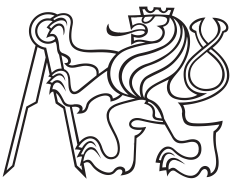
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Realtime prezentace soutěžních výsledků a tweetů v systému MyICPC

Aplikace modulární architektury pro Scoreboard a PhotoKiosk

Ondřej Musil

Září 2017
Vedoucí práce: Ing. Tomáš Černý, MSc., Ph.D.

Poděkování / Prohlášení

Chtěl bych poděkovat především vedoucímu Ing. Tomáši Černému, MSc. za zajímavé téma a aktivní vedení bakalářské práce. Dále bych chtěl poděkovat celému týmu vývojářů MyICPC za spolupráci při vývoji. V neposlední řadě také rodině za jejich podporu.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 21. 9. 2017

.....

Abstrakt / Abstract

Systém MyICPC byl navržen pro podporu informovanosti a zájmu účastníků a diváků během programovacích soutěží ACM-ICPC. Jeho hlavním úkolem je poskytovat soutěžní výsledky v reálném čase. Tento úkol s sebou přináší řadu úskalí. Jako největší problém se ukázala nutnost zpracovávat velké množství dat s výsledky a zároveň tato data prezentovat velkému počtu klientů připojených v průběhu soutěže. Proto bylo navrženo rozdělení systému na několik modulů, aby se tyto činnosti oddělily a neprobíhaly na stejných serverech. Cílem této práce je navrhnout modul DataProvider na ukládání soutěžních dat do databáze, navrnout prezentační logiku pro komponenty Scoreboard a PhotoKiosk, propojit jednotlivé moduly a otestovat jejich činnost pomocí simulace průběhu soutěže.

Klíčová slova: MyICPC; ACM; programovací soutěž; modulární architektura; zpracování dat; Scoreboard; PhotoKiosk; real-time prezentace.

System MyICPC was designed to support awareness and interest of participants and spectators during programming contests ACM-ICPC. Main task of the system is to provide contest results in real time. This task brings many challenges, and the need to process large amounts of data with results and present them to a large number of clients at the same time turned out to be the hardest one. Therefore it was proposed to divide the system into several modules to seperate these tasks, so they don't have to run on the same servers. Main goal of this thesis is to design DataProvider module for storing contest data into database, desing presentation logic for components Scoreboard and PhotoKiosk, link up the individual modules and test their functionality by simulating the course of a programming contest.

Keywords: MyICPC; ACM; programming contst; modular architecture; data processing; Scoreboard; PhotoKiosk; real-time presentation.

Title translation: Realtime presentation of contest results and tweets in MyICPC system (Application of modular architecture on Scoreboard and PhotoKiosk)

Obsah /

Obrázky

Kapitola 1
Úvod

ACM-ICPC jsou týmové programovací soutěže s několika úrovněmi [1]. Týmy programátorů z různých světových univerzit soutěží v regionálních kolech a ty nejlepší z nich se potom každoročně potkávají na světovém finále (ICPC World Finals). Vzhledem k zapojení mnoha velkých světových univerzit se jedná o prestižní události se širokým zájmem veřejnosti.

Systém MyICPC vznikl před 3 roky za účelem zlepšení informovanosti účastníků a diváků soutěží ICPC [2]. Jeho hlavním úkolem je shromažďovat data o těchto soutěžích z různých zdrojů a poskytovat je uživatelům v reálném čase v uživatelsky přizpůsobivém webovém rozhraní. Dále si klade za cíl zlepšit zážitek účastníků pomocí podpory různých vedlejších aktivit. Od doby vzniku je systém pravidelně používán během ICPC soutěží, zejména na světových finále (ICPC World Finals). Za tu dobu si získal velkou popularitu mezi účastníky a návštěvníky těchto soutěží.

1.1 Problémy systému MyICPC

Velká popularita systému sebou nese vysoký počet současně přistupujících uživatelů, zejména během konání samotné soutěže. Kvůli tomu jsou kladeny vysoké nároky na efektivitu a škálovatelnost celého systému. Největší problémy způsobovala vysoká zátěž u modulu s aktuální tabulkou výsledků (Scoreboard). Dalším úskalím je rozsáhlost celého systému. Systém je složen z řady modulů, pracuje s velkým množstvím dat a využívá řadu technologií a frameworků, což přináší vysoké nároky na údržbu a aktualizaci systému. Některé z použitých technologií a přístupů jsou postupem času překonávány a je proto potřeba je aktualizovat nebo nahradit novějšími.

1.2 Cíle práce

Hlavními cíli této práce jsou:

- Vyřešit problém se škálovatelností modulu Scoreboard: Stránka se Scoreboardem by měla zvládat přístup řádově tisíců uživatelů bez znatelného zpomalení. K řešení tohoto problému bylo navrženo přesunutí některých funkcí systému MyICPC do samostatných modulů. Tento přístup by měl vést ke snížení zátěže na servery s webovou aplikací. V rámci této práce byl vyvíjen modul DataProvider.
- Vyhovět požadavku na použití oficiálního zdroje dat v modulu Scoreboard: Organizátor soutěže požaduje, aby tabulka s výsledky soutěže byla založena na oficiálním datovém zdroji JSON Scoreboard. Dosud byla data pro modul Scoreboard generována systémem MyICPC z dat o výsledcích hodnocení odevzdaných programovacích úloh, získaných ze zdroje Event Feed.
- Navrhnout modernější uživatelské rozhraní pro komponentu Scoreboard. Tato komponenta byla realizována zejména pomocí frameworků Java Server Pages a AngularJS. Nově navržená komponenta implementovaná ve frameworku ReactJS by měla

přinést zjednodušení vývoje prezentační vrstvy a také klást větší důraz na přehlednost tabulky s výsledky, kterou zobrazuje.

- Navrhnout komponentu PhotoKiosk pro prezentaci fotografií z průběhu soutěže. Komponenta by měla vycházet ze stávající komponenty Kiosk, navržené pro prezentaci příspěvků ze sociálních sítí. Komponenta Kiosk byla letošním roce přepracována pomocí frameworku ReactJS, tak aby poskytovala atraktivnější prezentaci příspěvků zejména na velkoplošných obrazovkách.
- Integrovat navržené moduly do systému MyICPC a otestovat jejich funkčnost pomocí simulace průběhu soutěže. Testování se týká zejména komponenty Scoreboard, která pro svoji správnou činnost nově vyžaduje spolupráci několika modulů.

Kapitola 2
Analýza stávajícího systému a architektury

Systém MyICPC byl vyvíjen v rámci diplomové práce na ČVUT [2] a jeho stávající verze je využívána v průběhu velkých programovacích soutěží, včetně ACM-ICPC World Finals. Zdrojové kódy systému se nacházejí v repozitáři BitBucket. [3] V této kapitole bude stručně popsána architekta systému a budou zde zmíněny nejdůležitější technologie, na kterých je systém postaven. Podrobnější popis architektury systému lze potom najít v odkazované práci.

2.1 Architektura systému

MyICPC využívá standardní vícevrstvou architekturu, založenou na frameworku Spring, využívající databázi PostgreSQL a Java Server Pages pro webovou prezentaci. Systém je navržen pro provoz v clusteru z důvodu lepší škálovatelnosti. Pro synchronizaci instancí systému je použit framework JMS a systém Redis. Hlavní část systému (MyICPC Webapp) je rozdělena do několika Maven modulů oddělujících jednotlivé vrstvy systému, jak ukazuje diagram 2.1.

2.2 Přehled použitých technologií

MyICPC využívá celou řadu technologií. Zde si uvedeme ty nejdůležitější.

2.2.1 Maven

Maven je nástroj pro správu softwarových projektů, zejména těch vyvíjených v jazyce Java. [4] Umožňuje snadné sestavování aplikací využívajících externí knihovny díky automatické správě závislostí. Dále umožňuje členění aplikací na jednotlivé moduly. MyICPC této vlastnosti využívá a díky tomu bylo možné oddělit jednotlivé vrstvy do samostatných modulů. Základní struktura je zobrazena na diagramu 2.1.

