УДК 512.6, 517.9, 519.6

# ОРКЕСТРАЦИЯ КОНТЕЙНЕРОВ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ В СИСТЕМЕ РАСПОЗНАВАНИЯ ЛИЦ

**Филипьев Андрей Владимирович[1], Махалкина Татьяна Олеговна[2], Дмитриев Дмитрий Сергеевич[3]**

*[1]Ассистент;*
*ГБОУ ВО МО «Университет «Дубна»,*
*Институт системного анализа и управления;*
*141980, Московская обл., г. Дубна, ул. Университетская, 19;*
*e-mail: avfilipev@gmail.com.*

*[2]Старший преподаватель;*
*ГБОУ ВО МО «Университет «Дубна»,*
*Институт системного анализа и управления;*
*141980, Московская обл., г. Дубна, ул. Университетская, 19;*
*e-mail: mahalkina@gmail.com.*

*[3]Студент;*
*ГБОУ ВО МО «Университет «Дубна»,*
*Институт системного анализа и управления;*
*141980, Московская обл., г. Дубна, ул. Университетская, 19;*
*e-mail: dds98@mail.ru.*

*В данной статье решалась задача применения технологии оркестрации контейнеров при решении одной из задач распознавания образов, а именно распознавания лиц. Благодаря тому, что технология распознавания лиц доросла до такого уровня, что ее можно использовать в коммерческих целях, многие компании начинают активно внедрять подобные решения. Использование микросервисной архитектуры позволило добиться независимого развертывания и масштабирования каждого сервиса, а также получить четкую физическую границу между сервисами. Для создания более безопасного приложения, облегчения развертывания, улучшения возможности масштабирования и управления нагрузками были использованы технологии контейнеризации и оркестрации.*

Ключевые слова: оркестрация, контейеризация, распознавание, микросервисы, масштабирование.

# ORCHESTRATION OF CONTAINERS OF MICROSERVICE ARCHITECTURE IN FACE RECOGNITION SYSTEM

**Filipiev Andrey[1], Makhalkina Tatyana[2], Dmitriev Dmitry[3]**

*[1]Assistant;*
*Dubna State University,*
*Institute of the system analysis and management;*
*141980, Dubna, Moscow reg., Universitetskaya str., 19;*
*e-mail: avfilipev@gmail.com.*

*[2]Senior teacher;*
*Dubna State University,*
*Institute of the system analysis and management;*
*141980, Dubna, Moscow reg., Universitetskaya str., 19;*
*e-mail: mahalkina@gmail.com.*

*[3]Student;*
*Dubna State University,*
*Institute of the system analysis and management;*
*141980, Dubna, Moscow reg., Universitetskaya str., 19;*
*e-mail: dds98@mail.ru.*

*This paper solved the problem of applying the technology of containers orchestration in solving one of the problems of pattern recognition, namely face recognition. Due to the fact that face recognition technology has grown to such a level that it can be used for commercial purposes, many companies are beginning to actively implement such solutions. The use of microservice architecture allowed independent deployment and scaling of each service, as well as a clear physical boundary between the services. Containerization and orchestration technologies were used to create a more secure application, facilitate deployment, improve scalability and load management.*

Keywords: orchestration, containerization, recognition, microservices, scaling.

## *Введение*

«Микросервисы» – относительно новый термин в разработке программного обеспечения (далее – ПО). Этот термин описывает стиль разработки ПО, который разработчики находят все более и более привлекательным [1].

Архитектурный стиль микросервисов – это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными, используя легковесные механизмы, как правило *HTTP* [1]. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

Монолит – приложение, построенное как единое целое. Любое изменение в системе приводит к пересборке и развертыванию новой версии серверной части приложения. Их достаточно сложно поддерживать и развивать, со временем становиться труднее сохранять хорошую модульную структуру, изменения логики одного модуля имеют тенденцию влиять на код других модулей. Масштабировать приходиться все приложение целиком, даже если это требуется только для одного модуля этого приложения. Эти неудобства привели к архитектурному стилю микросервисов: построению приложений в виде набора сервисов.

Приложение, разработанное при таком подходе, может решать достаточно трудоемкие задачи, ведь имеется возможность использования преимуществ различных языков программирования и платформ. В данной статье рассматривается задача применения микросервисного подхода при разработке приложения, решающего задачу распознавания персоны по лицу.

## *Распознавание лиц*

Сегодня распознавание лиц не просто теоретическая задача или футуристическая фантазия – это удобная и практическая функция идентификации в системе без пароля. Сама технология относится к области применения теории распознавания образов, которая возникла значительно раньше современных компьютерных систем. Современный уровень развития IT позволяет воплощать модели распознавания образов в жизнь. Поскольку технология распознавания лиц уже дошла до такого уровня готовности, что ее можно использовать в коммерческих проектах, многие компании внедряют подобные решения.

