ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

УДК 004.744.056 **doi: 10.32620/reks.2020.2.07**

В. В. ФРОЛОВ, А. А. ОРЕХОВ, В. С. ХАРЧЕНКО, А. В. ФРОЛОВ

Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ»

АНАЛИЗ ВАРИАНТОВ ДВУХВЕРСИОННЫХ МНОГОМОДУЛЬНЫХ ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНЫХ СЕРВИСОВ

Статья посвящена анализу вариантов двухверсионных многомодульных веб-приложений с использованием облачных сервисов. Поскольку проектирование и разработка веб-приложений ведется все более активно, возникает необходимость повышать их надежность в условиях повышения сложности самих приложений и инфраструктуры на которых они базируются. Одним из ключевых решений данной проблемы является использование облачных сервисов, которые позволяют значительно упростить задачу обеспечения надежности и безопасности различных приложений. При этом облачные провайдеры не могут полностью гарантировать отказоустойчивость приложений, которые запускаются в их среде. Таким образом пользователи должны сами побеспокоиться об этом. Одним из наиболее перспективных подходов является применения диверсности для повышения безопасности и надежности веб-приложений, размещенных в облаках. ***Объектом*** *исследования и анализа данной работы являются многомодульные веб-приложение, спроектированные при помощи облачных сервисов.* ***Целью*** *исследования данной работы является анализ и разработка вариантов структур двухверсионных многомодульных веб-приложений с использованием облачных сервисов. Так как множество компаний переносят свою инфраструктуру в облака, то возникает необходимость рассмотреть возможность применения диверсности при помощи облачных сервисов. Они позволяют создать и развернуть веб-приложения, разработанные на различных языках программирования на серверах облачных провайдеров. Таким образом часть ответственности за обеспечения надежности перекладывается на них. Однако необходимо по-прежнему обеспечивать отказоустойчивость собственных программ, которые могут отказать по причине дефектов в программном коде. Одним из основных решений этой проблемы является N-версионное программирование, которое позволяет создавать приложение из нескольких независимых версий. Каждая версия может быть написана на разных язык программирования и с использованием различных технологий отдельными командами разработчиков, тем самым повышая надежность конечного программного продукта.* ***Как результат,*** *в данной работе делается заключение о том, что ведущие облачные провайдеры предоставляют возможность реализовать диверсность на практике при помощи сервисов различных моделей представления, таких как IaaS и PaaS. Используя принцип диверсности можно спроектировать надежное веб-приложение, которое позволит избежать его отказа в случае ошибки в программном коде.*

Ключевые слова: *облачные сервисы; диверсность веб-приложений; облачные провайдеры; модель предоставления облачных сервисов; отказ по общей причине; N-версионное программирование; обеспечение надежности и безопасности программного продукта; двухверсионное приложение.*

Введение

На сегодняшний день разработка веб-приложений с использованием облачных сервисов набирает все большую популярность. В таких условиях необходимо обеспечивать безопасность и надежность работы веб-приложений, учитывая специфику облачных сервисов. Несмотря на то, что облачные провайдеры берут на себя ответственность за охрану серверов и центров обработки данных, а также за своевременное обновление необходимого программного обеспечения, никто не гарантирует безопасность приложений, которые запускаются с использованием их сервисов [1-3]. А значит по-прежнему существует необходимость понижать риск отказа программного обеспечения по причине дефектов в программном коде. Для решения этой задачи широко известно ряд средств, одним из которых является использования принципа диверсности при проектировании программ.

Для повышения надежности веб-приложений целесообразно использовать N-версионное программирование, которое основывается на проектировании приложения, которое состоит из различных версий, написанных на различных языках программирования [4-6]. Реализовать это на веб-приложениях можно при помощи балансировки нагрузки между различными версиями, которые функционально выполняют одну и ту же задачу. Таким образом, используя облачные сервисы и принцип диверсности можно разработать более надежное веб-приложение.

В статье [7] рассматриваются три возможных

распределенных решения, предлагаемых для балансировки нагрузки. Анализ модели на основе балансировки нагрузки для динамического выбора веб-службы, основанный на соответствующей архитектуре, наряду с функциональными требованиями с учетом балансировки нагрузки на сервис, представлен здесь [8]. В работе [9] предлагается кластерная система веб-сервера, организованная в виде нескольких логических кольцевых соединений, в которой DNS интегрирован с алгоритмом адаптивной балансировки нагрузки. Авторы статьи [10] предоставляют набор критериев оценки для оценки и сравнения различных языков спецификации шаблонов проектирования, чтобы помочь практикам и исследователям выбрать подходящий язык для их использования. В работе [11] рассматривается и сравнивается производительность различных механизмов безопасности, применяемых к простому веб-сервису, протестированному с различными исходными размерами сообщений. В статье [12] рассматривается безопасность современных веб-приложений и перечислим наиболее распространенные атаки на них, такие как внедрение, межсайтовый скриптинг и небезопасные прямые ссылки на объекты. Авторы в работе [13] уделяют большое внимание принципу создания визуальных и запоминающихся веб-приложений, но с учетом потенциальной уязвимости большого количества конфиденциальных и пользовательских данных.

Целью данной статьи является анализ и разработка вариантов структур двухверсионных многомодульных веб-приложений с использованием облачных сервисов.

1. Анализ многоверсионных веб-приложений

Балансировка нагрузки относится к эффективному распределению входящего сетевого трафика по группе внутренних серверов, также известной как ферма серверов или пул серверов. Современные веб-сайты с высоким трафиком должны обслуживать сотни тысяч, если не миллионы, одновременных запросов от пользователей или клиентов и возвращать правильный текст, изображения, видео или данные приложений, все быстро и надежно. Для экономически эффективного масштабирования в соответствии с этими большими объемами современные передовые технологии обычно требуют добавления большего количества серверов.

Балансировщик нагрузки действует как администратор, сидящий перед вашими серверами и направляющий клиентские запросы на сервер, способный выполнять эти запросы таким образом, чтобы максимально использовать скорость и емкость и гарантировать, что ни один сервер не перегружен, что может снизить производительность. Если один сервер выходит из строя, балансировщик нагрузки перенаправляет трафик на остальные онлайн-серверы. Когда новый сервер добавляется в группу серверов, балансировщик нагрузки автоматически начинает отправлять запросы на него.

Балансировщик нагрузки наиболее часто применяется для следующих типов траффика (протоколов): http, https, tcp и udp [14].