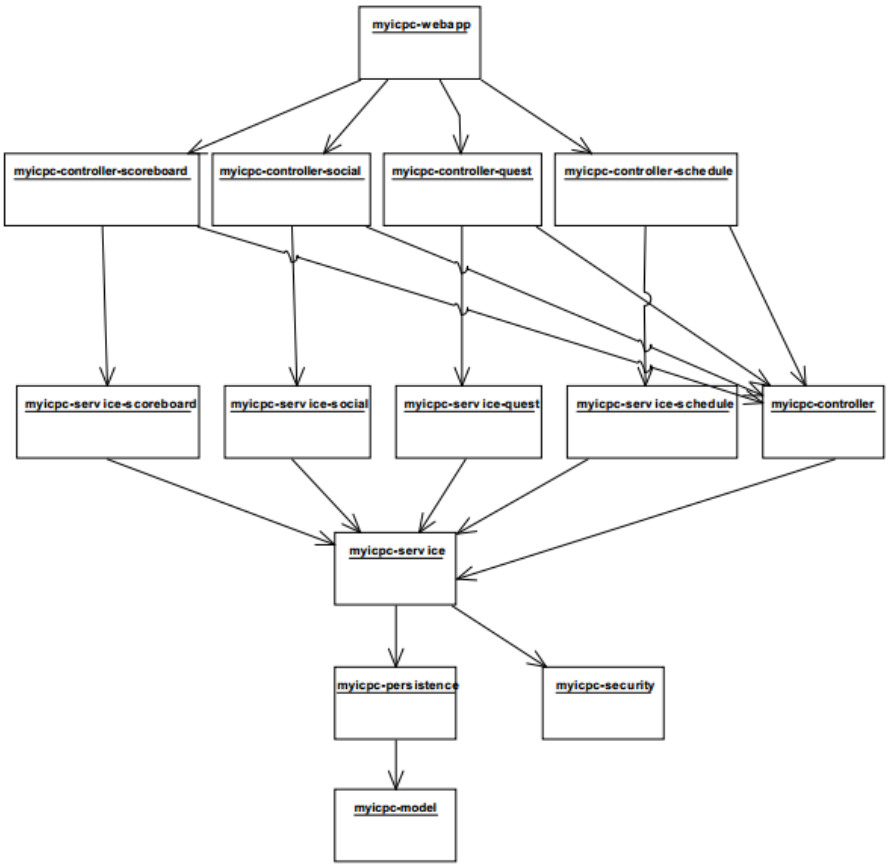
2.2.2 Spring

MyICPC je postaven na webovém frameworku Spring. Ten poskytuje zejména Inversion of Control (IoC), inicializaci jednotlivých tříd při startu aplikace a Context and Dependency Injection (CDI). [5] V projektu je použita konfigurace pomocí anotací.

Spring dále poskytuje rozhraní pro komunikaci s databází, takzvané repozitáře. Tyto repozitáře využívají nástroj Hibernate pro komunikaci s databází PostgreSQL. [6] Umožňují jednoduše definovat metody pro práci s databází bez zbytečného kódu a vyhnout se používání HQL dotazů pro běžné operace.

2.2.3 Java Message Service (JMS) a Redis

Java Message Service je API umožňující komunikaci mezi jednotlivými instancemi MyICPC, běžícími na různých uzlech clusteru. Funguje na bázi posílání zpráv pomocí front (Queue) a témat (Topic). [7] MyICPC dále využívá systém Redis jako zprostředkovatele této komunikace. [8]



Obrázek 2.1. Struktura Maven modulů systému MyICPC [2]

2.2.4 Java Server Pages (JSP)

Java Server Pages je technologie umožňující vytvářet dynamicky generované webové stránky pro aplikace vyvíjené v jazyce Java. Pro zobrazení vytvořených stránek je potřeba, aby aplikace běžela na aplikačním serveru poskytující servlet kontejner. [9]

2.2.5 AngularJS a ReactJS

AngularJS a ReactJS jsou aplikační frameworky pro vytváření prezentační logiky webových stránek v jazyce JavaScript. [10–11] Tyto frameworky se od sebe v mnoha ohledech odlišují. Největším rozdílem je způsob vytváření modelu zobrazeného dokumentu. Zatímco AngularJS pracuje s modelem vytvořeným v rozšířeném jazyce HTML a tento model za běhu upravuje, ReactJS model sám vytváří pomocí speciálních komponent implementovaných v jazyce JavaScript.

2.2.6 Atmosphere a Redis

Framework Atmosphere umožňuje přenos dat směrem od serveru ke klientovi. Poskytuje abstrakci pro přenosové technologie SSE a WebSocket. Během spojení serveru s klientem dojde k výběru jedné z těchto dvou technologií, v závislosti na klientském prohlížeči. [12] Odesílání dat z MyICPC směrem ke klientům zprostředkovává systém Redis.

2.3 Zpracování soutěžních dat (Event Feed)

Datový zdroj Event Feed, zpracovávaný během soutěže, má podobu XML souboru, ve kterém v průběhu soutěže přibývají data. K jeho zpracování je využívána služba EventFeedProcessor, implementovaná jako součást modulu myicpc-service-scoreboard. Zpracování jednotlivých objektů ze zdroje Event Feed zajišťuje služba EventFeedVisitor.

Nově vyhodnocené úlohy (TeamProblem) je třeba kromě uložení do databáze také zahrnout do celkových výsledků dané soutěže a zároveň aktualizovat skóre a pořadí všech týmů. Tuto službu zajišťuje komponenta FeedRunStrategy.

2.4 Prezentace soutěžních výsledků (Scoreboard)

Prezentaci aktuálních výsledků soutěže poskytují komponenty Scoreboard, Scorebar a Map. Scoreboard zobrazuje podrobné výsledky soutěže v podobě tabulky, ve které je pro každý tým a úlohu uveden aktuální stav hodnocení. Scorebar poskytuje vizualizaci skóre týmů pomocí řádkových diagramů. Map poskytuje vizualizaci pořadí týmů na mapě světa.

2.4.1 Komponenta Scoreboard

Komponenta Scoreboard je založena na datovém modelu vytvořeném z výsledků soutěže aktuálně uložených v databázi MyICPC. Datový model ve formátu JSON poskytuje komponentě služba ScoreboardService. Jako view slouží JSP stránka, využívající controller implementovaný ve frameworku AngularJS. Controller poskytuje stránce funkce pro práci s datovým modelem a jeho aktualizaci na základě nových dat přijatých pomocí frameworku Atmosphere. Zdrojový kód controlleru byl psán v jazyce CoffeeScript a překládán do jazyka JavaScript externím programem. [13]

2.4.2 Komponenty Scorebar a Map

Komponenty Scorebar a Map jsou implementovány podobně jako Scoreboard, využívají však datové modely upravené pro odlišnou formu prezentace.

2.5 Prezentace fotografií (Gallery)

K prezentaci fotografií z oficiální fotogalerie slouží komponenta Gallery. Komponenta využívá data ze systému Flickr jako zdroj fotografií a informací o jejich obsahu. [14] V detailním zobrazení poskytuje informace o událostech, týmech a osobách na fotografiích zobrazených včetně zvýraznění obličejů označených osob. Komponenta ovšem neposkytuje žádný prezentační mód a v základním režimu zobrazuje pouze náhledy fotografií oříznuté na čtverec, což neuspokojuje požadavky pro prezentaci na velkoplošných obrazovkách. K implementaci galerie byla použita opět kombinace technologií JSP a AngularJS.

Kapitola 3
MyICPC DataProvider

Modul DataProvider byl navržen jako middleware pro příjem soutěžních dat z nového modulu EventFeed Middleware a jejich ukládání do databáze MyICPC. V této kapitole je popsán jeho návrh a implementace. Jeho nasazení a testování je popsáno v šesté kapitole.

3.1 Event Feed a jeho zpracování

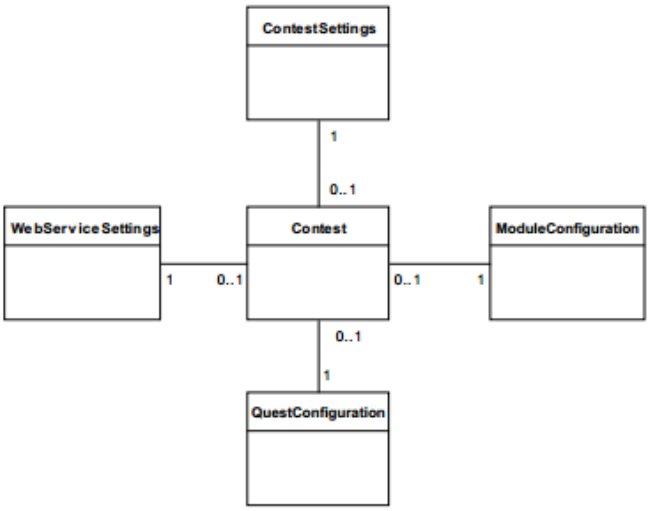
Během programovacích soutěží ACM-ICPC jsou k dispozici soutěžní data z Contest Data Serveru (CDS). [15] Primárním zdrojem těchto dat je Event Feed ve formátu XML. Původně byl tento zdroj parsován a zpracováván přímo systémem MyICPC v průběhu soutěže. Tento způsob zpracování ovšem představoval nadměrnou zátěž na systém a také při něm vznikaly problémy s dostupností dat v průběhu soutěže. Z toho důvodu jsme se rozhodli zpracování Event Feedu oddělit do samostatného modulu, označeného jako EventFeed Middleware. Tento modul zpracovává Event Feed v průběhu soutěže a poskytuje ho prostřednictvím REST API ve formátu JSON. Dále poskytuje druhý datový zdroj z CDS, kterým je JSON Scoreboard. Event Feed Middleware je vyvíjen současně v rámci jiné bakalářské práce. [16] Oba datové zdroje jsou specifikovány na stránce o systému Contest Data Server. [17–18]