Последовательность действий при распознавании лица обычно такова:

1. Выделяется лицо человека на изображении (рис. 1).
2. Вычисляются антропометрические точки. Система находит опорные точка на лице, которые определяют индивидуальные характеристики. Алгоритм вычисления характеристик различен для каждой из систем и является главным секретом разработчиков. Раньше основной опорной точкой для алгоритмов были глаза, но алгоритмы эволюционировали и стали учитывать минимум 68 точек на лице (расположены по контуру лица, определяют положение и форму подбородка, глаз, носа и рта, расстояние между ними) (рис. 1).
3. Проводятся дополнительные преобразования изображения (устранение наклона головы и так далее) с целью получения четкого фронтального снимка (рис. 1).

4. Вычисляется эмбеддинг – набор характеристик, описывающих лицо независимо от посторонних факторов (возраст, прическа, макияж). Анализируются специальные локальные признаки, характеризующие, например, текстуру определенных областей на лице. Сопоставление разных эмбеддингов позволяет оценить, относятся ли два полученных изображения лица к одному и тому же человеку (рис. 2).
5. Сравнивается полученный эмбеддинг (вектор лица) с имеющимся в базе эмбеддингами.

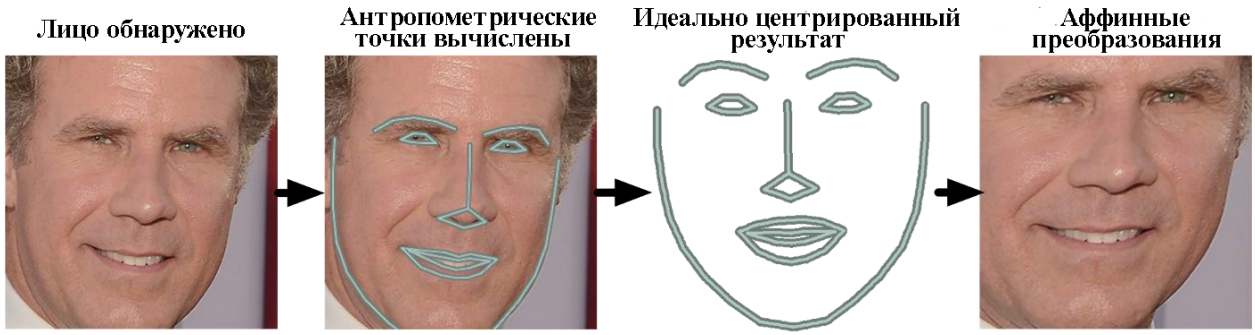


*Рис. 1. Демонстрация пунктов 1, 2 и 3*

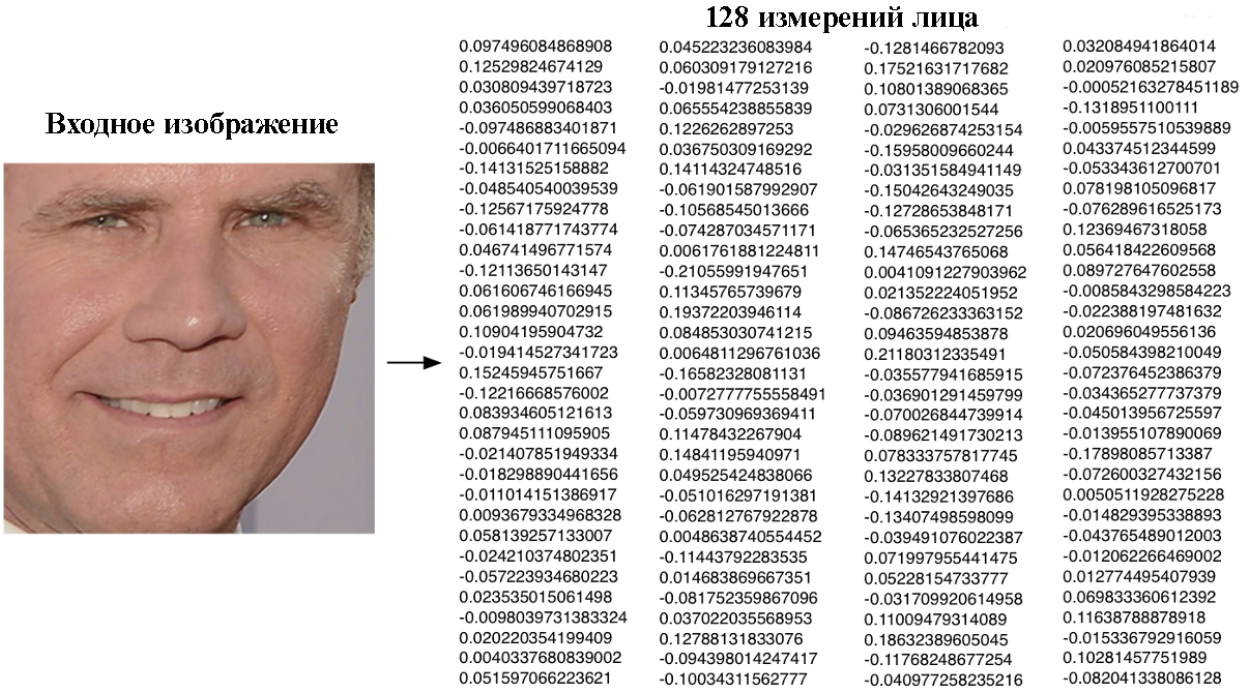


*Рис. 2. Демонстрация пункта 4*

Чтобы реализовать приложение, способное корректно выполнять все вышеописанные действия и не потратить при этом значительное количество времени, для изучения и применения была выбрана технология контейнеризации.

## *Docker*

Технологии контейнеризации приложений нашли широкое применение в сферах разработки ПО и анализа данных. Эти технологии помогают сделать приложения более безопасными, облегчают их развертыванию и улучшают возможности по их масштабированию. Рост и развитие технологий контейнеризации можно считать одним из важнейших трендов современности.

*Docker* – это платформа, которая предназначена для разработки, развертывания и запуска приложений в контейнерах. Слово «*Docker*» в последнее время стало чем-то вроде синонима слова «кон-

тейнеризация». И если разработчики еще не используют *Docker*, то *Docker* – это то, с чем они непременно встретятся в будущем.

