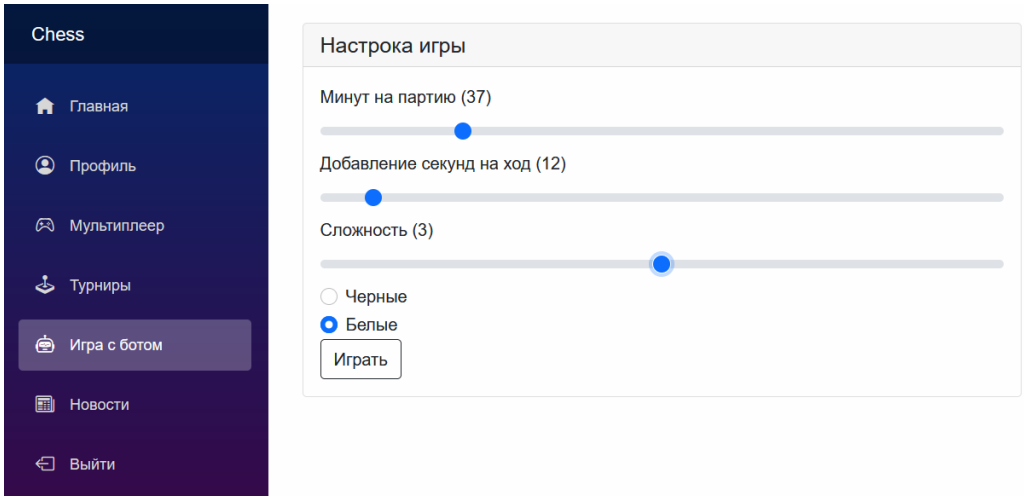

Рис. 4. Меню настройки игры

Разработанное приложение поможет шахматистам поддерживать свой уровень игры, а также позволит проводить турниры для всех шахматистов из любой страны мира. В дополнение к данному функционалу любой пользователь может прочитать последние новости по ходу проведения турниров.

ПРИМЕНЕНИЕ BLAZOR В РАЗРАБОТКЕ WEB-ПРИЛОЖЕНИЙ

Е. А. Усачёв

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель Т. Л. Романькова

Любой сайт на сегодняшний момент состоит из некоторого количества *HTML*-страниц. Когда пользователь нажимает на кнопку или любой другой элемент интерфейса, представляющий ссылку, браузер загружает полностью новые страницы, изображая переход от одной страницы к другой. Страница может быть в виде файла на сервере или сгенерированной (используя страничные файлы на сервере), но так или иначе генерация целой страницы на сервере занимает какое-то количество времени. Данная ситуация изменилась после 2010 г. Появилась новая архитектура, при которой получилось нечто среднее между приложением и сайтом, названное *SPA* (англ. Single Page Application).

Большинство приложений на сегодняшний день в той или иной степени являются *SPA*-приложениями. *SPA*-приложение – это веб-приложение, которое выполняется на стороне клиента, использующее один *HTML* документ как стартовую страницу для всех других веб-страниц и организующий взаимодействие с пользователем посредством подгружаемых *HTML*, *CSS* и *JavaScript*. *SPA*-приложения работают следующим образом:

– при открытии пользователем веб-страницы браузер загружает весь код приложения, выводя для отображения только ту часть, которую запросил пользователь;

– при переходе на другую страницу пользователю выводятся ранее загруженные данные вместе с новыми подгруженными с сервера данными (по необходимости).

Согласно статистике, большая часть пользователей не ждут загрузки страницы,

если время составляет более двух секунд. И подход с *SPA*-приложениями становится предпочтительным для решения данной задачи, поскольку браузер отображает выбранное пользователем состояние страницы, изменяя лишь ее содержимое, но не ее саму.

Исходя из вышеперечисленного, выделим плюсы и минусы *SPA*-приложений. К плюсам можно отнести:

– быстрый интерфейс, повышающий вероятность совершения целевых действий;
– возможность использования для сайтов практически любого предназначения;
– меньший объем трафика за счет отправки только полезных данных;
– возможность работы в автономном режиме, без подключения к Интернету;
– меньшая нагрузка на сервер.