3.2 Funkce modulu DataProvider

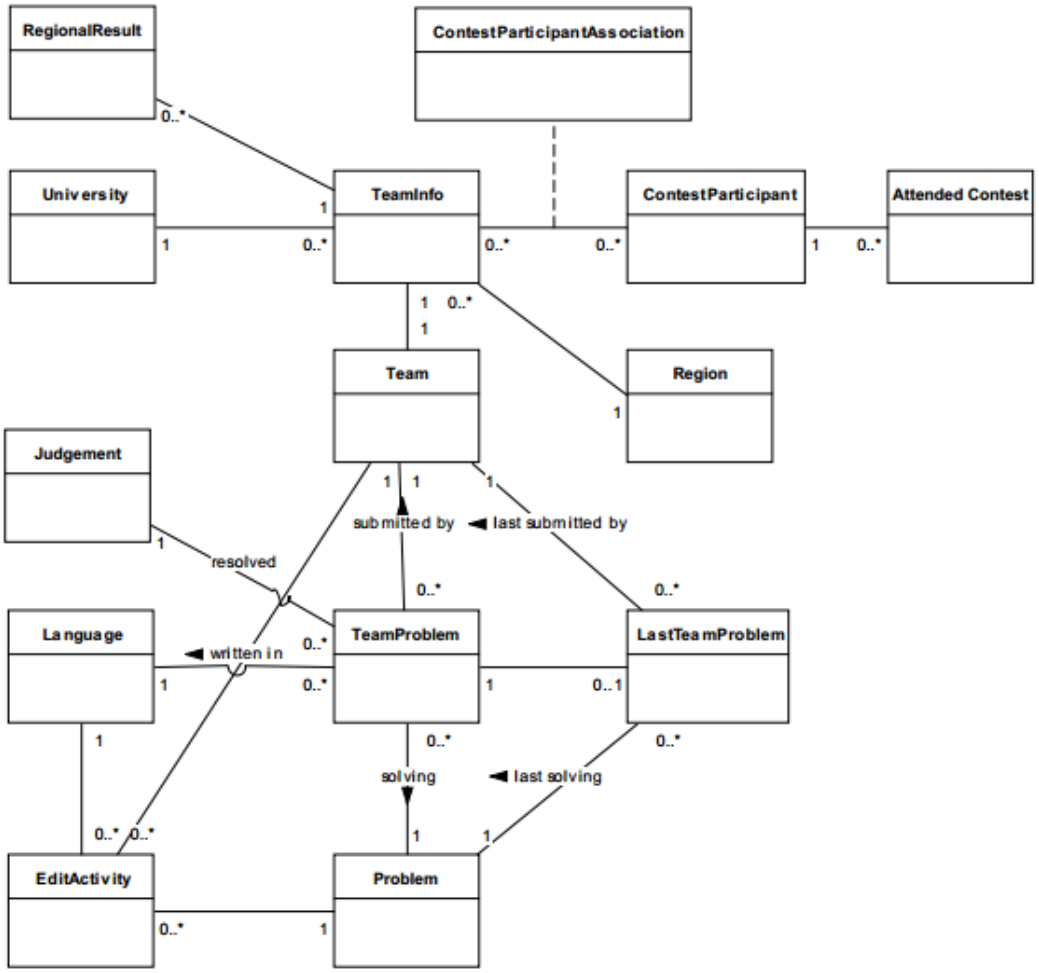
Účelem modulu DataProvider je přijímat zdroje Event Feed a JSON Scoreboard od modulu EventFeed Middleware a ukládat je do databáze systému MyICPC. Aby DataProvider mohl efektivně pracovat, potřebuje komunikovat s oběma moduly. Od Middleware potřebuje být notifikován o přítomnosti nových dat, aby nemusel provádět zbytečné požadavky na REST API. Dále potřebuje notifikovat MyICPC o uložení nových dat do databáze. Některá data je potřeba dále v MyICPC dále zpracovat podle zvolené strategie. Dále je potřeba v MyICPC aktualizovat datové zdroje poskytované připojeným klientům v reálném čase, zejména pomocí technologie SSE.

3.3 Architektura modulu DataProvider

DataProvider je navržen jako samostatná aplikace běžící na serveru WildFly. [19] Jeho návrh vychází z návrhu MyICPC, je také postaven na frameworku Spring a využívá některé moduly z MyICPC. Konkrétně se jedná o *myicpc-model* pro práci s doménovým modelem MyICPC a *myicpc-persistence* pro práci s databází MyICPC prostřednictvím repozitářů. Na diagramech 3.1 a 3.2 jsou znázorněny části doménového modelu potřebné pro zpracování zdroje Event Feed.



Obrázek 3.1. Doménový model pro soutěž [2]



Obrázek 3.2. Doménový model pro Scoreboard [2]

3.4 Příjem zpráv z EventFeed Middleware

DataProvider potřebuje reagovat na zprávy o nových datech z Middleware. K tomu využívá JMS frontu, do které jsou tyto zprávy doručovány přes JSM Bridge vytvořený modulem Middleware. Předpokládá se, že modul DataProvider bude nasazován jako první, vytvoří frontu pro příjem zpráv a Middleware s ní poté v okamžiku nasazení naváže spojení přes jím vytvořený JMS Bridge.

Jako typ zprávy je využíván serializovatelný objekt třídy EventFeed. Ke správné deserializaci objektu na straně příjemce je nutné, aby odesílatel a příjemce používaly stejnou implementaci příslušné třídy. V opačném případě příjemce takové zprávě neporozumí. [20] Příjem zpráv je v modulu DataProvider realizován pomocí anotace JmsListener. Metoda, přijímající zprávu, na ni reaguje provedením potřebné akce, závislé na typu události obsažené ve zprávě.

```
@JmsListener(destination =
  "java:/jboss/exported/jms/queue/NotificationQueueTarget")
@Transactional
public void processEventFeedNotification(FeedEvent feedEvent) {
  if (feedEvent == null) {
    return;
  }
  String contestId = feedEvent.getContestID();
  switch (feedEvent.getEventType()) {
    case HEADER:
      eventFeedReceiver.processContestHeaderEvent(contestId);
      break;
    case RUN:
      eventFeedReceiver.processSubmissionEvent(
        contestId, feedEvent.getEventIdentifier());
      break;
    case RESOLUTION:
      eventFeedReceiver.processResolutionEvent(
        contestId, feedEvent.getEventIdentifier());
      break;
    case ANALYST_MSG:
      eventFeedReceiver.processAnalystMessageEvent(
        contestId, Math.round(feedEvent.getEventIdentifier()));
      break;
    case SYS_EVENT_FINALIZED:
      eventFeedReceiver.finalizeContest(contestId);
      break;
    case SYS_EVENT_CANCEL:
      eventFeedReceiver.cancelContest(contestId);
      break;
  }
}
```

3.5 Příjem dat z EventFeed Middleware

DataProvider si udržuje stav zpracování zdroje Event Feed pro každou soutěž zvlášť v objektu EventFeedContestStatus. Informace o nových datech ve zdroji Event Feed

získává ze zpráv od EventFeed Middleware. Data jsou řazena chronologicky a opatřena časovými známkami, což umožňuje vyžádat si data vzniklá od určitého okamžiku. Nová data přijímá DataProvider pomocí REST rozhraní v pravidelných intervalech, avšak pouze tehdy, když mu nějaká data chybí. K tomu slouží metoda opatřená anotací Scheduled. K vytváření synchronních HTTP požadavků byla použita knihovna Unirest. Její použití znázorňuje následující ukázka metody pro příjem vyhodnocených úloh (TeamProblem).

```
public void requestRecentRuns(@NonNull String eventFeedContestId) {
  EventFeedContestSettings settings =
    contestSettingsStorage.getContestSettings(eventFeedContestId);
  final EventFeedContestStatus status =
    contestStatusStorage.getContestStatus(eventFeedContestId);
  final String contestCode = settings.getMyicpcContestCode();
  String requestUrl = settings.getProviderUrl()
    + EventFeedConstants.REQUEST_RUNS;
  String timestampParam = Double.toString(
    status.getProcessedRunTimestamp());
  logger.info("Requesting runs for contest {} since timestamp {}...",
    eventFeedContestId, timestampParam);
  try {
    HttpResponse<String> httpResponse = Unirest.get(requestUrl)
      .headers(settings.getRestHeaders())
      .routeParam("contestId", eventFeedContestId)
      .routeParam("timestamp", timestampParam)
      .asString();
    String jsonString = httpResponse.getBody();
    status.updateProcessedRunTimestamp(
      runProcessor.processRuns(jsonString, contestCode));
  } catch (UnirestException e) {
    logger.error("Run request for contest {} failed.",
      eventFeedContestId);
    e.printStackTrace();
  }
}
```

V požadavku na REST API je specifikován identifikátor soutěže a dále časová známka označující okamžik vzniku nejnovějšího objektu, který byl již zpracován. V odpovědi na tento požadavek tedy přijdou objekty vzniklé po tomto okamžiku.

Kromě vyhodnocených úloh jsou podobným způsobem přijímány také hlavička soutěže, obsahující zejména seznam týmů a zadaných úloh, zprávy Analyst Message a zdroj JSON Scoreboard.

3.6 Zpracování dat z EventFeed Middleware

Data z REST API modulu Middleware přichází jako odpověď na HTTP požadavek ve formátu JSON. Následně jsou tato data parsována, validována, mapována na datový model systému MyICPC a ukládána do jeho databáze pomocí repozitářů frameworku Spring. Po zpracování dat přijatých v jedné odpovědi se aktualizuje stav zpracování dané soutěže, aby v následujícím požadavku nebyla zbytečně posílána již zpracovaná data.