Контейнер – это стандартная единица программного обеспечения, которая упаковывает код и все его зависимости, поэтому приложение работает быстро и надежно из одной вычислительной среды в другую. Образ контейнера *Docker* – это легкий, автономный исполняемый пакет программного обеспечения, который включает в себя все необходимое для запуска приложения: код, среду выполнения, системные инструменты, системные библиотеки и настройки.

Изображения контейнеров становятся контейнерами во время выполнения. Контейнерное программное обеспечение, доступное как для *Linux*, так и для *Windows*, всегда будет работать одинаково, независимо от инфраструктуры. Контейнеры изолируют программное обеспечение от его среды и обеспечивают его равномерную работу, несмотря на различия, например, между разработкой и подготовкой

## *Kubernetes*

*Kubernetes* – это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема. Сервисы, поддержка и инструменты *Kubernetes* широко доступны [2].

Контейнеры – отличный способ связать и запустить приложения. В производственной среде необходимо управлять контейнерами, которые запускают приложения, и гарантировать отсутствие простоев. Например, если контейнер выходит из строя, необходимо запустить другой контейнер.

*Kubernetes* дает фреймворк для гибкой работы распределенных систем. Он занимается масштабированием и обработкой ошибок в приложении, предоставляет шаблоны развертывания и многое другое.

## *Разработка*

Функциональные требования к приложению были сформулированы следующим образом:

- Система должна обеспечивать хранение персон для распознавания.
- Система должна позволять добавлять персону.
- Система должна позволять удалять ранее созданную персону.
- Система должна хранить и отображать фотографию и другую информацию персоны.
- Система должна иметь пользовательский интерфейс, с помощью которого можно проводить распознавания персон, добавлять персоны и удалять их.
- Система дает возможность проведения распознавания персоны по фотографии.

С точки зрения пользования приложением – оно не должно казаться сложным для пользователя (рис. 3).


*Рис. 3. Основные действия пользователя*

Также вся работа сервисов не должна быть заметна пользователю при работе с системой, когда исполняется сценарий непосредственного распознавания лица (рис. 4).