К недостаткам *SPA* относятся:

– медленная загрузка начальной страницы;
– проблематичная индексация страниц для поисковых систем (*Seo*);
– повышенная нагрузка на браузер клиента;
– невозможность работы при отключенной поддержке *JS*.

При проектировании приложения встал вопрос выбора архитектуры. Поскольку страниц в разрабатываемом приложении не так много, а само приложение имеет достаточно много интерактивных элементов, в качестве клиентской библиотеки был выбрана технология *SPA*.

На рис. 1 представлена схема работы *SPA*-приложений.

Рис. 1. Сравнительная схема *SPA*- и *PWA*-приложений

В качестве фреймворка клиентского приложения для шахматного портала был выбран *Blazor*. Существует две версии реализации фреймворка:

– *Blazor Server* (*MPA*);

– *Blazor WebAssembly* (*SPA*).

Для данного проекта была выбрана реализация *SPA*. Данная реализация основана на технологии *WebAssembly*. *WebAssembly* представляет собой среду выполнения непосредственно в браузере. Схема работы для реализации *.NET* следующая:

– файлы с *C#* кодом компилируются в сборки *.NET*;

– посредством взаимодействия с *JavaScript* сборки загружаются в *WebAssembly*;

– последующие выполнение происходит уже непосредственно в сам *WebAssembly*. Также для обновления *DOM* и некоторых других действий фреймворк обращается к *API* браузера. Функциональная схема работы *Blazor* представлена на рис. 2.