3.7 Odesílání zpráv do systému MyICPC

Systém MyICPC potřebuje v průběhu soutěže vědět o přítomnosti nových dat v databázi, aby je mohl dále zpracovávat a publikovat. K informování systému je využíváno posílání zpráv pomocí JMS na stejném principu, jaký využívá modul Middleware pro posílání zpráv do modulu DataProvider. Systém MyICPC by měl být nasazen permanentně a poskytovat JMS frontu modulu DataProvider, který s ní v okamžiku nasazení naváže spojení přes JMS Bridge.

3.8 Zpracování nových dat v systému MyICPC

Systém MyICPC přijímá JMS zprávy z modulu DataProvider v komponentě DataProviderNotificationReceiver pomocí anotace JmsListener. Na přijaté zprávy reaguje zpracováním nových dat v závislosti na jejich typu.

Nejdůležitější typ dat ze zdroje Event Feed představují vyhodnocené úlohy, které jsou v MyICPC reprezetované třídou TeamProblem. Hlavním účelem jejich zpracování je aktualizace bodových skóre týmů a jejich pořadí ve výsledkové listině. K tomu je používána stejná strategie, která byla původně implementována pro zpracování zdroje Event Feed, parsovaného systémem MyICPC přímo ze souboru ve formátu XML.

Kapitola 4
Komponenta Scoreboard

V této kapitole je popsán návrh a implementace nové verze komponenty Scoreboard.

4.1 Funkce komponenty Scoreboard

Komponenta Scoreboard je využívána pro prezentaci tabulky výsledků soutěže. Tabulka v průběhu soutěže zobrazuje vždy aktuální výsledky bez nutnosti obnovení stránky. Pořadí týmů je určeno dle pravidel ICPC počtem vyřešených úloh a celkovým časem řešení (časy úspěšných odevzdání úloh + penalizace za neúspěšná odevzdání). U každého týmu je dále zobrazen stav hodnocení jednotlivých úloh, včetně počtů odevzdání a časů řešení.

Scoreboard dále umožňuje uživateli označit si týmy, které chce sledovat. Tyto týmy se poté zobrazují v kopii na začátku tabulky, což je užitečné zejména při účasti většího počtu týmů, kdy velikost tabulky vyžaduje dlouhé scrollování. Podrobné informace o jednotlivých týmech a úlohách lze získat na samostatných stránkách přístupných pomocí odkazů v tabulce.

4.2 Původní implementace komponenty Scoreboard

V původní verzi tato komponenta využívala datový model založený výhradně na datech vzniklých zpracováním zdroje Event Feed. Tato data byla získána pomocí složitých dotazů do databáze, následně v servisní vrstvě převáděna do formátu JSON a poté prezentována pomocí webové stránky založené na frameworcích JSP a AngularJS. K aktualizaci dat v tabulce bez nutnosti obnovení stránky byly využívány technologie SSE a WebSocket, poskytované frameworkem Atmoshpere.

4.3 Nová verze komponenty Scoreboard

Nová verze komponenty Scoreboard je součástí projektu označeného pracovně jako MyICPC3. (Aktuálně používaná verze systému MyICPC nese označení 2.0) [21] V rámci projektu MyICPC3 jsou vyvíjeny single-page aplikace založené na frameworku ReactJS, které jsou následně využívány v prezentační vrstvě systému MyICPC. Aplikace jsou vyvíjeny pomocí metody Create React App.Díky tomu je možné je vyvíjet a testovat zcela samostatně a následně do systému MyICPC integrovat odladěnou produkční verzi. To představuje velkou výhodu, jelikož nasadit takovou aplikaci na malý server za účelem testování je výrazně jednodušší a rychlejší než nasadit celý systém MyICPC na server WildFly.

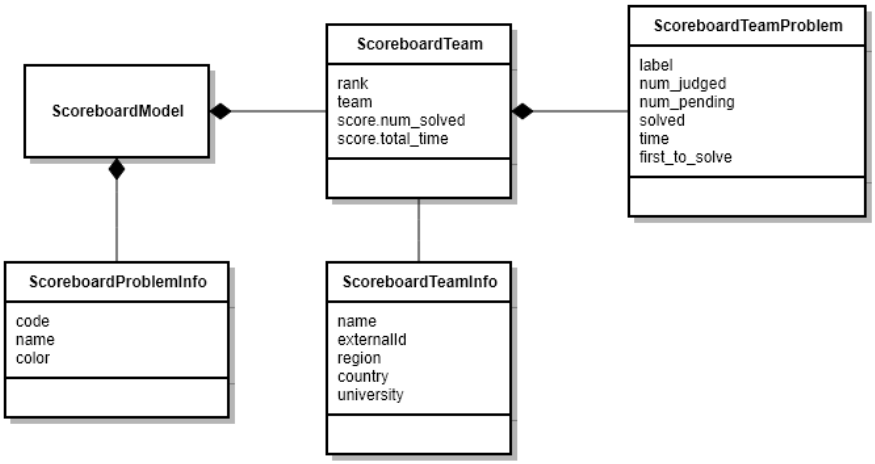
4.4 Metoda Create React App

Create React App je oficiálně podporovanou metodou vývoje aplikací v ReactJS. Využívá systém Webpack pro vytváření tzv. bundlů, které výrazně zmenšují počet souborů v produkční verzi aplikace. (Například pro nasazení jedné komponenty do systému MyICPC stačí pouze 4 výsledné soubory). Dále využívá překladač Babel, který umožňuje využívat nové vlastnosti jazyka JavaScript, představené ve verzi ES6, při zachování kompatibility se staršími verzemi webových prohlížečů. Celý překlad vytvořené aplikace se provádí jediným příkazem. [22]

4.5 Datový model pro Scoreboard

Základ pro datový model komponenty Scoreboard nově tvoří oficiální JSON Scoreboard, který je ukládán v nezměněné podobě do databáze MyICPC. JSON Scoreboard bsahuje data o průběžných výsledcích soutěže, neobsahuje však podrobnosti o soutěžních týmech a řešených úlohách. Data o týmech a úlohách jsou proto nadále poskytována servisní vrstvou systému MyICPC, kde jsou převáděna do formátu JSON a spolu s objektem JSON Scoreboard tvoří ucelený datový model. Model je znázorněn na diagramu 4.1. ScoreboardProblemInfo a ScoreboardTeamInfo reprezentují data přidaná do modelu systémem MyICPC.

Tento model je poté posílán klientům jako součást webové stránky s aplikací Scoreboard a současně je poskytován klientům pomocí kanálu založeného na Atmosphere frameworku, který posílá data klientům pomocí systému Redis. Tím je zajištěna konzistence ReactJS komponent v aplikaci Scoreboard a jejich automatická aktualizace v průběhu soutěže.

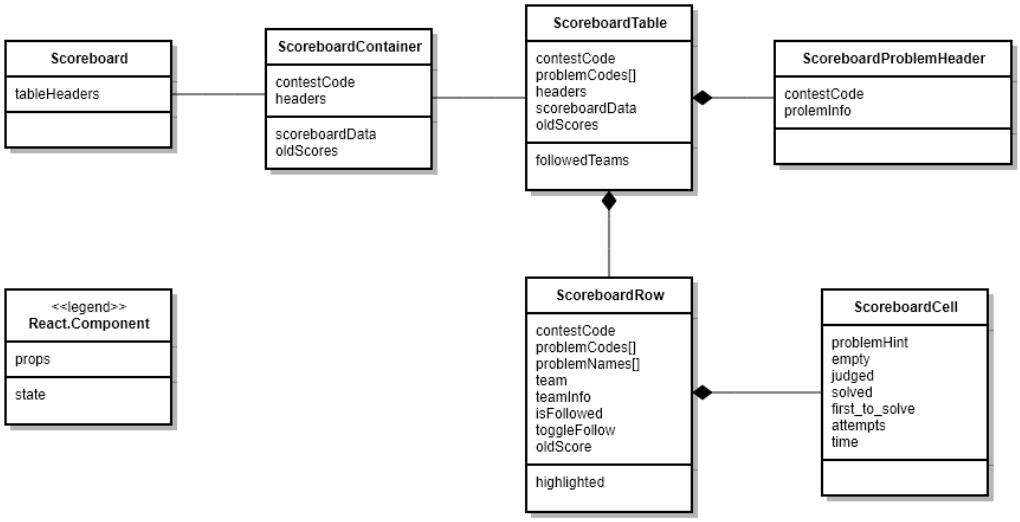


Obrázek 4.1. Datový model aplikace Scoreboard

4.6 Objektový model React aplikace

Objektový model je znázorněn na diagramu 4.2. Scoreboard je kořenovou komponentou aplikace. ScoreboardContainer obsahuje datový model a reaguje na jeho změny. Score-

boardTable zajišťuje vykreslení tabulky a poskytuje funkci připnutí oblíbených týmů na začátek tabulky. ScoreboardRow představuje řádek tabulky, obsahující výsledky jednoho týmu. ScoreboardCell představuje buňku v tabulce, obsahující stav jedné úlohy pro daný tým.