Для веб-приложений, которые используют http и https можно использовать nginx как очень эффективный балансировщик нагрузки и распределения трафика между несколькими серверами приложений и для повышения производительности, масштабируемости и надежности веб-приложений.

В nginx поддерживаются следующие механизмы балансировки нагрузки [15]:

– round-robin – запросы к серверам приложений распределяются в циклическом порядке

– наименьшее количество подключенный – следующий запрос назначается серверу с наименьшим количеством активных подключений;

– ip-hash – хеш-функция используется для определения, какой сервер следует выбрать для следующего запроса (на основе IP-адреса клиента).

Балансировка для веб-приложения может включать использования сразу нескольких языков, фреймворков и паттернов. То есть разные пользователи ресурса видят одинаковый контент ресурса, однако он подгружается с разных backend серверов, с абсолютно разной реализацией.

Подобный подход хорошо подходит для тестирования паттернов в реальных условиях. Паттерны обычно показывают отношения и взаимодействия между классами или объектами. Идея состоит в том, чтобы ускорить процесс разработки, предоставив хорошо проверенную, подтвержденную парадигму разработки/проектирования [16]. Внедрение паттернов проектирования в ваш проект не обязательно. Они предназначены не для разработки проектов, а для решения распространенных проблем. Всякий раз, когда есть необходимость, вы можете реализовать подходящий паттерн, чтобы избежать подобных проблем в будущем. Чтобы узнать, какой паттерн использовать, вы просто должны попытаться понять саму идею паттерна и его цель. Только после анализа нескольких, возможно самых популярных, вы сможете выбрать правильный.

Существует несколько классификаций паттернов проектирования, в зависимости от разных критериев и от разных авторов. Однако самая популярная из них делит их на три основных категории [17]:

– порождающие (creational);
– поведенческие (behavioral);
– структурные (structural).

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

2. Проектирование диверсного веб-приложения

Было осуществлено проектирование диверсного веб-приложения, реализованного по принципам N-версионности, где все версии разработаны на различных языках программирования. Каждая версия представляет из себя многомодульное приложение, где модуль является реализацией паттерна проектирования.

В состав каждой версии входит 3 модуля, поскольку целесообразно использовать по одному паттерну проектирования каждого вида (порождающие, поведенческие и структурные). В результате чего было получено N независимых приложений, которые реализуют одинаковый функционал различными средствами, которые подключаются к одной базе данных.

После чего выполняется их объединение в одно веб-приложение и настройка балансирования нагрузки между версиями. Как результат, конечные пользователи получают единый интерфейс вне зависимости от того, какая версия приложения обрабатывала их запрос. Таким образом повышается надежность программного продукта, так как отказ одной версии никак не влияет на работу остальных версий.

Исходя из востребованности языков программирования, которые подходят для реализации поставленной задачи были выбраны следующие языки: Java, C#, Python, NodeJS, PHP и Ruby. Все они имеют свои особенности и разные области применения, однако их можно использовать для проектирования веб-приложения. В дальнейшем было проанализировано, поддержку каких языков предоставляют облачные сервисы модели PaaS (табл. 1) и в соответствии с этим было принято окончательное решение.

Таблица 1

Сравнение технологий облачных провайдеров для веб-сервисов

Технология	Провайдер		
	Azure	GCP	AWS
.NET	+	+	+
Java	+	+	+
NodeJS	+	+	+
PHP	+	+	+
Python	+	+	+
Ruby	+	+	+
Go	-	+	+
ASP.NET	+	-	-

В дальнейшем необходимо выбрать какие паттерны проектирования лягут в основу модулей приложения. Так как необходимо 3 паттерна различных видов, было решено сделать выбор случайным образом.

В результате получилось следующее:

- строитель (порождающий);
- наблюдатель (поведенческий);
- мост (структурный).

Таким образом, структура каждой версии имеет следующий вид (рис. 1).

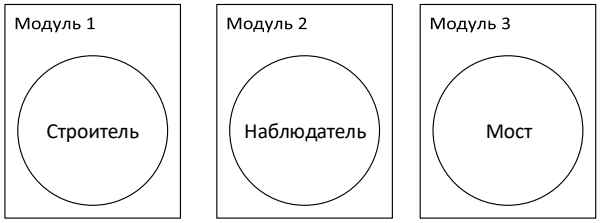

Рис. 1. Структура версии

Каждая версия приложения реализована на разных языках, для каждого из которого необходимо дополнительные программные средства, такие как фреймворки, менеджеры пакетов и зависимостей и другие. А также необходимо использовать коннектор к базе данных. Дополнительно, возможно, использование веб-сервера и сервера приложения. Были проанализированы требования для каждого языка и выбраны наиболее популярные решения.

Существует большое количество различных фреймворков, которые помогают в разработке веб-приложения на языке Java, однако самым популярными являются Spring, Hibernate, Google Web Toolkit и другие. В качестве менеджера зависимостей и пакетов обычно используют Maven, Gradle или Ant. Для данного эксперимента была выбрано следующее (рис. 2).

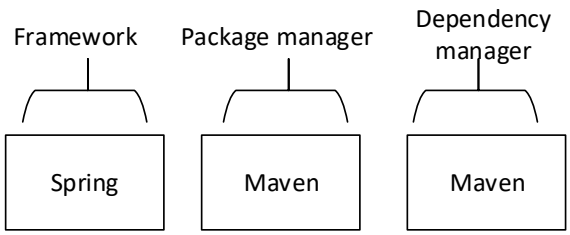

Рис. 2. Структура программных средств для Java

Компания Microsoft предлагает разрабатывать веб-приложения на языке C# при помощи их технологии ASP.NET. Так же менеджером пакетов и зависимостей рекомендовано использовать NuGet (рис. 3).

PHP разрабатывался специально для веб-приложений, таким образом нет необходимости использовать дополнительные программные средства. Однако

для удобства и получения большей функциональности рекомендуется использовать современные фреймворки, такие как, Laravel, CakePHP, Symfony и другие. Также основным менеджером пакетов и зависимостей является Composer, без которого сложно поддерживать большие веб-приложения (рис. 4).