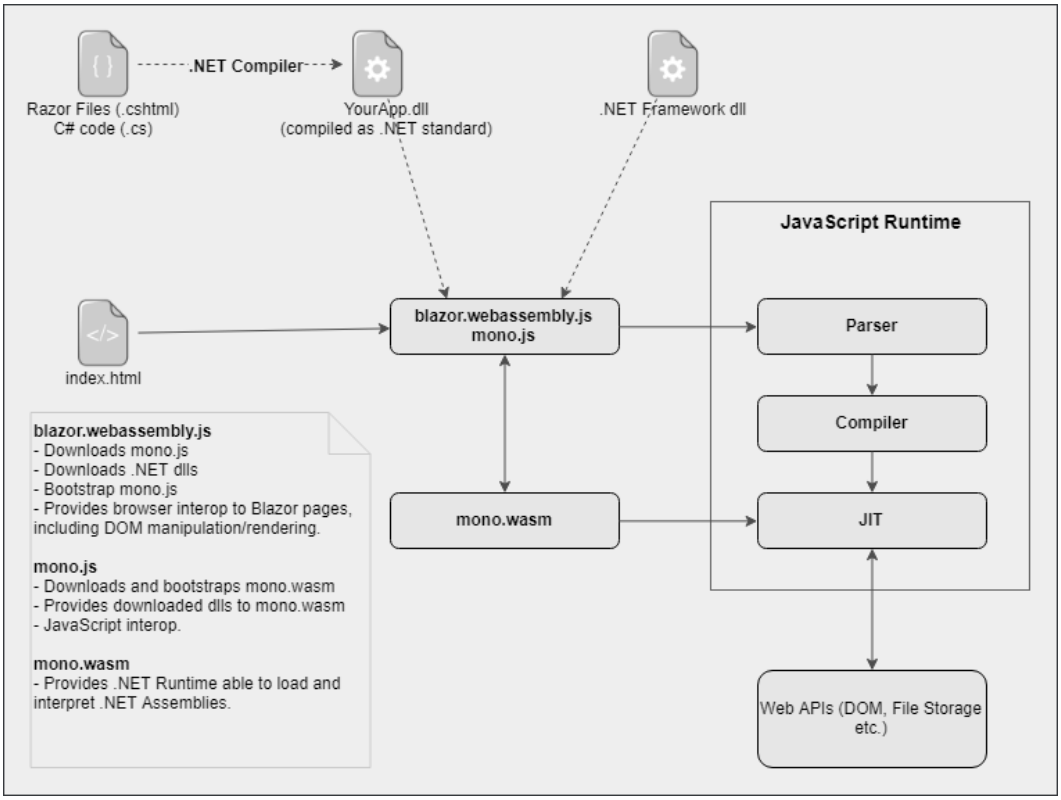

Рис. 2. Схема загрузки и работы *Blazor*

Все вышеперечисленные особенности хорошо подходят для шахматного портала, поскольку страниц для отображения не очень много, а следовательно, можно максимально переиспользовать компоненты. Поскольку реализация *SPA* выполнена через *WebAssembly*, для написания серверной и клиентской частей используется один и тот же язык, что значительно ускоряет разработку приложения, а также является достаточно удобным. Также однообразие языка позволяет для клиентской и серверной части использовать одну и ту же логику. Компонентная структура позволяет уменьшить размер загружаемых библиотек. Вместе с тем остается возможность использовать полноценно *JavaScript* библиотеки, что необходимо в случаях, когда библиотека слишком большая или ее портирование под *WebAssembly* займет слиш-

ком много времени. И самым главным преимуществом является то, что на основе данной технологии в будущем можно сделать полноценное десктопное приложение, приложение для мобильных устройств и другие платформы с сохранением большинства кода в неизменности. Такие приложения называются *PWA* (англ. *Progressive Web Applications*).

Таким образом, технология *Blazor* оптимальна для данного приложения, поскольку является кроссплатформенной, запускается и работает в браузере (вследствие чего не требует установки на компьютер пользователя), позволяет создавать самые разнообразные интерфейсы, а также писать клиентские приложения на том же языке, что и серверные. Помимо вышеперечисленных преимуществ данная технология позволяет в будущем портировать приложения практически на любую операционную систему.

ПРИМЕНЕНИЕ ОБЛАЧНЫХ ТЕХНОЛОГИЙ В РАЗРАБОТКЕ WEB-ПРИЛОЖЕНИЯ ДЛЯ НАЧИНАЮЩИХ МУЗЫКАНТОВ

А. Н. Малецкий

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель Т. Л. Романькова

В настоящее время большинство web-сайтов являются динамическими и активно используют *JavaScript*, с помощью которого загружают необходимые данные для своей работы. Web-приложение для начинающих музыкантов не стало исключением, ведь оно должно позволять управлять контентом, загружать и проигрывать аудио, обеспечивать возможность чата между продюсерами и исполнителями и выполнять множество других задач. В этом случае явно нельзя обойтись без серверной части приложения. В классическом варианте возможно использование сервера, который запущен и работает, пока не будет остановлен. В противовес этому варианту возможно применение *serverless*.

Бессерверная архитектура обеспечивает четкое разделение между кодом и его средой размещения. Вы реализуете код в функции, которая вызывается триггером. После завершения работы этой функции ее ресурсы могут быть освобождены. Триггер может быть *HTTP*-запросом, добавлением сообщения в очередь, планировщиком и др. Результатом срабатывания триггера является выполнение кода.

Бессерверная архитектура – это архитектура, которая в значительной степени зависит от абстрагирования среды хоста, чтобы сосредоточиться на коде, поэтому это можно рассматривать как подход «без сервера».

Абстракция означает, что не нужно управлять серверами и конкретными контейнерами. Бессерверная платформа размещает код либо в виде скриптов, либо в виде исполняемых файлов, и выделяет необходимые ресурсы для масштабирования кода.

Преимущества бессерверной архитектуры:

– высокая плотность. Многие экземпляры одного и того же бессерверного кода могут выполняться на одном и том же хосте;

– оплата по факту использования;

– значительная экономия средств в определенных сценариях;

– мгновенное масштабирование. Бессерверная система может автоматически и быстро масштабироваться в соответствии с рабочими нагрузками;