Obrázek 4.2. Objektový model aplikace Scoreboard (ReactJS Components)

4.7 Uživatelské rozhraní komponenty Scoreboard

Pro novou verzí komponenty Scoreboard byl navržen nový vzhled tabulky. Účelem návrhu bylo zejména zlepšení čitelnosti a přehlednosti tabulky a efektivnější využití místa na obrazovce s ohledem na možnosti současných mobilních zařízení. V tabulce bylo použito barevné schéma definované standardem Material Design. [23] Původní a nové uživatelské rozhraní je možné porovnat na obrázcích 4.3 a 4.4 . Nová tabulka navíc umožňuje zobrazení podrobnějších informací o datech v jednotlivých buňkách. Pro zobrazení těchto informací stačí najet kurzorem myši na příslušnou buňku.

4.8 Aktualizace tabulky během soutěže

V průběhu soutěže přijímá aplikace Scoreboard aktualizovaný datový model, poskytovaný systémem MyICPC pomocí technologie SSE nebo WebSocket. K vytvoření spojení se serverem využívá framework Atmosphere. [12] Přijetí nových dat vyvolá změnu stavu komponenty ScoreboardContainer, což způsobí automatické překreslení tabulky.

```
onNewData() {
  const ref = this;
  return (response: any) => {
    try {
      const updatedData = JSON.parse(response.responseBody);
      const scoresToSave = this.getScoreMap();
      ref.setState({
```

```
          scoreboardData: updatedData,
          oldScores: scoresToSave
        });
      } catch (error) {
        return console.log(
          "An error occurred while updating scoreboard: " + error);
      }
    };
  }
```

Po překreslení tabulky dojde ke krátkodobému zvýraznění řádků s týmy, jejichž skóre se oproti předchozímu stavu změnilo. Zvýraznění je řešeno pomocí stavové proměnné highlighted ve třídě ScoreboardRow. Tato proměnná řídí přítomnost CSS třídy zajišťující zvýraznění příslušného řádku.

```
componentWillReceiveProps(newProps: any) {
  const {team, oldScore} = newProps;
  if (team.score.num_solved > oldScore) {
    this.setState({
      highlighted: true;
    });
    setTimeout(function() {
      this.setState({highlighted: false});
      }.bind(this), 5000);
  }
}
```

4.9 Integrace do systému MyICPC

4.9.1 Atmosphere kanál jsonscoreboard

V systému MyICPC byl vytvořen nový Atmosphere kanál, poskytující nový datový model aplikaci Scoreboard. Data v kanálu jsou publikována v reakci na JMS zprávu od modulu DataProvider. Ten informuje MyICPC o uložení nové verze objektu JSON Scoreboard do databáze.

4.9.2 ScoreboardService

ScoreboardService/ScoreboardServiceImpl nyní obsahuje metodu getFullJsonScoreboardModel(), která generuje nový model pro aplikaci Scoreboard ve formátu JSON. Tento model slouží pro inicializaci aplikace Scoreboard při načtení webové stránky a dále pro aktualizaci dat v aplikaci prostřednictvím Atmosphere kanálu. Vytváření nového modelu by mělo být efektivnější než u předchozího modelu, jelikož odpadlo generování výsledkové listiny v JSON formátu a s ním spojené složité dotazy do databáze.

4.9.3 ScoreboardController

Metoda scoreboard() ve třídě ScoreboardController byla upravena tak, aby do modelu stránky s aplikací Scoreboard přidávala nový datový model. Původní datový model byl z metody odstraněn.

4.9.4 Scoreboard View

View pro Scoreboard poskytuje JSP stránka scoreboard.jsp. Ze stránky byla odstraněna původní struktura s kostrou dokumentu, určená pro logiku implementovanou v AngularJS. Místo ní byl vložen build ReactJS aplikace, který je tvořen čtyřmi soubory (CSS+map a JS+map). ReactJS si narozdíl od AngularJS vystačí s minimální kostrou dokumentu, jelikož objektový model dokumentu (DOM) generuje pomocí metod render() ve svých komponentách.

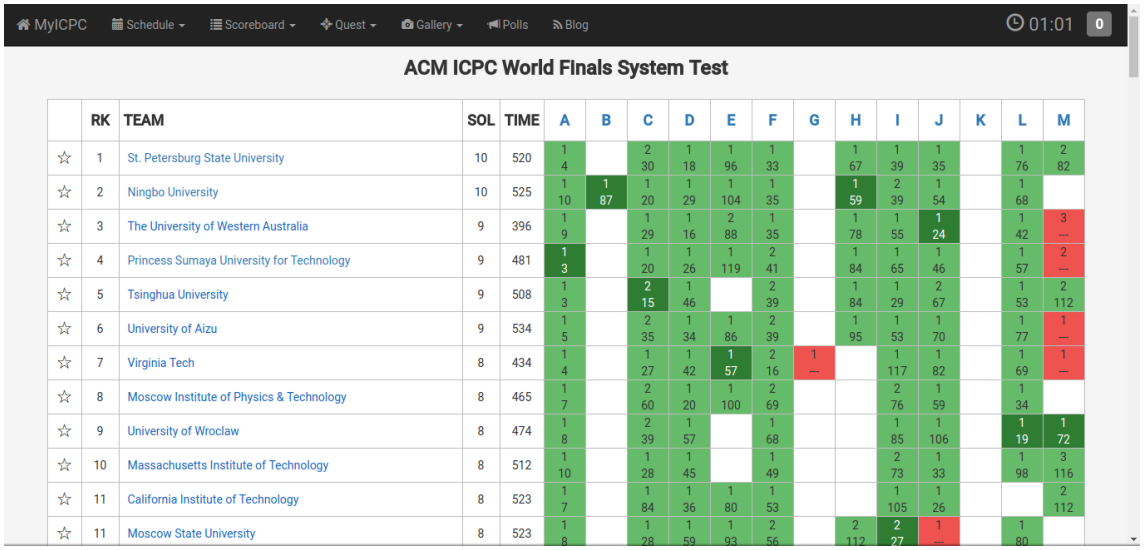
4.10 Komponenty Scorebar a Map

Systém MyICPC dále poskytuje vizualizaci výsledků pomocí diagramů v komponentě Scorebar a dále pomocí mapy světa v komponentě Map. Tyto komponenty zatím fungují na původním datovém modelu a používají původní implementaci v AngularJS. Do budoucna se v rámci projektu MyICPC3 počítá s přepracováním těchto komponent do ReactJS podobným způsobem, jako u komponenty Scoreboard.

MyICPC | Schedule | Scoreboard | Quest | Gallery | Polls | Blog — 05:00 — 1

	Rk	Name	Solved	Time	A	B	C	D	E	F	G	H	I	J	K	L
☆	1	St. Petersburg ITMO University	10	1093	1 - 71	2 - 219	1 - 59	1 - 88	1 - 9	1 - 28	1 - 248		1 - 16		2 - 121	3 - 154
☆	2	University of Warsaw	10	1240	1 - 58	1 - 287	2 - 91	1 - 208	1 - 9	1 - 24	1 - 125		1 - 15	1	1 - 201	4 - 142
☆	3	Seoul National University	10	1320	4 - 246	2 - 236	2 - 125	1 - 29	1 - 10	1 - 39	1 - 149	5	1 - 17	1	1 - 185	4 - 124
☆	4	St. Petersburg State University	10	1377	1 - 63	4 - 299	1 - 100	1 - 81	1 - 7	1 - 28	2 - 270		1 - 22		1 - 241	3 - 146
☆	5	Moscow Institute of Physics & Technology	10	1460	1 - 91	3 - 252	1 - 54	3 - 207	1 - 8	1 - 29	1 - 287		1 - 15		1 - 168	2 - 249
☆	6	Tsinghua University	9	964	1 - 59		1 - 93	1 - 113	1 - 6	1 - 14	1 - 221	3	1 - 21		2 - 233	2 - 164
☆	7	Peking University	9	1245	3 - 188		1 - 60	2 - 259	1 - 6	1 - 43	2 - 236		1 - 13		1 - 274	1 - 86
☆	8	Fudan University	8	1103	2 - 114		1 - 67	5 - 275	1 - 13	2 - 43	1 - 261		1 - 17		1	1 - 193
☆	9	KAIST	8	1119	4 - 213		2 - 122	2 - 70	1 - 5	1 - 20		1	1 - 12		2 - 205	5 - 272
☆	10	Ural Federal University	8	1264	3 - 188		1 - 83		1 - 20	1 - 43	1 - 252		1 - 22		2 - 291	3 - 267
☆	11	KTH - Royal Institute of Technology	8	1357	7 - 240		1 - 120	9	1 - 9	1 - 36	2 - 231		1 - 14		6 - 290	3 - 137
☆	12	The University of Tokyo	8	1490	7 - 201		5 - 274	3 - 283	1 - 13	1 - 58	1 - 132		1 - 24			3 - 225
☆	13	Shanghai Jiao Tong University	7	767	1 - 126		1 - 125	2 - 250	1 - 10	1 - 24			1 - 6			2 - 186
☆	14	Perm State University	7	791	13	3 - 216	1 - 111	6	1 - 8	1 - 22			1 - 13		3 - 82	2 - 239
☆	15	University of Electronic Science and Technology of	7	846	1 - 103		2 - 137	1 - 191	1 - 12	1 - 29			1 - 18			3 - 296
☆	16	University of New South Wales	7	908	0		1 - 94	3 - 256	1 - 10	3 - 121	1 - 138		1 - 14		1	1 - 195
☆	17	University of Helsinki	7	956	9 - 268		1 - 115	1 - 82	3 - 16	1 - 30	1 - 225		1 - 20			6
☆	18	University of Waterloo	7	1019	28	2 - 219	1 - 152	4 - 85	1 - 5	2 - 97			1 - 12		6 - 249	