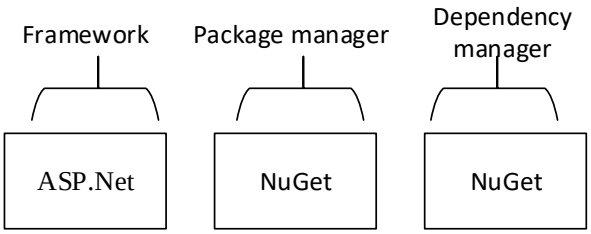

Рис. 3. Структура программных средств для C#

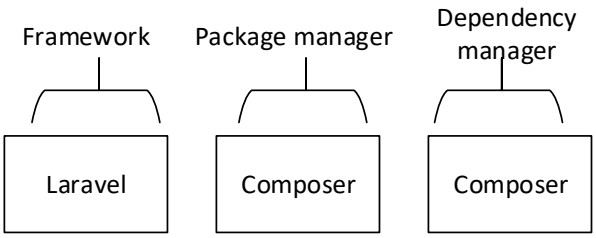

Рис. 4. Структура программных средств для PHP

Скриптовый язык Python подходит для разработки приложения практически любого вида. Так же его можно использовать и для проектирования динамического контента веб-сайта. Однако для упрощения процесса разработки эффективно использовать различные фреймворки, такие как Django, TurboGears, web2py и другие. В качестве пакетного менеджера по умолчанию используется pip, при этом можно дополнительно управлять зависимостями с помощью pipenv (рис. 5).

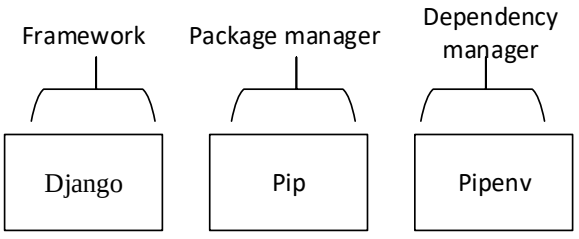

Рис. 5. Структура программных средств для Python

Язык Ruby является универсальным и подходит для решения многих задач. Для облегчения веб-разработки существует ряд фреймворков таких как Ruby on Rails, Sinatra, Padrino и множество других. Общепринято использовать связку Gem и Bundle для управления пакетами и зависимостями, а также Rake для запуска различных задач (рис. 6).

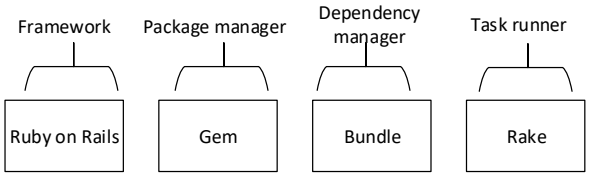

Рис. 6. Структура программных средств для Ruby

Технология NodeJS позволяет выполнять JavaScript код на стороне сервера, таким образом с ее помощью можно написать многофункциональное приложение, не уступающее популярным языкам программирования. Существует множество фреймворков помогающие решить задачу построения веб-приложения, самыми популярными из которых являются Express, Meteor. В качестве менеджера пакетов и зависимостей выступает npm (рис. 7).

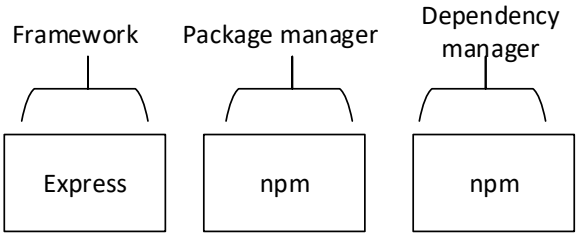

Рис. 7. Структура программных средств для NodeJS

Был проведен анализ существующих коннекторов к базам данных. По итогам которого был получен список баз данных доступный для конкретного языка программирования, который поддерживается фреймворком, используемый в процессе разработки. В результате чего было принято решение использовать базу данных MySQL, так как она поддерживается всеми необходимыми инструментами, используемыми при проектировании веб-приложения. Результат анализ представлен на рисунке 8.

3. Модели представления облачных сервисов (IaaS, PaaS)

Реализация веб-приложения может быть осуществлена с помощью облачных сервисов, которые предоставляются с помощью различных моделей предоставления облачных сервисов. Основными моделями для реализации веб-приложения являются:

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

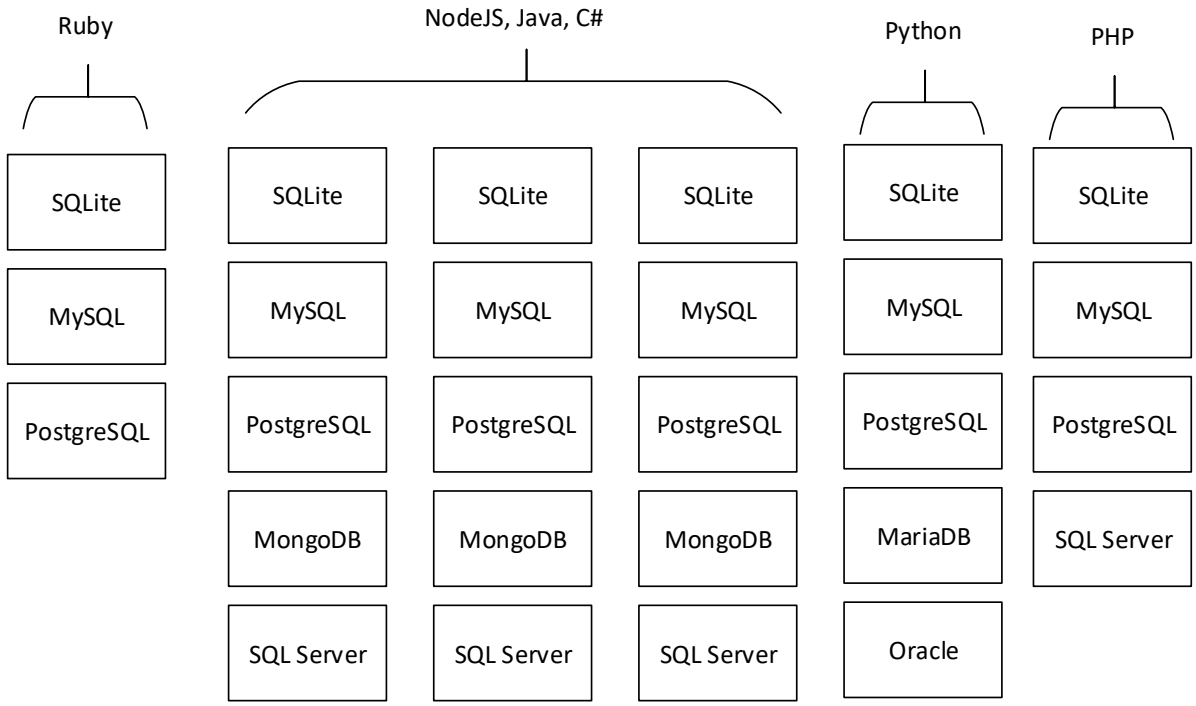

Рис. 8. Доступные базы данных для каждого языка программирования

– Инфраструктура как сервис (IaaS) – модель, в которой облачные провайдеры предоставляют вычислительные ресурсы, хранилища данных и сети в качестве интернет сервисов. Эта сервисная модель основана на технологии виртуализации. Amazon EC2, Compute engine от Google cloud, Virtual machine от Microsoft Azure являются наиболее известными сервисами модели IaaS;

