СИСТЕМА ИНТЕРНЕТ-КОММУНИКАЦИЙ ДЛЯ ЖЮРИ И УЧАСТНИКОВ СОРЕВНОВАНИЙ

В. К. Марченко

ВВЕДЕНИЕ

При проведении олимпиад по программированию (в частности, заключительных этапов школьных олимпиад по информатике) организаторы сталкиваются с проблемой обеспечения связи между участниками соревнования и членами жюри. Эта проблема возникает из-за ограничений, предъявляемых форматом проведения олимпиады.

Олимпиада проходит в два тура, каждый длительностью по 5 часов. В течение этого времени у участников могут возникать вопросы, а у членов жюри – важные организационные объявления и пояснения к условиям задач. Кроме этого, количество всех участников превышает несколько десятков, что ведет к необходимости размещения участников по различным компьютерным классам. Это усложняет коммуникацию, так как для сообщения информации требуется обойти все кабинеты с участниками.

Согласно регламенту, все общение между участниками и членами жюри должно записываться и сохраняться для того, чтобы была возможность просмотреть вопросы участников, а также сообщения и ответы жюри в случае подачи апелляции. Чтобы исключить возможность подсказок, участникам запрещено общаться друг с другом, а члены жюри должны отвечать на вопросы, используя только термины «да», «нет» и «без комментариев». Однако, если вопрос задан таким образом, что этих терминов недостаточно, член жюри может послать сообщение с произвольным текстом отдельному участнику.

Проблему коммуникации можно решать разными способами. Первый способ заключается в том, чтобы привлечь третьих лиц, которые будут служить посредниками между жюри и участниками. В этом случае вопросы участников и ответы жюри на вопросы будут сообщаться на листе бумаги. Альтернативный способ – создание приложения, которое будет выполнять эту работу автоматически.

ОПИСАНИЕ ФУНКЦИОНАЛА

При создании программы выбор пал на web-приложение, которое упрощает разработку клиентской части, а также позволяет в дальнейшем

развернуть приложение в Интернете, используя облачные сервисы. Пользователи сайта взаимодействуют друг с другом через web-браузер, который установлен на рабочем месте. На одном из компьютеров установлена база данных и сервер, который принимает и отправляет HTTP сообщения.

Перед началом соревнования для каждого участника и члена жюри генерируются логины и пароли. Авторизация выполняется с использованием этих паролей. Каждая роль (участник либо член жюри) обладает своими собственными привилегиями, согласно которым пользователь может читать и отправлять сообщения разных типов.

Разработанное приложение позволяет отправлять сообщения четырех типов:

1. Сообщение от члена жюри ко всем участникам.
2. Вопрос от участника к одному из членов жюри.
3. Ответ жюри на вопрос участника.
4. Сообщения от члена жюри к конкретному участнику.

Первый тип сообщения отвечает за объявления участникам информации, связанной с условиями задач, организационными вопросами (например, сообщение о начале и окончании соревнования). С помощью второго типа сообщения участники задают вопросы членам жюри. Ответ на вопросы (третий тип сообщений) может быть сформулирован в одном из трёх вариантов: «да», «нет» и «без комментариев». Для исключительных случаев предусмотрен еще один тип сообщения, через который член жюри может отослать сообщение конкретному участнику (например, с замечанием). Эти ограничения на типы сообщения накладываются исходя из регламента соревнований, чтобы участники не могли общаться между собой.

Наряду с ограничениями на отправление сообщений, в системе накладываются ограничения на чтение сообщений. Если жюри видит все сообщения в системе, то участники видят только те, которые они написали или получили. То есть участник не может видеть вопросы, ответы и замечания другого участника.

Все сообщения в системе при отправлении записываются в базу данных и доступны для чтения всем членам жюри. Удалить сообщение через web-приложение невозможно, это сделано для того, чтобы сообщения оставались в базе данных до возможной апелляции. У каждого сообщения в момент его создания устанавливается идентификатор автора, который определяется из сессии авторизованного пользователя.