Obrázek 4.3. Původní uživatelské rozhraní komponenty Scoreboard



	RK	TEAM	SOL	TIME	A	B	C	D	E	F	G	H	I	J	K	L	M
☆	1	St. Petersburg State University	10	520	1 4		2 30	1 18	1 96	1 33		1 67	1 39	1 35		1 76	2 82
☆	2	Ningbo University	10	525	1 10	1 87	1 20	1 29	1 104	1 35		1 59	2 39	1 54		1 68	
☆	3	The University of Western Australia	9	396	1 9		1 29	1 16	2 88	1 35		1 78	1 55	1 24		1 42	3 ---
☆	4	Princess Sumaya University for Technology	9	481	1 3		1 20	1 26	1 119	2 41		1 84	1 65	1 46		1 57	2 ---
☆	5	Tsinghua University	9	508	1 3		2 15	1 46		2 39		1 84	1 29	2 67		1 53	2 112
☆	6	University of Aizu	9	534	1 5		2 35	1 34	1 86	2 39		1 95	1 53	1 70		1 77	1 ---
☆	7	Virginia Tech	8	434	1 4		1 27	1 42	1 57	2 16	1 ---		1 117	1 82		1 69	1 ---
☆	8	Moscow Institute of Physics & Technology	8	465	1 7		2 60	1 20	1 100	2 69			2 76	1 59		1 34	
☆	9	University of Wroclaw	8	474	1 8		2 39	1 57		1 68			1 85	1 106		1 19	1 72
☆	10	Massachusetts Institute of Technology	8	512	1 10		1 28	1 45		1 49			2 73	1 33		1 98	3 116
☆	11	California Institute of Technology	8	523	1 7		1 84	1 36	1 80	1 53			1 105	1 26			2 112
☆	11	Moscow State University	8	523	1 8		1 28	1 59	1 93	2 56		2 112	2 27	1 ---		1 80	

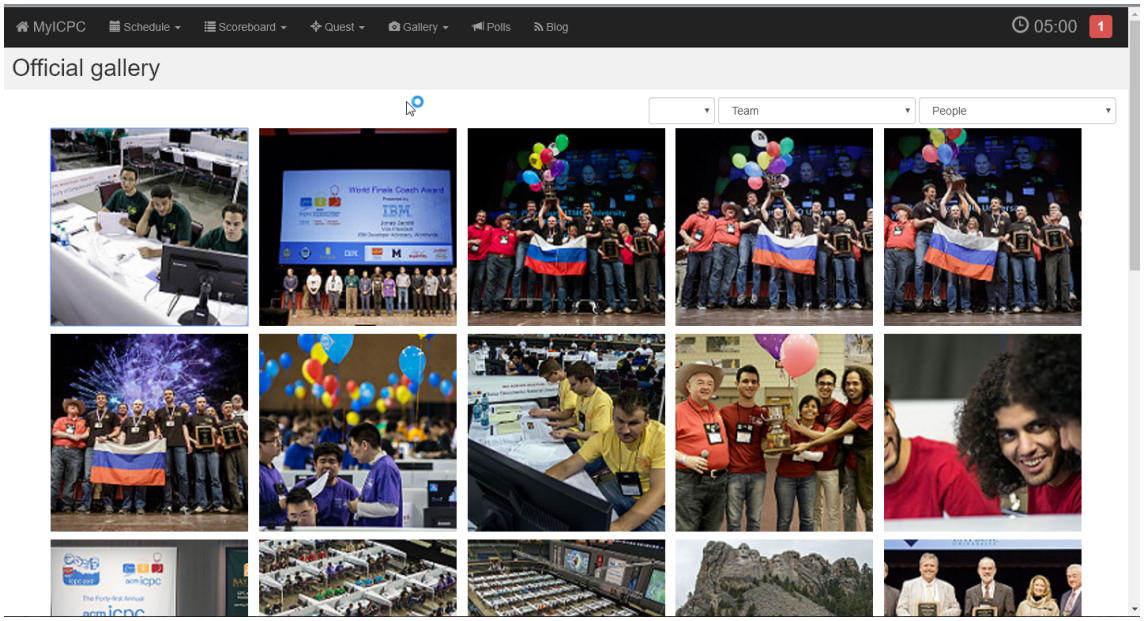
Obrázek 4.4. Nové uživatelské rozhraní komponenty Scoreboard

Kapitola 5
Komponenta PhotoKiosk

V této kapitole je popsán návrh a implementace nové komponenty PhotoKiosk. Implementace této komponenty je náhradou za implementaci komponenty Kiosk v rámci tématu této bakalářské práce.

5.1 Funkce komponenty PhotoKiosk

PhotoKiosk vznikl za účelem poskytnutí prezentace fotografií z oficiální galerie ICPC v divácky atraktivnější formě, než nabízí současná komponenta Gallery. 5.1 Měl by primárně sloužit pro prezentaci fotografií na velkoplošných obrazovkách bez zásahu uživatele. Z toho důvodu je třeba, aby se v něm v průběhu času střídaly fotografie z galerie a zobrazovaly detaily náhodných fotografií.



Obrázek 5.1. Uživatelské rozhraní komponenty Gallery

5.2 Návrh komponenty PhotoKiosk

Návrh PhotoKiosku vychází z nové verze komponenty Kiosk, která byla implementována v letošním roce pro prezentaci příspěvků ze sociálních sítí. Jedná se o single-page aplikaci implementovanou ve frameworku ReactJS v rámci projektu MyICPC3. [21] Grafický návrh Kiosku je založen na knihovně Material UI, implementující standard Material Design [24] a poskytuje mřížku karet s notifikacemi, která vyplňuje celou obrazovku. Kiosk také poskytuje funkci automatického zvětšení náhodné karty na celou obrazovku včetně animací.

5.3 Zdroj fotografií

Zdrojem fotografií pro PhotoKiosk je systém Flickr, který poskytuje uložené fotografie v mnoha různých rozlišeních včetně podrobných informací ve formě tagů. Pro vyhledávání fotografií nabízí rozsáhlé API, poskytující podrobná data v různých formátech. [14] Příklady dotazů a odpovědí...

5.4 Grafický návrh

Pro prezentaci v komponentě PhotoKiosk byly zvoleny fotografie pořízené na šířku, zmenšené na rozměr 800 pixelů (šířka). Fotografie pořízené na výšku by na kartách nevypadaly dobře, proto jsou z výsledků filtrovány. Galerie pro rok 2017 nabízí 50 fotografií, z toho 45 vyhovuje požadavkům. Mřížka zobrazuje ve výchozím nastavení 4x3 karty, což vyhovuje typickým rozměrům fotografií a obrazovek.

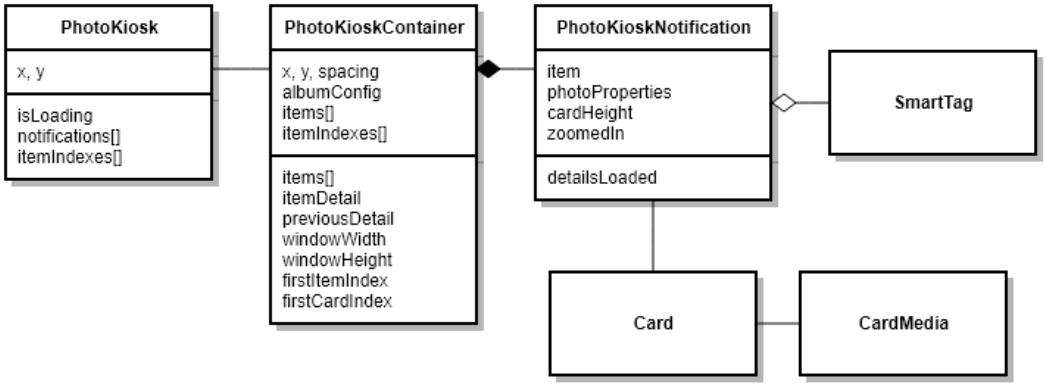
Pro automatické střídání fotografií byl navržen 30-sekundový cyklus, rozdělený na tři 10-sekundové intervaly. Na konci prvního intervalu dochází k náhradě náhodné fotografie za jinou. Po druhém intervalu se zobrazí detail náhodné karty na celou obrazovku. Detail je opatřen překryvnou vrstvou s informacemi o fotografii, získanými z jejích tagů. Po třetím intervalu se detail skryje a karta se vrátí zpět na původní místo v mřížce. Ukázky rozhraní je možné najít na obrázcích 5.2 a 5.3 a dále na videoukázce, která je součástí CD přílohy.