– Платформа как сервис (PaaS) – модель, в которой облачные провайдеры предоставляют платформы, инструменты и другие бизнес сервисы, которые позволяют пользователям разрабатывать, разворачивать и управлять их собственными приложениями, при этом не требуется устанавливать эти платформы и заниматься поддержкой инструментов на своих компьютерах. Модель PaaS может быть расположена на верхнем уровне модели IaaS или непосредственно над облачными инфраструктурами. При этом Google App Engine, Microsoft Azure Web App и AWS Elastic Beanstalk являются наиболее известными представителями модели PaaS [18].

Для сравнения решений, которые предлагаются моделями IaaS и PaaS было принято решение использовать сервисы Microsoft Azure, как одного из самых популярных облачных провайдеров. Поскольку, в зависимости от региона расположения центров обработки данных, условия по предоставлению сервисов отличаются – необходимо сравнивать сервисы в одном регионе. Для примера был выбран регион Central USA. Результаты сравнения представлены в таблице 2 и 3 [19].

Таблица 2

Основные предложения по предоставлению виртуальных машин

Тип	ОЗУ, Гб	ЦП	Объем памяти, Гб	Количество дисков, шт.	Поддержка премиум дисков	Цена, доллар/месяц
A1_v2	2	1	10	2	Нет	31.39
A2_v2	4	2	20	4	Нет	66.43
A2m_v2	16	2	20	4	Нет	94.17
A4_v2	8	4	40	8	Нет	139.43
B1ls	0.5	1	4	2	Да	4.56
B1ms	1	1	4	2	Да	9.13

Продолжение табл. 2

Тип	ОЗУ, Гб	ЦП	Объем памяти, Гб	Количество дисков, шт.	Поддержка премиум дисков	Цена, доллар/месяц
B1s	2	1	4	2	Да	18.25
B2ms	4	2	8	4	Да	36.43
B2s	8	2	16	4	Да	72.85
B4ms	16	4	32	8	Да	146.00

Таблица 3

Основные предложение по предоставлению веб-сервисов

Тип	ОЗУ, Гб	Объем памяти, Гб	ЦП, ACU	Масштабируемость, шт.	Кол-во слотов	Кол-во ежедневных бэкапов, шт.	Цена, доллар/месяц
F1	1	1	-	-	-	-	Бесплатно**
B1	1.75	10	100	M* 3	-	-	13.14
B2	3.5	10	200	M* 3	-	-	25.55
B3	7	10	400	M* 3	-	-	51.10
S1	1.75	50	100	A* 10	5	10	69.35
S2	3.5	50	200	A* 10	5	10	138.70
S3	7	50	400	A* 10	5	10	277.40
P1V2	3.5	250	210	A* 20	20	50	81.03
P2V2	7	250	420	A* 20	20	50	161.33
P3V2	14	250	840	A* 20	20	50	322.66

* - A – автомасштабируемость, M – ручное масштабирование.

** - 60 минут/день бесплатного использования

4. Веб-приложение на основе модели IaaS

Для реализации веб-приложения требуются виртуальные машины с установленным специальным программным обеспечением. Так как в состав приложения входит 6 независимых версий, база данных и http балансировщик, то для реализации необходимо использование 8 виртуальных машин и 8 скриптов, которые установят соответствующие программы.

Таким образом необходимо 6 Virtual machine: g1-small (1.7 Гб/1 ЦП/10 Гб). Инфраструктура приложения представлена на рисунке 9. Используя ее, а также рассмотренные ранее технологии была получена схема диверсного веб-приложения, спроектированного при помощи облачных сервисов модели представления IaaS (рис. 10).