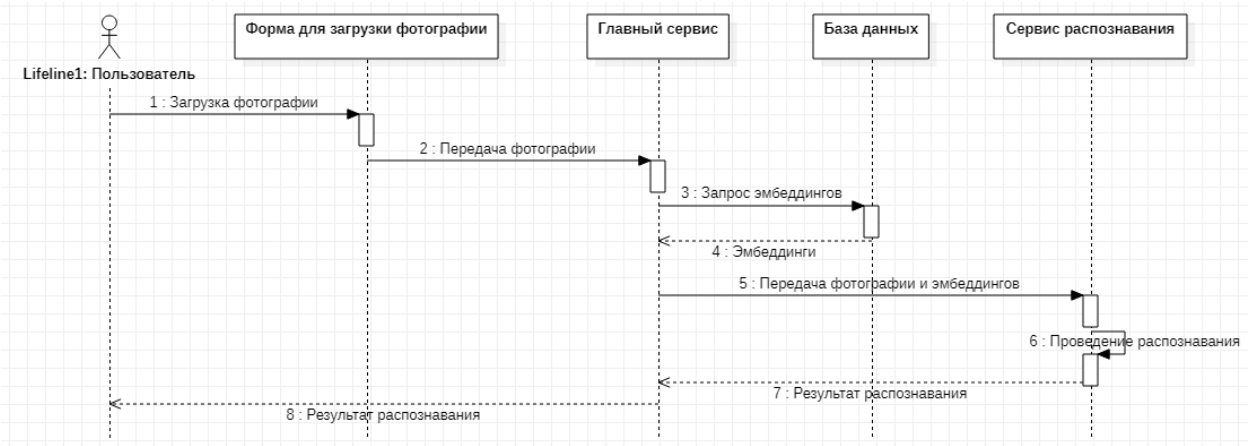


*Рис. 4. Диаграмма последовательности сценария «Проведение распознавания персоны по фотографии»*

Модули приложения были реализованы как микросервисы, которые были развернуты в кластере, управляемом *Kubernetes*. С технической точки зрения приложение состоит из четырех микросервисов и базы данных (рис. 3):

- *FR-Frontend* – веб-сервер *Nginx*, который обслуживает статические файлы *React*.
- *FR-WebApp* – веб-приложение, написанное на *Java*, которое обрабатывает запросы от фронтенда и взаимодействует с базой данных.
- *FR-Logic* – *Python*-приложение, которое выполняет анализ изображения и возвращает эмбеддинг лица на изображении и выполняет распознавание по лицу.
- *FR-Storage* – *Python*-приложение, которое выполняет функции управления фотографиями, хранимых в файловой системе.
- *FR-DB* – *MySQL* база данных.

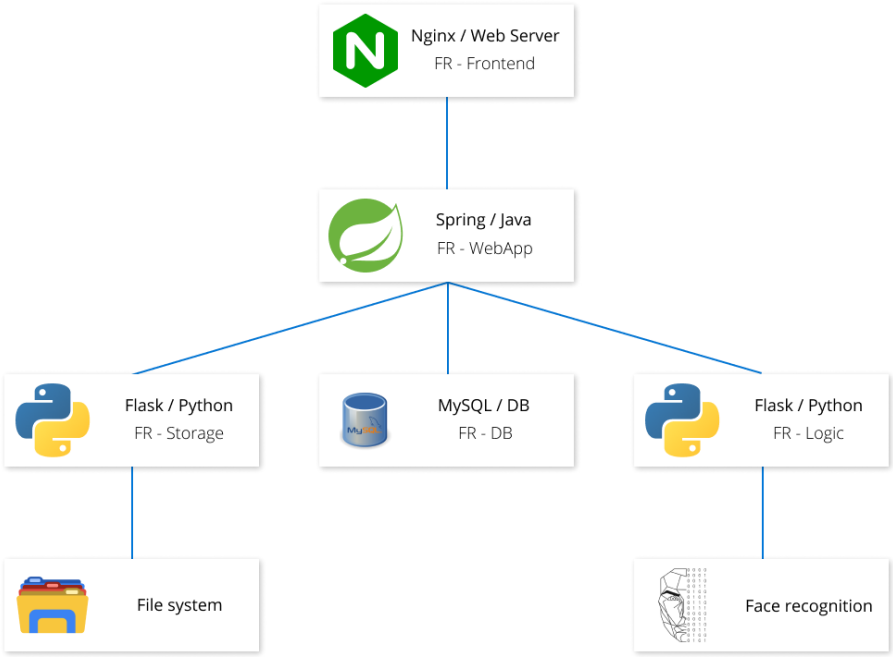


*Рис. 5. Структура приложения*

*Kubernetes* работает с контейнерами поэтому прежде чем развернуть микросервисы в кластере необходимо было создать контейнеры используя готовые микросервисы. Основным строительным блоком контейнера *Docker* являются *Dockerfile*. В нем необходимо указывать информацию о базовом образе контейнера и описывать последовательность создания самого контейнера. В последовательности создания контейнера обычно ведется подготовка окружения и установка необходимых инструментов. После создания контейнеров можно приступить к разворачиванию их на кластере *Kubernetes*.

