Základní informace

Jméno projektu	ReCodEx - Code Examiner
Zkratka	*ReCodEx*
Vedoucí	*Martin Kruliš <krulis@ksi.mff.cuni.cz>*
Konzultanti	*Martin Mareš <mj@ucw.cz>*
Anotace	*Webová aplikace sloužící vytváření programátorských úloh a následně testování a hodnocení odevzdaných řešení pomocí předpřipravených sad vstupů a vzorových výstupů. Cílem je kompletní reimplementace předchozí verze (CodEx), která slouží především k podpoře výuky programování (domácí úkoly) na MFF a k online programátorským soutěžím (KSP, Kasiopea).*

Motivace

Projekt navazuje na již existující aplikaci CodEx, která byla vyvinuta na MFF (jako SW projekt, jehož tehdejším řešitelem byl vedoucí tohoto projektu) v roce 2007/2008. I přesto že se současná aplikace hojně využívá, trpí řadou nedostatků, které původní projekt nedokázal podchytit. Na svou dobu se jednalo o velmi inovativní systém a řada skutečností a požadovaných změn vyplynula na povrch až po několika letech používání.

Druhým důvodem pro vývoj kompletně nového řešení je fakt, že CodEx je technologicky zastaralý. Současný systém pro vyhodnocování úloh bude nahrazen kompletně novým sandboxem, který zlepší bezpečnost, udržitelnost i rozšiřitelnost testovaných úloh a podporovaných programátorských platforem. Frontend bude nahrazen moderním webovým řešením založeným na soudobých trendech a standardu HTML5.

Vzhledem k požadavkům uživatelů a stavu současného CodExu očekáváme, že nový systém ReCodEx nahradí stávající řešení v plném rozsahu. Stávající aplikace poslouží pouze jako zdroj pro analýzu problému a množství přebraného kódu bude zcela jistě nižší než 10%, přičemž jeho většina bude sloužit zejména k účelům migrace dat ze starého systému.

Popis projektu

Uživatelské rozhraní je vhodné chápat v kontextu tří typů uživatelů (resp. uživatelských rolí):

- Student – uživatel, který řeší úlohy a vidí svoje výsledky
- Vyučující – uživatel, který zadává předpřipravené úlohy studentům a sleduje jejich výsledky
- Autor – uživatel, který vytváří nové úlohy

Student po přihlášení vidí skupiny, jichž je členem. Tyto skupiny odpovídají studijním skupinám (např. rozvrhovým lístkům) a v rámci skupiny dostává student zadané úlohy. Student řeší úlohy tak, že nahraje zdrojové kódy svého řešení do systému a systém jej automaticky vyhodnotí a oboduje. Student může odevzdat řešení opakovaně (zejména po opravě svých chyb), přičemž systém bere nejlepší odevzdané řešení k dané úloze jako směrodatné pro účely bodování.

Vyučující spravuje jednu nebo více skupin, které typicky odpovídají cvičením nebo přednáškám, které vyučuje. V rámci skupin může z připravené databáze zadávat úlohy. Při zadání se specifikují

upřesňující parametry (např. deadline, max. počet bodů, způsob hodnoceni, povolené jazyky řešení, …), které omezují způsob řešení úlohy. Vyučující má rovněž přehled o získaných bodech studentů a o odevzdaných řešeních. Každé řešení si přitom může zobrazit nebo stáhnout, případně ručně upravit jeho bodové hodnocení.

Autor úloh vytváří a konfiguruje nové úlohy. Úloha má tři podstatné součásti – zadání, konfiguraci a testovací data. Zadání je zobrazeno vyučujícím a řešitelům (v HTML) a jeho účel je čistě informativní. Konfigurace řídí vyhodnocování úlohy – jak se zdrojový kód přeloží, jak se bude testovat, časové a paměťové limity jednotlivých testů atd. Testovací data jsou pak použita pro vyhodnocení funkčnosti a případně i efektivity řešení.

Typický příklad průběhu vyhodnocování vypadá následovně. Odevzdaný zdrojový kód je na cílové platformě přeložen pomocí kompilátoru, který odpovídá jazyku řešení (např. gcc pro řešení v jazyce C). Následně se řešení spustí v sandboxu pro každou vstupní testovací sadu. Sandbox kontroluje, aby řešení nemohlo provádět operace nesouvisející s úlohou (pracovat s jinými než danými soubory, komunikovat po síti, …) a hlídá spotřebovaný čas a paměť. Výstup řešení je porovnán se vzorovým výstupem pomocí definované hodnotící aplikace (u triviálních dat např. diff).

Samozřejmostí je pak administrátorské rozhraní pro správu uživatelů, skupin, databáze úloh a ostatních částí aplikace. Systém bude rovněž připraven na import dat ze souvisejících zdrojů používaných na MFF-UK (zejména SIS) a autentizaci proti existujícím bezpečnostním entitám (CAS UK). Jako doplněk bude implementován notifikační systém, který bude rozesílat důležité zprávy uživatelům pomocí e-mailu.

Jednotlivé detaily funkcí zde nejsou z důvodu omezeného prostoru rozebrány a vyplynou z analýzy požadavků, které již byly posbírány od vyučujících používajících současný CodEx.