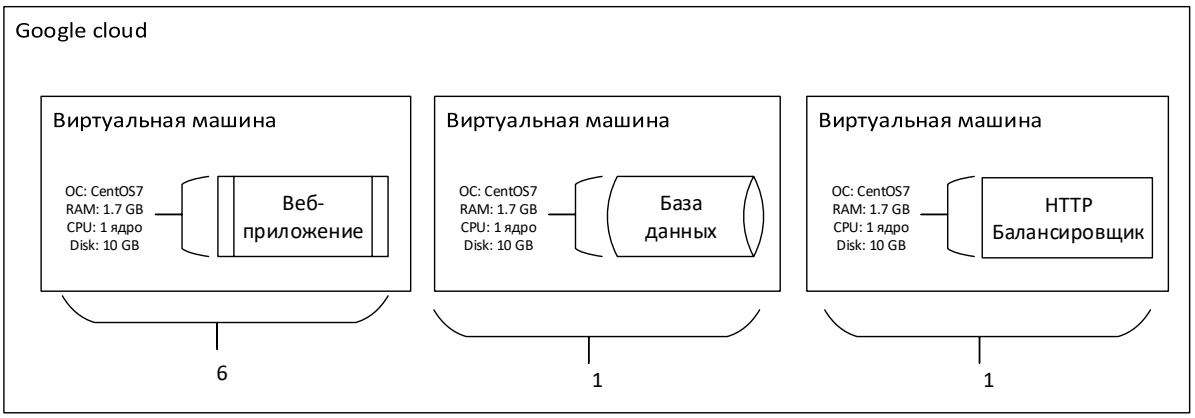

Рис. 9. Инфраструктура веб-приложения на модели IaaS

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

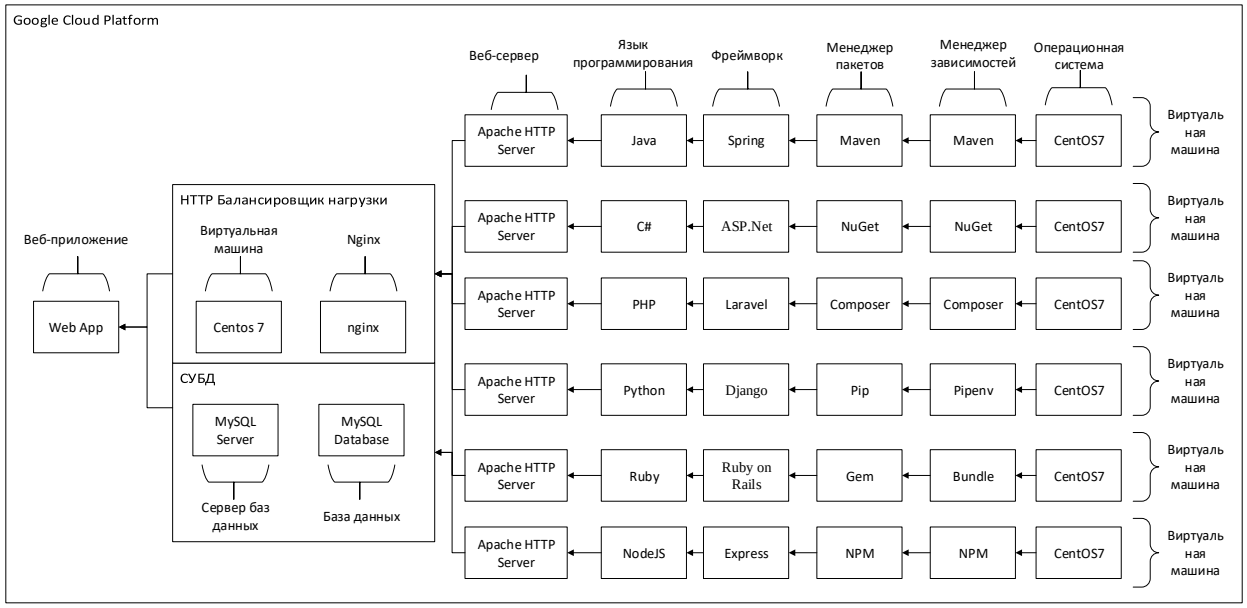

Рис. 10. Схема веб-приложения на основе виртуальных машине (IaaS)

5. Веб-приложение на основе модели PaaS

Веб-приложение может быть реализовано при помощи модели PaaS, в таком случае обеспечение инфраструктуры и аппаратного обеспечения для веб-приложения перекладывается на облачного провайдера. Пользователю необходимо только выбрать платформу, на которой будет работать приложение и загрузить его. Самые популярные облачные провайдеры предоставляют возможность использовать самые востребованные языки программирования. В качестве облачного провайдера для веб-сервисов был выбран Microsoft Azure, так как он единственный дает возможность использовать технологию ASP.NET, необходимую для реализации диверсного веб-приложения. А также, поскольку для реализации веб-приложения на основе модели PaaS используется GCP, то для повышения диверсности необходимо использовать другой облачный провайдер. Альтернативная реализация приложения выполняется с использованием веб-сервисов, готовых облачных решений модели PaaS. Инфраструктура приложения находится в Microsoft Azure и состоит из:

– Azure Application Gateway (тип: Standard V2, ручное масштабирование: 6 шт.);

– 6 App Service: Basic B1 (1.75 Гб/ 100 ACU/10 Гб);

– SQL БД: (5 DTU/1 Гб).

Инфраструктура приложения представлена на рисунке 11. Далее, используя полученную инфраструктуру и рассмотренные ранее технологии, была получена итоговая схема диверсного веб-приложения, спроектированного при помощи сервисов модели PaaS (рис. 12).