Сервисы *Kubernetes* играют роль точек доступа к наборам подов, которые предоставляют тот же функционал, что и эти поды. Сервисы выполняют решение непростых задач по работе с подами и по балансировке нагрузки между ними. Создание и настройка сервиса достаточно проста, нам необходимо подготовить *YAML*-описание сервиса, которое включает в себя название сервиса, тип ресурса, метку для подов и другие сетевые настройки сервиса.

В данном кластере *Kubernetes* должны быть поды, реализующие разные функции. Это – *frontend*-приложение, веб-приложение *Spring* и два *Flask*-приложения. Для их развертывания нам необходимо подготовить *YAML*-описание ресурса вида *Deployment*, которое включает в себя название ресурса, количество реплик подов для запуска, название стратегии развертывания, максимальное количество недоступных подов, максимальное число подов, которое можно добавить в развертывание, метку для подов и множество других настроек.

Развертывание – это абстракция *Kubernetes*, которая позволяет нам управлять тем, что всегда присутствует в жизненном цикле приложения.

После выполнения развертываний и создания сервисов система распознавания лиц (рис. 6) полностью функционирует и готова к использованию.

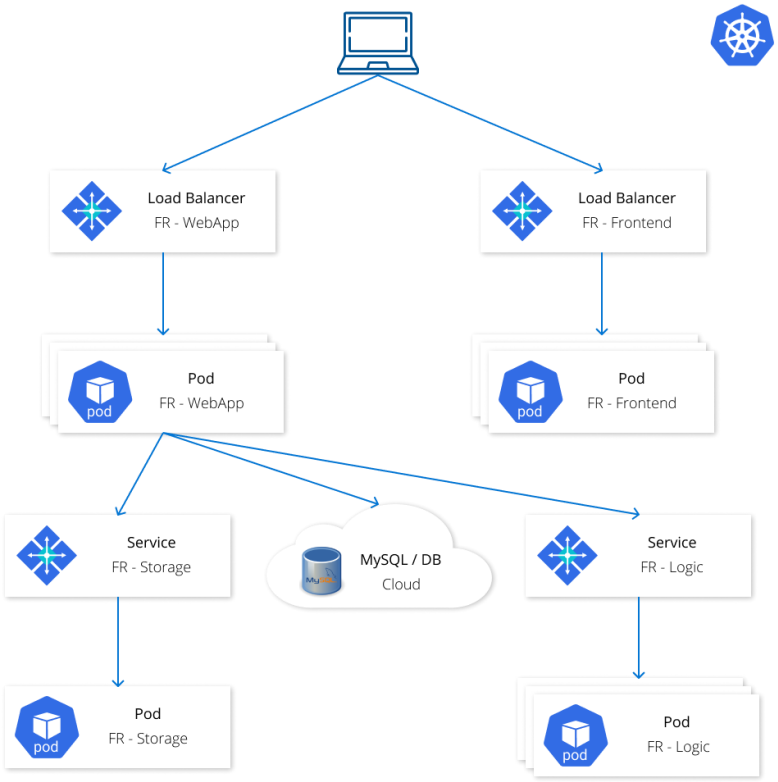


*Рис. 6. Готовое кластерное приложение*

## *Заключение*

На основе микросервисной архитектуры была реализована система распознавания лиц, содержащая четыре микросервиса, написанные на разных языках программирования и состоящие из разных фреймворков: *frontend*-приложение, веб-приложение, микросервис распознавания лиц и микросервис управления фотографиями.

Созданные микросервисы были успешно «упакованы» в контейнеры, для дальнейшего их распространения и использования.

Был создан кластер *Kubernetes*, где были успешно развернуты контейнеры с микросервисами, сервисы для работы с наборами контейнеров.

Использование технологии *Kubernetes* упростило развертывание микросервисов, из которых состоит система распознавания лиц, помогла их масштабировать, сделает их устойчивыми к сбоям. Благодаря *Kubernetes* можно пользоваться ресурсами, предоставляемыми самыми разными облачными провайдерами и при этом не зависть от решений конкретных поставщиков облачных услуг

Программный код находится в открытом доступе по ссылке [3].

## *Список литературы*

1. *Martin Fowler, James Lewis Microservices*. — [Электронный ресурс]. *URL*: https://martinfowler.com/articles/microservices.html.
2. *Kubernetes. Documentation.* — [Электронный ресурс]. *URL*: https://kubernetes.io/ru/docs/concepts/overview/what-is-kubernetes/.
3. Репозиторий *Face recognition*. — [Электронный ресурс]. *URL*: https://github.com/Virtual08/Face_recognition.