Platforma, technologie

Aplikace je rozdělena do tří logických celků, mezi nimiž je velmi striktní rozhraní:

- Uživatelský frontend implementovaný jako HTML5 aplikace běžící čistě v prohlížeči. Použit bude vhodný framework (např. Angular JS) a prezentační logika bude implementována v Javascriptu.
- Jádro ReCodExu obsahující aplikační logiku implementovanou v PHP, databázi a souborové úložiště. Tato část aplikace nabízí jednoduché API nad protokolem HTTP (bude zvoleno buď klasické REST API nebo vlastní AJAXové API), které je využíváno frontendem, ale částečně i backendem. Dále jádro předává backendu pokyny pro vyhodnocování úloh.
- Vyhodnocovací backend bude tvořit relativně nezávislý distribuovaný systém postavený na message queues frameworku (pravděpodobně Zero MQ). Systém bude mít architekturu master-worker, kde master poskytuje API pro vyhodnocování, se kterým komunikuje jádro. Workers zajišťují vyhodnocení úloh, přičemž dodatečnou komunikaci (stažení testovacích dat, odeslání výsledných logů o vyhodnocení, …) řeší přímo s jádrem skrz HTTP.

Díky tomuto dělení bude možné používat společný vyhodnocovací backend z více instalací aplikace nebo jej dokonce použít samostatně se separátním frontendem v podobě aplikace spouštěné z příkazové řádky. Obdobně klientské rozhraní bude možné zcela zaměnit a používat tak ReCodEx

pro značně odlišné účely, jejichž společným jmenovatelem je pouze vyhodnocování odevzdaných zdrojových kódů (výuka na MFF vs. programátorské online soutěže), nebo bude možné použít funkce jádra aplikace přímo pomocí skriptů a z již existujících webových stránek (např. KSP).

Vyhodnocování řešení bude implementováno přímo v backendu, který bude používat vlastní konfiguraci pro řízení vyhodnocování. Tato konfigurace je reprezentována jako acyklický závislostní graf kroků, kde jednotlivé kroky můžou být buď provedení interní operace (zkopírování souboru, komunikace s jádrem, …) nebo spuštění externí aplikace uvnitř sandboxu (kompilace, testování zkompilovaného řešení, …). Jako sandbox na platformě Linux bude použit isolate, jehož spoluautorem je konzultant Martin Mareš. Na platformě Windows momentálně neexistuje podobný software, ale bude možné zde např. využít specifické platformy, které mají vlastní bezpečnostní mechanismy (.NET a Java).

Datové úložiště využívané jádrem bude rozděleno do tří částí. Základem je databáze (MySQL), která bude uchovávat všechna data vyjma souborů testů a souborů s odevzdanými zdrojovými kódy. Odevzdaná řešení, vygenerované konfigurace pro vyhodnocovací backend, a záznamy (logy) z vyhodnocování budou uchovány jako soubory v adresářové struktuře. Soubory testů budou uloženy ve speciálním hash-store, kde každý soubor je uložen pod názvem odpovídajícím SHA1 hash otisku jeho obsahu. Přiřazení souborů k jednotlivým úlohám bude uloženo v databázi. Tento mechanismus usnadní sdílení souborů při kopírování (klonování) úloh a také jejich cachování na straně klienta.

Odhad náročnosti

Očekávaný počet řešitelů: 5
Doba projektu: 9 měsíců

Předběžný plán prací

měsíc	práce
1.	• návrh formátu konfigurace pro workery, implementace prototypu workera (2 lidé) • nasazení MQ systému, implementace mastera (1 člověk) • analýza posbíraných uživatelských požadavků (2 lidé)
2.	• práce na implementaci backendu (2 lidé) • návrh a příprava DB schématu, příprava testovacích dat (1 člověk) • návrh a implementace základů jádra – napojení na DB, příprava API, logování (2 lidé)
3.	• dokončení implementace backendu + testování (2 lidé) • příprava interního datového modelu a databázové abstrakce (1 člověk) • návrh a implementace souborového úložiště a jeho API (2 lidé)
4.	• dokončení souborového úložiště + testování s backendem (2 lidé) • návrh autorizačního a autentizačního modelu a jeho implementace (1 člověk) • zahájení prací na frontendu (2 lidé)
5. – 6.	• implementace frontendu (2 lidé) • implementace API a funkcí jádra (2 lidé) • návrh a implementace UI pro editaci konfigurací úloh (1 člověk)

7.	• finalizace plánovaných uživatelských features ve frontendu a jádře (4 lidé) • příprava na import dat (úloh) ze současného CodExu (1 člověk)
8. – 9.	• testování ve spolupráci s vyučujícími na MFF, oprava chyb • dokumentace, finalizace, import dat • bude-li možné, realizace napojení na SIS a CAS

Vymezení projektu

Projekt je zaměřen na následující oblasti (zaškrtněte vyhovující):

Diskrétní modely a algoritmy	
	diskrétní matematika a algoritmy
	geometrie a matematické struktury v informatice
	optimalizace
Teoretická informatika	
	Teoretická informatika
Softwarové a datové inženýrství	
x	softwarové inženýrství
x	vývoj software
x	webové inženýrství
x	databázové systémy
	analýza a zpracování rozsáhlých dat
Softwarové systémy	
x	systémové programování
	spolehlivé systémy
	výkonné systémy
Matematická lingvistika	
	počítačová a formální lingvistika
	statistické metody a strojové učení v počítačové lingvistice
Umělá inteligence	
	inteligentní agenti
	strojové učení
	robotika
Počítačová grafika a vývoj počítačových her	
	počítačová grafika
	vývoj počítačových her