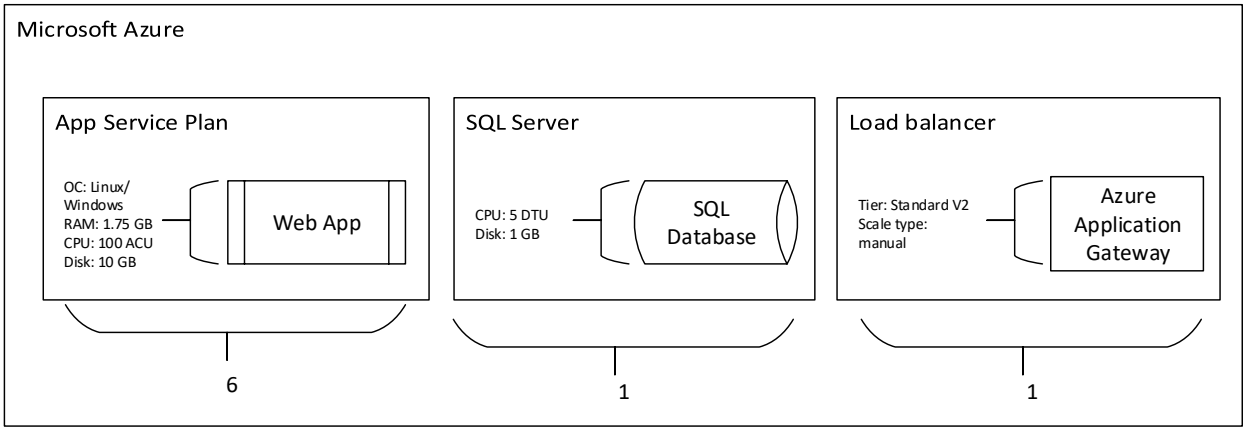

Рис. 11. Инфраструктура веб-приложения на модели PaaS

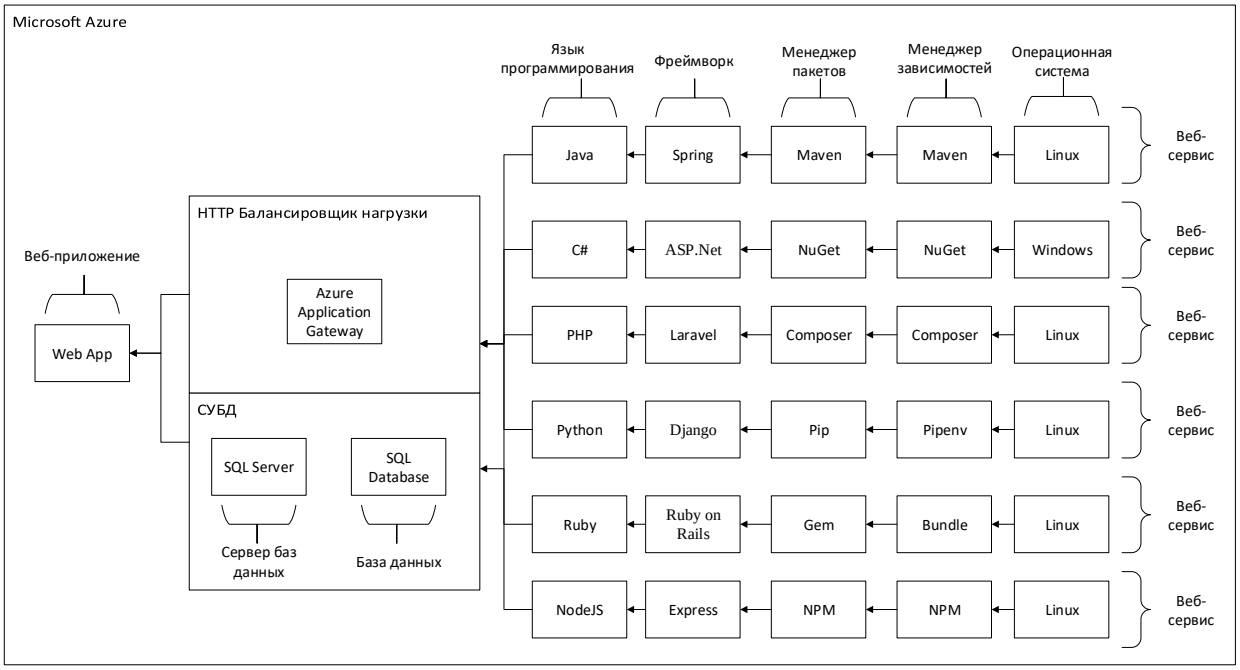

Рис. 12. Схема веб-приложения на основе веб-сервисов (PaaS)

Заключение

Повышение надежности и безопасности веб-приложений является чрезвычайно важной задачей и в условиях появления новых технологий она может быть решена значительно проще, для чего не требуется расходование большого количества средств. Облачные провайдеры предлагаю ряд решений, которые могут быть для этого полезными. Особенно стоит уделить внимания сервисам модели PaaS, так она позволяет проще и дешевле обеспечивать надежность и безопасность. Ведущие облачны провайдеры предоставляют возможность разрабатывать веб-приложения на самых популярных языках программирования при помощи модели PaaS.

Реализация принципа диверсности для веб-приложений с использованием облачных сервисов может стать очень удобным средством снижения вероятности его отказа по причине дефекта в программном коде. В результате анализа облачных сервисов, которые позволят реализовать принцип диверсности на практике, были получены схемы для различных моделей представления, с помощью которых можно спроектировать надежное веб-приложение.

Также был проведен сравнительный анализ различных видов реализаций, как с использованием диверсности, так и без. В ходе которого была осуществлена экспертная оценка, которая оценивала сложность реализации, ее стоимость и производительность готового веб-приложения, а также его информационную безопасность. Под стоимостью подразумевается оплата самой разработки, а также облачных сервисов, на которых функционируете итоговое веб-приложение.

Рассматривались варианты реализации веб-приложения при помощи облачных сервисов моделей представления IaaS и PaaS. При этом для сравнения были также взяты двухверсионные приложения, которые используют те же сервисы. Дополнительно предлагается разрабатывать приложение с одновременным использованием сервисов разных моделей представления, такой вариант реализации был назван смешанное приложение. Следовательно, возможно совместить смешанное приложение с концепцией двухверсионности. Таким образом получится диверсность разных видов и в результате ожидается повышенная безопасность. Данный вариант реализации был назван двухверсионное смешанное приложение.

По каждому критерию была выставлена оценка от низкой до высокой, а также промежуточные значения, которые находятся между двумя величинами, отмеченные плюсом. Оценки экспертов оценивались с учетом практики разработки подобных приложений и на ожиданиях, связанных с результатами, полученных на других системах и в других областях. Результаты сравнительного анализа представлены в таблице 4. В дальнейшем предполагается рассмотреть использование более точных оценок и подобрать дополнительные критерии, по которых можно сравнить варианты реализации.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Таблица 4

Сравнение вариантов реализации веб-приложений с применением различных видов диверсности

Критерий	Вариант реализации веб-приложения					
	IaaS приложение	Двухверсионное IaaS приложение	PaaS приложение	Двухверсионное PaaS приложение	Смешанное приложение	Двухверсионное смешанное приложение
Сложность реализации	Средняя	Средняя+	Низкая	Средняя+	Средняя+	Высокая
Стоимость	Средняя	Высокая	Низкая	Высокая	Средняя+	Высокая+
Безопасность	Низкая	Средняя	Низкая	Средняя	Средняя	Высокая
Производительность	Средняя+	Средняя	Высокая	Средняя+	Средняя	Низкая+

Как видно из полученных результатов, применение диверсности на уровне проектирования приложение, разработка независимых версий повышает безопасность итогового продукта, однако вместе с тем растёт и его стоимость и понижается производительность. Таким образом, это имеет смысл для критического программного обеспечения.

С другой стороны, диверсность облачных сервисов может также повысить безопасность, при меньшей затрате ресурсов, таким образом этот вид диверсности больше подходит для обычных веб-приложений. Касательно сложности реализации, то она растет при использовании любого вида диверсности приблизительно одинаково, однако при одновременном использование двух видов диверсности, сложность становится наиболее высокой.

В дальнейшем следует провести эксперимент по оценке влияния диверсности облачных сервисов на надежность и информационную безопасность системы, в которой они применяются. Также следует более подробно изучить возможность одновременного использования сервисов моделей IaaS и PaaS, для повышения диверсности итоговой системы, так как такой подход выглядит наиболее предпочтительным по совокупности критериев, рассмотренных в данном сравнительном анализе.

Литература

1. AWS security best practices [Electronic resource]. – Access mode: https://aws.amazon.com/whitepapers/aws-security-best-practices/ – 22.02.2020.

2. Microsoft Azure security best practices [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/azure/security/security-best-practices-and-patterns – 22.02.2020.