Obrázek 5.2. Uživatelské rozhraní komponenty PhotoKiosk

5.5 Objektový model React aplikace

Objektový model aplikace je znázorněn na diagramu 5.4. Jeho základem je React komponenta PhotoKiosk, která získává hlavní data o fotografiích ze systému Flickr a vytváří z nich datový model. Ten je předáván do komponenty PhotoKioskContainer, která obsahuje logiku pro běh prezentace fotografií. PhotoKioskContainer obsahuje instanci komponenty PhotoKioskNotification pro každou fotografii aktuálně zobrazenou v mřížce. SmartTag reprezenuje informaci o osobě, týmu nebo události zobrazené na fotografii,

Obrázek 5.3. Uživatelské rozhraní komponenty PhotoKiosk – detail fotografie



Obrázek 5.4. Objektový model aplikace PhotoKiosk (ReactJS)

získanou z tagu přidaného k fotografii v systému Flickr. Pro zpracování těchto tagů slouží komponenta GalleryService, implementovaná s využitím zdrojových kódů z OfficialGalleryService, poskytujícího prezentační logiku pro MyICPC komponentu Gallery.

5.6 Prezentační logika

Prezentace je řízena v komponentě PhotoKioskContainer časovačem nastaveným metodou setInterval(). Časovač každých 10 sekund volá metodu onTimerTick(). Tato metoda mění na základě aktuální fáze stav komponenty, což vede k automatickému překreslení obrazovky.

```
_onTimerTick() {
  const { items, itemDetail,
    firstCardIndex, firstItemIndex } = this.state;
  const { x, y } = this.props;
  this.ticks++;
```

```
      switch (this.ticks) {
        case 1: // change one photo in grid
          this.setState({
            firstCardIndex: (firstCardIndex + 1) % (x*y),
            firstItemIndex: (firstItemIndex + 1) % items.length
          });
          break;
        case 2: // show detail of random photo
          let detailIndex = Math.floor(Math.random() * x * y);
          this.setState({
            itemDetail:  detailIndex
           });
          break;
        case 3: // hide detail
          this.ticks = 0;
          this.setState({
            itemDetail: null,
            previousDetail: itemDetail
          });
    }
  }
```

Fotografie zobrazené na jednotlivých pozicích jsou určeny pomocí náhodně zamíchaných polí indexů, která se v každém cyklu posouvají o jednu pozici (jedno pole pro indexy fotografií, druhé pro indexy pozic). Díky tomu dochází k rovnoměrnému střídání fotografií tak, že v každém cyklu je nahrazena právě jedna fotografie, zatímco ostatní zůstavají na svých pozicích. Zároveň je možné při každém obnovení stránky získat jiné složení fotografií.

5.7 Integrace do systému MyICPC

5.7.1 PhotoKioskController

Do třídy PhotoKioskController byla přidána metoda photoKiosk(), poskytující nové view s komponentou PhotoKiosk. Model pro PhotoKiosk není potřeba, jelikož veškerá potřebná data jsou získávána ze systému Flickr.

5.7.2 PhotoKiosk View

Jako view pro PhotoKiosk byla přidána JSP stránka photoKiosk.jsp. Do stránky byl vložen build ReactJS aplikace, který je tvořen čtyřmi soubory (CSS+map a JS+map).

Kapitola 6
Nasazení a testování nové architektury

V této kapitole je popsáno nasazení celého systému a jeho testování pomocí simulace průběhu soutěže. Otestování celého systému v praxi je třeba zejména kvůli nové verzi komponenty Scoreboard, která nyní ke své činnosti v průběhu soutěže vyžaduje spolupráci všech nových modulů.

6.1 Nasazení systému

Nasazení hlavní aplikace MyICPC je možné jako Standalone (jedna instance) nebo jako Cluster (více stejných instancí), v závislosti na plánované zátěži. V případě Clusteru spolu jednotlivé instance komunikují pomocí JMS zpráv posílaných prostřednictvím systému Redis a jsou řízeny jedním modulem Master. Ostatní moduly (EventFeed Middleware a Data Provider) jsou navrženy jako Standalone. Nasazení je plánováno na instance aplikačního serveru WildFly v následujícím pořadí:

- Master + Webapp
- Data Provider
- EventFeed Middleware

Pořadí nasazení je třeba dodržet proto, aby fungovala komunikace pomocí JMS zpráv mezi uvedenými moduly. Například modul Data Provider vytváří během startu JMS Bridge, pomocí kterého se připojuje na JMS frontu vytvořenou v modulu Webapp. Na stejném principu funguje i propojení modulů Data Provider a Event Feed Middleware.

Nasazení systému v konfiguraci použité pro testování Scoreboardu je znázorněno na diagramu 6.1

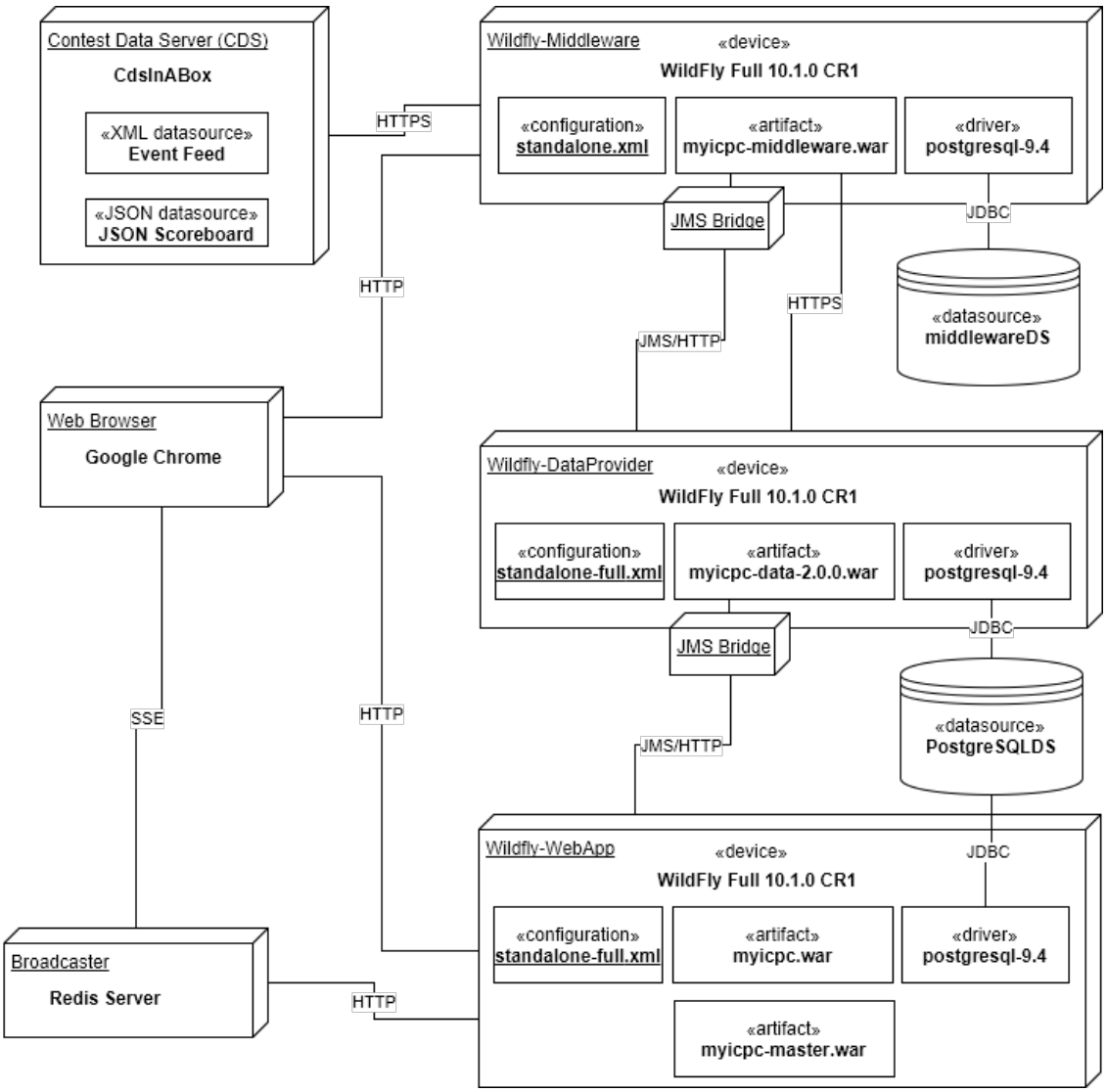
6.2 Forma testování

Testování probíhalo formou simulace průběhu soutěže. K simulaci byl použit systém CdsInABox, který přehrává zrychlený průběh soutěže World Finals 2016. CdsInABox během přehrávání poskytuje dva datové zdroje- Event Feed a JSON Scoreboard, které jsou v souladu se specifikací organizátora soutěže [15], poskytují tedy data ve stejném formátu, jako zdroje používané v reálných soutěžích od roku 2016 dosud.