АРХИТЕКТУРА

Клиентская часть состоит из браузера и html–страниц. Браузер создает HTTP–запросы и отправляет их на сервер. Сервер получает HTTP–запрос, обрабатывает его и отправляет HTTP–ответ клиенту. Ответ может содержать html–страницу, на которую браузер выполнит переход, или другую информацию, в зависимости от логики программы.

Серверная часть web-приложения устроена сложнее. При проектировании выбрана многослойная архитектура, состоящая из уровней контроллера, сервиса и слоя DAO. Многослойная архитектура позволяет инкапсулировать логику компонентов и уменьшить зависимость компонентов друг от друга. Уровень контроллера работает только с классами сервиса, сервис работает только с уровнем DAO. В качестве параметров и возвращаемых значений функций, доступных для вызова из другого слоя приложения, часто используются примитивные типы и domain-объекты. Domain-объекты хранят в себе данные, используются во всех слоях приложения и предназначены для передачи данных между слоями. Рассмотрим многослойную архитектуру более подробно.

Уровень контроллера отвечает за обработку клиентских HTTP-запросов и создание HTTP-ответа. Запросы клиента обрабатываются следующим образом:

- из HTTP-запроса извлекается тип команды, которую вызвал пользователь;
- определяется роль пользователя и выполняется проверка его привилегий на выполнение данной команды;
- из HTTP-запроса извлекаются необходимые данные и создаются domain объекты;
- вызывается бизнес-логика из слоя сервиса, соответствующая данной команде;
- создается HTTP-ответ на запрос, который может содержать в себе данные MIME типа, начиная с html-страницы, заканчивая mp4 видео.

Слой сервиса отвечает за реализацию бизнес-логики приложения. Под бизнес-логикой понимают основную логику приложения, которая отвечает за выполнение требований, наложенных на систему, и взаимодействие domain-объектов между собой. Примером такой логики может служить создание и рассылка отчета по электронной почте, анализ данных, создание одних domain-объектов на основе других. Слой сервиса работает с хранилищем данных через слой DAO.

Слой DAO ответственен за работу с хранилищем данных и инкапсулирует работу с ним. В качестве хранилища могут выступать различные виды баз данных и файлов. Основная задача слоя – предоставление ме-

тодов для работы с хранилищем данных. При изменении типа хранилища требуется переписать только реализацию этих методов, методы уровня сервиса изменять не требуется.

ТЕХНИЧЕСКИЕ ДЕТАЛИ

Клиентская часть поддерживает браузеры Google Chrome и Internet Explorer. Пользователи взаимодействуют с html страницами. Для дизайна использовались такие возможности, как CSS и Javascript, фреймворк bootstrap.

Серверная часть базируется на технологиях Servlet API и JSP API. В качестве HTTP-сервера и контейнера сервлетов использовался Apache Tomcat. Этот контейнер является сегодня одним из наиболее часто используемых. Его исходный код является открытым и сам продукт может использоваться бесплатно, согласно лицензии Apache License version 2. Сам сервер является легковесным, и его запуск сервера выполняется быстро.

В качестве СУБД использовалась MySql 5.6, так как данная база данных является одной из популярных, и поиск информации о ней в интернете не представляет трудностей. Для работы с ней использовалось JDBC API и библиотека mysql-connector-java.

Для сборки проекта использовался декларативный сборщик Maven. Среди плюсов данного сборщика стоит отметить упрощенный контроль зависимостей в проекте, а также возможность скачивать библиотеки сразу при собирании проекта с удаленных и локальных репозиториев. Так как сборщик является декларативным, этапы сборки проекта уже созданы, и их требуется всего лишь кастомизировать, что для небольших и тривиальных проектов делается просто и быстро.

Описанные технологии и архитектура позволили создать web-приложение перед Минской городской олимпиадой по информатике (третьим этапом республиканской олимпиады) 2015/2016 учебного года в короткий срок. Во время проведения олимпиады приложение отработало без сбоев.