3. Google cloud platform security best practices [Electronic resource]. – Access mode: https://cloud.google.com/docs/enterprise/best-practices-for-enterprise-organizations – 22.02.2020.

4. Горбенко, А. В. Анализ особенностей создания и эксплуатации гарантоспособных сервис-ориентированных систем [Текст] / А. В. Горбенко. Радіоелектронні і комп'ютерні системи. – 2013. – № 5(64). – С. 237-242.

5. Yastrebenetsky, M. Nuclear Power Plant Instrumentation and Control Systems for Safety and Security [Text] / M. Yastrebenetsky, V. Kharchenko. – IGI Global, USA, 2014. – 450 p.

6. Frolov, V. Classification of Diversity for Dependable and Safe Computing [Electronic resource] / V. Frolov, O. Frolov, V. Kharchenko // COLINS, 2019. – Access mode: http://ceur-ws.org/Vol-2362/paper32.pdf – 22.12.2019.

7. Randles, M. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing [Text] / M. Randles, D. Lamb, A. Taleb-Bendiab // 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, WA, 2010. – P. 551-556.

8. Zhao, Q. A Load Balancing Based Model for Dynamic Web Service Selection [Text] / Q. Zhao, Y. Tan // Second International Symposium on Computational Intelligence and Design, Changsha, 2009. – P. 501-505.

9. Hong, Y. S. DNS-based load balancing in distributed Web-server systems [Text] / Y. S. Hong, J. H. No, S. Y. Kim // The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06), Gyeongju, 2006. – P. 4.

10. Khwaja, S. A framework for evaluating software design pattern specification languages [Text] / S. Khwaja, M. Alshayeb // IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS), Niigata, 2013. – P. 41-45.

11. Alrouh, B. A Performance Evaluation of Security Mechanisms for Web Services [Text] / B. Alrouh, G. Ghinea // Fifth International Conference on Information Assurance and Security, Xi'an, 2009. – P. 715-718.

12. Analysis and suggestions for the security of web applications [Text] / You Yu, Yuanyuan Yang, Jian Gu, Liang Shen // Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, 2011. – P. 236-240.

13. Yadav, D. Vulnerabilities and Security of Web Applications [Text] / D. Yadav, D. Gupta, D. Singh, D. Kumar, U. Sharma // 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018. – P. 1-5.

14. What Is Load Balancing [Electronic resource]. – Access mode: https://www.nginx.com/resources/glossary/load-balancing/ – 20.02.2020.

15. Using nginx as HTTP load balancer Balancing [Electronic resource]. – Access mode: http://nginx.org/en/docs/http/load_balancing.html – 21.02.2020.

16. Understanding software design patterns [Electronic resource]. – Access mode: https://opensource.com/article/19/7/understanding-software-design-patterns – 24.02.2020.

17. Design Patterns [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/design-patterns-set-1-introduction/ – 21.02.2020.

18. Фролов, В. В. Анализ подходов к обеспечению безопасности облачных сервисов [Текст] / В. В. Фролов // Радіоелектронні і комп'ютерні системи. – 2020. – № 1 (93). – С. 70-82. DOI: 10.32620/reks. 2020.1.07.

19. Microsoft Azure marketplace [Electronic resource]. – Access mode: https://azuremarketplace.microsoft.com – 22.02.2020.

References

1. *AWS security best practices.* Available at: https://aws.amazon.com/whitepapers/aws-security-best-practices/ (accessed 22.02.2020).

2. *Microsoft Azure security best practices.* Available at: https://docs.microsoft.com/en-us/azure/security/security-best-practices-and-patterns (accessed 22.02.2020).

3. *Google cloud platform security best practices.* Available at: https://cloud.google.com/docs/enterprise/best-practices-for-enterprise-organizations (accessed 22.02.2020).

4. Gorbenko, A. Analiz osobennostei sozdaniya i ekspluatatsii garantosposobnykh servis-orientirovannykh sistem [Analysis of dependable service-oriented systems development features]. *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2013, no. 5(64), pp. 237-242.

5. Yastrebenetsky, M., Kharchenko, V. *Nuclear Power Plant Instrumentation and Control Systems for Safety and Security*. IGI Global, USA, 2014. 450 p.

6. Frolov, V., Frolov, O., Kharchenko, V. *Classification of Diversity for Dependable and Safe Computing. COLINS, 2019*. Available at: http://ceur-ws.org/Vol-2362/paper32.pdf (accessed 22.12.2019).

7. Randles, M., Lamb, D., Taleb-Bendiab, A. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. *24th IEEE International Conference on Advanced Information Networking and Applications Workshops*, Perth, WA, 2010, pp. 551-556. DOI: 10.1109/WAINA.2010.85.

8. Zhao, Q., Tan, Y. A Load Balancing Based Model for Dynamic Web Service Selection. *Second International Symposium on Computational Intelligence and Design*, Changsha, 2009, pp. 501-505. DOI: 10.1109/ISCID.2009.132.

9. Hong, Y. S., No, J. H., Kim, S. Y. DNS-based load balancing in distributed Web-server systems. *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, Gyeongju, 2006, pp. 4. DOI: 10.1109/SEUS-WCCIA.2006.23.

10. Khwaja, S., Alshayeb, M. A framework for evaluating software design pattern specification languages. *IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS)*, Niigata, 2013, pp. 41-45. DOI: 10.1109/ICIS.2013.6607814.

11. Alrouh, B., Ghinea, G. A Performance Evaluation of Security Mechanisms for Web Services. *Fifth International Conference on Information Assurance and Security*, Xi'an, 2009, pp. 715-718. DOI: 10.1109/IAS.2009.252.

12. You, Yu., Yuanyuan, Yang., Jian, Gu., Liang, Shen. Analysis and suggestions for the security of web applications. *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Harbin, 2011, pp. 236-240. DOI: 10.1109/ICCSNT.2011.6181948.

13. Yadav, D., Gupta, D., Singh, D., Kumar, D., Sharma, U. Vulnerabilities and Security of Web Applications. *4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India, 2018, pp. 1-5. DOI: 10.1109/CCAA.2018.8777558.

14. *What Is Load Balancing*. Available at: https://www.nginx.com/resources/glossary/load-balancing/ (accessed 20.02.2020).

15. *Using nginx as HTTP load balancer Balancing.* Available at: http://nginx.org/en/docs/http/load_balancing.html (accessed 21.02.2020).