6.3 Požadavky na testování

Testovací konfigurace by měla splňovat následující požadavky:

- Unixový operační systém
- Běhové prostředí jazyka Java (JRE)
- Databázový systém PostgreSQL
- Systém Redis
- Webový prohlížeč



Obrázek 6.1. Diagram nasazení systému (Testování Scoreboardu)

6.4 Konfigurace pro testování

6.4.1 Databáze

K testu jsou potřeba dvě PostgreSQL databáze. První databáze slouží jako datový zdroj MiddlewareDS pro modul Event Feed Middleware. Její schéma si tento modul vytváří sám při spuštění. Druhá databáze slouží jako datový zdroj PostgreSQLDS pro systém MyICPC. Schéma této databáze bylo inicializováno daty ze stávající systému MyICPC (SQL dump). Schéma bylo poté upraveno SQL skriptem, aby odpovídalo aktuálnímu datovému modelu systému (nový model ještě nebyl nasazen v praxi, proto je třeba přidat podporu pro persistentní třídy, které v původním modelu nejsou). Použitý skript je součástí CD přílohy.

6.4.2 Aplikační servery

K testu jsou potřeba tři instance aplikačního serveru WildFly. Instance musí být nakonfigurovány tak, aby běžely na různých portech. Konfigurace se prování pomocí XML

souborů, jejichž ukázky jsou součástí CD přílohy. V konfiguračních souborech je třeba nastavit zejména:

- Porty, na kterých server poběží
- Datové zdroje poskytované databází PostgreSQL a nasazení ovladače k databázi PostreSQL
- JMS fronty pro příjem a odesílání zpráv
- JMS Bridge pro spojení JMS front běžících na odlišných serverech

Dále je třeba v instancích serveru WildFly vytvořit uživatelské role použité pro propojení modulů pomocí JMS Bridge. Jejich vytvoření se provádí pomocí skriptu add-user.sh.

6.4.3 Konfigurace modulu DataProvider

Modul DataProvider potřebuje ke své činnosti konfiguraci pro soutěže, které má zpracovávat. Konfigurace se provádí pomocí jednoduchého XML souboru *eventFeedContestSettings.xml*, jehož ukázka je součástí CD přílohy. Soubor je třeba vložit do adresáře *bin* v příslušné instanci serveru WildFly.

6.4.4 Redis

Během testování je třeba, aby systém MyICPC měl přístup k běžícímu serveru systému Redis. Server musí mít nakonfigurovanou autorizaci, která je definovaná v systému MyICPC. Ověření autorizace je možné provést pomocí klienta Redis příkazem *AUTH*. [25]

6.5 Průběh testování

Nejprve je třeba uvést databázi MyICPC do stavu očekávaného před začátkem soutěže. V databázi by měly být informace o testované soutěži, neměla by tam ale být žádná data z průběhu soutěže (Event Feed a JSON Scoreboard).

Jako první spustíme instanci se systémem MyICPC (WildFly-Webapp). Po její spuštění se ujistíme, že fungují webové stránky poskytované systémem. Stránka s komponentou Scoreboard by měla zobrazit informaci, že Scoreboard zatím není k dispozici.

Jako druhou spustíme instanci s modulem Data Provider (WildFly-DataProvider). Poté spustíme instanci s modulem Event Feed Middleware (WildFly-Middleware). V jeho uživatelském rozhraní zaregistrujeme testovanou soutěž a zahájíme parsování dat.

Nakonec spustíme server testovacího systému SystemTestInABox. Jakmile testovací systém vygeneruje první soutěžní data, mělo by dojít k vykreslení tabulky s první odevzdanou úlohou v komponentě Scoreboard. V tabulce by se poté měly postupně objevovat další odevzdané úlohy a současně s tím by se měla celá tabulka aktualizovat na základě aktuálních výsledků soutěže. Celá simulace průběhu soutěže trvá něco málo přes hodinu a na jejím konci by tabulka měla zobrazovat konečné výsledky soutěže.

Průběh celého testu je zaznamenán na videu, které je součástí CD přílohy.

Kapitola 7
Závěr

V rámci této bakalářské práce se nám, společně s kolegou Vladem Gorbunovem, vyvíjejícím modul EventFeed Middleware, podařilo oddělit logiku parsování soutěžních dat a jejich ukládání do databáze systému MyICPC do samostatných modulů. Tyto moduly jsme poté propojili se systémem MyICPC. Tento krok by měl do budoucna umožnit zlepšení škálovatelnosti systému a jeho spolehlivosti v průběhu velkých programovacích soutěží.

Zároveň byla v rámci této práce navržena nová verze prezentační vrstvy pro komponentu Scoreboard, která by měla zpřehlednit a zatraktivnit prezentaci tabulky s výsledky soutěže. Dále byla navržena komponenta PhotoKiosk, jejíž účelem je poskytnout atraktivní prezentaci fotografií z oficiální galerie.

Nově navržené moduly a komponenty byly integrovány do systému MyICPC a otestovány pomocí simulace průběhu programovací soutěže.

Literatura

[1] *The ACM-ICPC International Collegiate Programming Contest*.
https://icpc.baylor.edu/.

[2] Roman Smetana. *Next Generation of Second-Screen - Realtime application MyICPC*. 2016.
https://icpc.baylor.edu/.

[3] Roman Smetana. *MyICPC 2.0 GIT repository*.
https://bitbucket.org/smetarom/myicpc-2.0.

[4] *Apache Maven Project*.
https://maven.apache.org/.

[5] *Spring by Pivotal*.
https://spring.io/.

[6] *PostgreSQL – The world's most advanced open source database*.
https://www.postgresql.org/.

[7] *Wikipedia, Java Message Service*.
https://en.wikipedia.org/wiki/Java_Message_Service.

[8] *Redis*.
https://redis.io/.

[9] *Wikipedia, JavaServer Pages*.
https://en.wikipedia.org/wiki/JavaServer_Pages.

[10] *AngularJS*.
https://angularjs.org/.

[11] *React – A JavaScript Library for Building User Interfaces*.
https://facebook.github.io/react/.

[12] *Atmosphere Framework*.
https://github.com/Atmosphere.

[13] *CoffeeScript*.
https://coffeescript.org/.

[14] *The Flickr API*.
https://flickr.com/services/api/.

[15] *Contest Data Server*.
https://clics.ecs.baylor.edu/index.php/CDS/.

[16] Vlad Gorbunov. *Data Middleware for ACM-ICPC*.
https://github.com/vldd/myicpc-middleware/.

[17] *Event Feed 2016*.
https://clics.ecs.baylor.edu/index.php/Event_Feed_2016/.

[18] *JSON Scoreboard 2016*.
https://clics.ecs.baylor.edu/index.php/JSON_Scoreboard_2016/.

[19] *WildFly Application Server*.
https://http://wildfly.org/.

[20] Mark Richards, Richard Monson-Haefel a David A. Chappell. *Java Message Service*. Sebastopol, CA: O'Reilly Media, Inc., 2009. ISBN 978-0-596-52204-9.

[21] Robert Cepa. *MyICPC3*.
https://github.com/robertcepa/MyICPC3/.

[22] *Create React App – Create React apps with no build configuration*.
https://github.com/facebookincubator/create-react-app/.

[23] *Material Design Color Palette*.
http://material.io/guidelines/style/color.html/.

[24] *Material-UI – A Set of React Components that Implement Google's Material Design*.
http://www.material-ui.com/.

[25] *Redis - Connection Auth Command*.
https://www.tutorialspoint.com/redis/connection_auth.

Příloha A
Zkratky

API	Application programming interface
CDS	Contest Data Server
CSS	Cascading Style Sheets
ES6	ECMAScript 6
HQL	Hibernate Query Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICPC	International Collegiate Programming Contest
JMS	Java Message Service
JS	JavaScript
JSON	JavaScript Object Notation
JSP	JavaServer Pages
REST	Representational state transfer
SQL	Structured Query Language
SSE	Server-sent events
UI	User Interface
XML	Extensible Markup Language

Příloha B
Obsah CD

root
- /config – Konfigurační soubory, použité pro testování
- /document – Zdrojové soubory tohoto dokumentu
- /sources – Zdrojové kódy vytvořených modulů a aktualizovaného systému MyICPC
- /video – Ukázky z testování komponent Scoreboard a PhotoKiosk

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Ondřej Musil

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: Realtime prezentace soutěžních výsledků a tweetů v systému MyICPC

Pokyny pro vypracování:

Analyzujte systém pro personifikovaný pohled nad realtime událostmi programovacích soutěží aplikace MyICPC.. Navrhněte řešení, které umožní realtime prezentaci soutěžních výsledků (scoreboard) v průběhu soutěže pro velký počet klientů. Dále navrhněte řešení umožňující realtime prezentaci uživatelských příspěvků ze sociální sítě Twitter (kiosk). Implementujte uživatelské rozhraní těchto modulů (scoreboard, kiosk) v rámci webové aplikace MyICPC. Dále vytvořte middleware pro efektivní parsování twitter feedu a komunikaci s informacemi o stavu soutěže umožnující nasazení celého rešení do cloudového prostředí. Otestujte funkčnost a efektivitu vašeho řešení pomocí simulace reálného nasazení systému. Při návrhu architektury, implementaci a testování spolupracujte s V. Gorbunovem, který bude současně vyvíjet middleware pro parsování soutěžních výsledků z XML feedu v rámci své bakalářské práce nad jejímž výstupem postavte část týkající se scoreboard.

Seznam odborné literatury:

[1] Roman Smetana. 2016. Next generation of Second-Screen. Realtime application MyICPC.

Vedoucí: Ing. Tomáš Černý, Ph.D.

Platnost zadání do konce letního semestru 2017/2018

L.S.

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 31.10.2016