/wiki/Sudoku

[2] *BrainBashers: Puzzles and Brain Teasers* [online]. 2014 [cit. 2014-04-02]. Dostupné z: http://www.brainbashers.com/

[3] MAKAROV, Aleksander a Qiang XUE. *Yii 1.1 application development cookbook: over 80 recipes to help you master using the Yii PHP framework.* Birmingham, U.K.: Packt Pub. Open Source, 2011, iv, 371 p. ISBN 978-1-849515-48-1.

[4] *Framework* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Framework

[5] YII SOFTWARE LLC. *Yii PHP Framework: Best for Web 2.0 Development* [online]. 2014 [cit. 2014-04-10]. Dostupné z: http://www.yiiframework.com/

[6] KOPRDA, Matúš *Ako sa stať PHP ninja s Yii Frameworkom.* In: Zajtra.sk [online]. 2010 [cit. 2014-03-18]. Dostupné z: http://www.zajtra.sk/programovanie/78/ako-sa-stat-php-ninja-s-yii-frameworkom-1

[7] STUART, Andrew. SYNDICATED PUZZLES INC. *SudokuWiki.org* [online]. 2013 [cit. 2014-04-08]. Dostupné z: Sudokuwiki.org