16. *Understanding software design patterns.* Available at: https://opensource.com/article/19/7/understanding-software-design-patterns (accessed 24.02.2020).

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

17. *Design Patterns.* Available at: https://www.geeksforgeeks.org/design-patterns-set-1-introduction/ (accessed 21.02.2020).

18. Frolov, V. Analiz podkhodov k obespecheniyu bezopasnosti oblachnykh servisov [Analysis of approaches providing security of cloud services]. *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2020, no.1(93), pp. 70-82. DOI: 10.32620/reks.2020.1.07.

19. *Microsoft Azure marketplace.* Available at: https://azuremarketplace.microsoft.com (accessed 22.02.2020).

Поступила в редакцию 24.03.2020, рассмотрена на редколлегии 15.04.2020

АНАЛІЗ ВАРІАНТІВ ДВОХВЕРСІЙНОГО БАГАТОМОДУЛЬНОГО ВЕБ-ДОДАТКА З ВИКОРИСТАННЯМ ХМАРНИХ СЕРВІСІВ

В. В. Фролов, А. А. Орехов, В. С. Харченко, А. В. Фролов

Стаття присвячена аналізу варіантів двохверсійних веб-додатків із використанням хмарних сервісів. Оскільки проектування і розробка веб-додатків ведеться все більш активно, виникає необхідність підвищувати їх надійність в умовах підвищення складності самих додатків та інфраструктури на яких вони базуються. Одним з ключових рішень даної проблеми є використання хмарних сервісів, які дозволяють значно спростити задачу забезпечення надійності та безпеки різних додатків. При цьому хмарні провайдери не можуть повністю гарантувати відмовостійкість додатків, які запускаються в їх середовищі. Таким чином користувачі повинні самі потурбуватися про це. Одним з найбільш перспективних підходів є застосування диверсності для підвищення безпеки і надійності веб-додатків, розміщених в хмарах. Об'єктом дослідження і аналізу даної роботи є багатомодульний веб-додаток, спроектований за допомогою хмарних сервісів. Метою дослідження даної роботи є аналіз та розробка структур варіантів двохверсійних багатомодульних веб-додатків із використанням хмарних сервісів. Так як безліч компаній переносять свою інфраструктуру в хмари, то виникає необхідність розглянути можливість застосування диверсності за допомогою хмарних сервісів. Вони дозволяють створити і розгорнути веб-додатки, розроблені на різних мовах програмування на серверах хмарних провайдерів. Таким чином частина відповідальності за забезпечення надійності перекладається на них. Однак необхідно як і раніше забезпечувати відмовостійкість власних програм, які можуть відмовити через дефекти в програмному коді. Одним з основних рішень цієї проблеми є N-версійне програмування, яке дозволяє створювати додаток з декількох незалежних версій. Кожна версія може бути написана на різних мову програмування і з використанням різних технологій окремими командами розробників, тим самим підвищуючи надійність кінцевого програмного продукту. Як результат, в даній роботі робиться висновок про те, що провідні хмарні провайдери надають можливість реалізувати диверсність на практиці за допомогою сервісів різних моделей уявлення, таких як IaaS і PaaS. Використовуючи принцип диверсності можна спроектувати надійний веб-додаток, який дозволить уникнути його відмови в разі помилки в програмному коді.

Ключові слова: хмарні сервіси; диверсність веб-додатків; хмарні провайдери; модель надання хмарних сервісів; відмова з загальної причини; N-версійне програмування; забезпечення надійності та безпеки програмного продукту; двоверсійний програмний додаток.

ANALYSIS OF VARIANT OF TWO-VERSION MULTI-MODULE WEB APPLICATION USING CLOUD SERVICES

V. Frolov, A. Orehov, V. Kharchenko, A. Frolov

The article is devoted to the analysis of a variant of two-version multi-module web application using cloud services. As the design and development of web applications are increasingly active, there is a need to increase their reliability in the face of the increasing complexity of the applications themselves and the infrastructure on which they are based. One of the key solutions to this problem is the use of cloud services, which can greatly simplify the task of ensuring the reliability and security of various applications. At the same time, cloud providers cannot fully guarantee the fault tolerance of applications that run in their environment. Therefore, users should worry about this themselves. One of the most promising approaches is the use of diversity to increase the security and reliability of web applications hosted in the clouds. The object of research and analysis of this work is a multi-module web application designed

using cloud services. The study of this work aims to compare modern solutions and technologies that allow implementing sabotage for a web application. Since many companies are moving their infrastructure to the clouds, it becomes necessary to consider the possibility of using diversity by cloud services. They allow you to create and deploy web applications developed in various programming languages on the servers of cloud providers. Thus, part of the responsibility for ensuring reliability is transferred to them. However, it is still necessary to ensure the resiliency of your programs, which may fail due to defects in the program code. One of the main solutions to this problem is N-version programming, which allows you to create an application from several independent versions. Each version can be written in different programming languages and using various technologies by separate development teams, thereby increasing the reliability of the final software product. As a result, in this paper, we conclude that leading cloud providers provide the opportunity to implement diversity using services of various presentation models, such as IaaS and PaaS. Using the principle of diversity, you can design a reliable web application that will avoid its failure in case of an error in the program code.

Keywords: cloud services; multi-cloud strategy; cloud security approach; diversity; cloud providers; cloud service delivery model; cloud deployment model; common cause failure; cloud security threats.

Фролов Вячеслав Вікторович – аспірант, асистент кафедри комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Орєхов Олександр Олександрович – канд. техн. наук, доцент, професор кафедри 503, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Харченко Вячеслав Сергійович – д-р техн. наук, професор, завідувач кафедрою 503, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Фролов Олександр Вікторович – аспірант, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Viacheslav Frolov – PhD student, assistant lecturer of Computer Systems, Networks and Cybersecurity department, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: v.frolov@csn.khai.edu, ORCID Author ID: 0000-0002-5860-7193.

Oleksandr Orehov – PhD in Technology, Associate Professor, Professor of Department 503, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: a.orehov@csn.khai.edu.

Vyacheslav Kharchenko – Doctor of Science on Engineering, Professor, Head of Department 503, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: v.kharchenko@khai.edu, ORCID Author ID: 0000-0001-5352-077X.

Oleksandr Frolov – PhD student, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: o.frolov@csn.khai.edu, ORCID Author ID: 0000-0002-0230-0790.