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;

– более быстрое время выхода конечного приложения на рынок. Разработчики сосредотачиваются на коде и развертывают его непосредственно на бессерверной платформе. Компоненты могут быть выпущены независимо друг от друга.

Бессерверные технологии чаще всего упоминается в контексте вычислений, но они также могут применяться и к данным.

Например, сервисы *DynamoDB* (*AWS*) и *CosmosDB* (*Azure*) предоставляют облачные базы данных, которые не требуют конфигурации хост-машины или кластера БД.

Рассмотрим применение бессерверных технологий на инфраструктурной схеме приложения. Архитектура web-приложения дана на рис. 1.

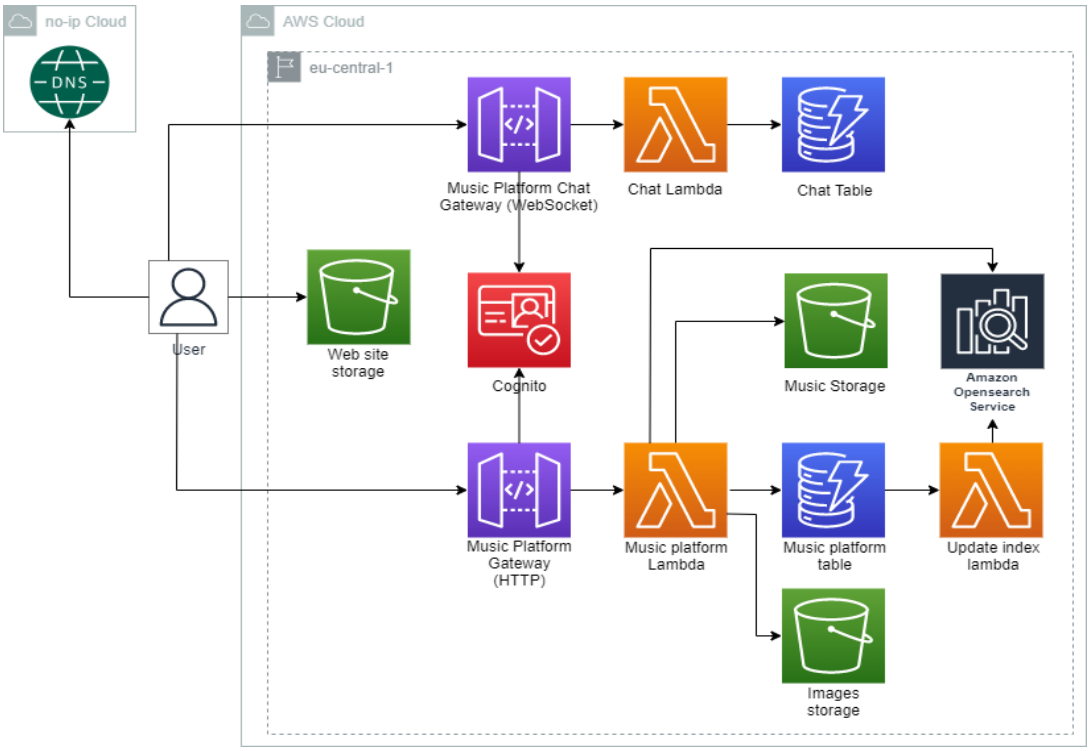

Рис. 1. Архитектура web-приложения

Архитектура не идеальна, так как большая часть функционала сконцентрирована в одной лямбда-функции «*Music-platform-lambda*», но код имеет хорошее логическое разделение, и она может легко быть разбита на несколько более специализированных функций.

Из схемы на рис. 1 видно, что использовались сервисы облачного провайдера *Amazon,* в частности, сервис *AWS Lambda*. *AWS Lambda* – это сервис бессерверных вычислений, который запускает программный код в ответ на определенные события, такие, как *HTTP*-запросы через *Amazon API Gateway*, изменение объектов в корзинах *Amazon Simple Storage Service* (*Amazon S3*), обновление таблиц в *Amazon DynamoDB* или смена состояний в *AWS Step Function*, и отвечает за автоматическое выделение необходимых вычислительных ресурсов.

В данном случае функцию приводит в действие *HTTP API Gateway*, который, получив запрос от пользователя, ищет функцию, которая подходит для обработки запроса (на основании адреса запроса), преобразует запрос в понятный для функции

набор параметров, вызывает функцию, получает результат выполнения, преобразует его в *HTTP*-ответ и, наконец, отправляет его пользователю. Единственным минусом применения лямбда-функций в данном варианте использования является время так называемого «холодного старта». При запуске бессерверной функции она будет оставаться активной (т. е «горячей») до тех пор, пока она часто вызывается. После периода бездействия облачный провайдер удаляет контейнер, и функция становится неактивной (т. е. «холодной»).

Холодный запуск происходит, когда запускается неактивная функция. Задержка возникает из-за того, что облачному провайдеру необходимо некоторое время на подготовку контейнера функции и на сам вызов.

При разработке также был применен сервис *AWS Cognito*. Этот сервис позволил быстро и просто добавить возможность регистрации, авторизации и контроля доступа пользователей. *Amazon Cognito* масштабируется до миллионов пользователей. Сервис также предоставляет встроенный настраиваемый интерфейс для регистрации и авторизации пользователей.

Также была использована облачная база данных *DynamoDB*. *Amazon DynamoDB* – это полностью управляемая бессерверная база данных *NoSQL*, которая поддерживает пары «ключ – значение» и документные модели данных.

С помощью *DynamoDB* разработчики могут создавать современные бессерверные приложения с возможностью глобального масштабирования и поддержкой петабайтов данных и десятков миллионов операций чтения и записи в секунду.

Каждая запись таблицы *DynamoDB* должна иметь свой уникальный ключ. Он может быть простым (только *hash key*) или составным (*hash key* + r*ange key*). *Hash key* иногда могут называть *partition key*, а *range key* может встретиться с названием *sort key*. Получение записей можно осуществлять по ключу или индексу. Быстрый поиск – по *hash key* и дополнительному условию по *range key*.

Таблицы поддерживают 2 типа индексов: *local secondary index* (*LSI*) и *global secondary index* (*GSI*). Индексы первого типа нельзя удалить после добавления, они применимы только к таблицам с составным ключом, их *hash key* должен совпадать с *hash key* основной таблицы.

В одной таблице может быть всего 5 индексов этого типа. *GSI* выгодно отличаются от *LSI* тем, что их можно добавить и удалить в любое время, их *hash key* необязательно должен совпадать с *hash key* основной таблицы, и, наконец, в одной таблице может быть до 25 индексов этого типа.

При разработке схемы было решено применить рекомендуемый сервисом однотабличный подход, используя при этом несколько *GSI*.

Пример данных, которые хранятся в таблице «*Music platform table*», показан на рис. 2.

Рассмотрим пример используемого индекса. Вторичный индекс под названием «*owner_resources*» состоит из двух атрибутов: *owner_username* был выбран в качестве *hash key*, а *metadata* в качестве *range key*. По этому индексу можно найти все записи, принадлежащие пользователю, по его имени пользователя.

Primary key		Attributes								
Partition key: id	Sort key: metadata									
PLAYLIST#1	PLAYLIST#1	name	owner	modifiedTS	type	tracks	owner_username			
		My favorite playlist	Aliaksandr Maletski	2022-02-12T09:53:36Z	PLAYLIST	["TRACK#1","TRACK#2"]	username			
COLLECTION#1	COLLECTION#1	name	owner	modifiedTS	type	playlists	owner_username			
		Time Machine	Music Service	2022-02-12T09:53:36Z	COLLECTION	["PLAYLIST#1","PLAYLIST#2"]	username			
ALBUM#1	ALBUM#1	name	artists	owner	image	genre	modifiedTS	likes	public	type
		Some album	["artist_username"]	Flo Rida	ALBUM#1.png	rap	2022-02-12T09:53:36Z	2500	true	ALBUM
	TRACK#51	name	downloads	artists	modifiedTS	duration	likes	public	type	
		Some track	5000	["artist_username"]	2022-02-12T09:53:36Z	125	105	false	TRACK	
	TRACK#510	name	downloads	artists	modifiedTS	duration	likes	public	type	
		Some track too	10000	["artist1_username","artist2_username"]	2022-02-12T09:53:36Z	320	999	true	TRACK	
LIKE#username	ALBUM#1	modifiedTS	type							
		2022-02-12T09:53:36Z	LIKE							
	COLLECTION#1	modifiedTS	type							
		2022-02-12T09:53:36Z	LIKE							

Рис. 2. Пример записей таблицы «*Music platform table*»

Таким образом, при разработке web-приложения для начинающих музыкантов были использованы облачные технологии, которые позволили существенно сократить время на разработку, платить только за использованные ресурсы и делегировать все заботы о размещении и выполнении кода облачному провайдеру.

МОДЕЛИРОВАНИЕ ТЕЧЕНИЯ ЖИДКОСТИ В ГИДРАВЛИЧЕСКИХ УСТРОЙСТВАХ В ПРИЛОЖЕНИИ KompasFlow

А. В. Ковалев

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

Научный руководитель Ю. А. Андреевец

Система КОМПАС-3D предназначена для создания трехмерных ассоциативных моделей отдельных деталей и сборочных единиц, содержащих как оригинальные, так и стандартизованные конструктивные элементы. Параметрическая технология позволяет быстро получать модели типовых изделий на основе проектированного ранее прототипа. Многочисленные сервисные функции облегчают решение вспомогательных задач проектирования и обслуживания производства [1].

Приложение KompasFlow представляет собой интегрированный в КОМПАС-3D инструмент экспресс-анализа аэрогидродинамики проектируемого устройства.

KompasFlow обладает простым интерфейсом для экспресс-анализа устройства на ранних этапах его проектирования и позволяет сделать первичную оценку влияния вносимых изменений в геометрию устройства на его эффективность [2].

Рассмотрим течение жидкости на примере течения жидкости в гидродросселе с обратным клапаном типа ДКТ20/3. Технические характеристики дросселя с обрат-